Akpojoto Akporido Siemuri

Adaptive Localization Using Machine Learning

Models, Methods, and Applications

ACTA WASAENSIA 578

University of Vaasa
VAASAN YLIOPISTO

ISBN 978-952-395-250-8 (print)
978-952-395-251-5 (online)

ISSN 0355-2667 (Acta Wasaensia 578, print)
2323-9123 (Acta Wasaensia 578, online)

URN https://urn.fi/URN:ISBN:978-952-395-251-5

PunaMusta Oy, Joensuu, 2026.

ACADEMIC DISSERTATION

To be presented, with the permission of the Board of the School of Technology and Innovation at the University of Vaasa, for public examination on 23rd of March 2026 at noon.

Article based dissertation, School of Technology and Innovations at the University of Vaasa, in the feld of automation technology.

Author	Akpojoto Akporido Siemuri https://orcid.org/0000-0002-2644-1985
Supervisor(s)	Professor Mohammed Elmusrati School of Technology and Innovations University of Vaasa
	Professor Heidi Kuusniemi Faculty of Information Technology and Communications University of Tampere
	Professor Petri Valisuo School of Technology and Innovations University of Vaasa
Custos	Professor Mohammed Elmusrati School of Technology and Innovations University of Vaasa
Reviewers	Naser Tarhuni, Associate Professor Department of Electrical and Computer Engineering Sultan Qaboos University Sultanate of Oman
	Joan Solà Ortega Institut de Robòtica i Informàtica Industrial Tenured scientist, CSIC-UPC Barcelona
Opponent	Professor Johan Lilius Faculty of Science and Technology, Information Technology Åbo Akademi https://orcid.org/0000-0002-9176-2881

TIIVISTELMÄ

Tämä väitöskirja tutkii koneoppimisen (ML) tekniikoiden käyttöä paikannustarkkuuden parantamiseksi käyttämällä edullisia antureita. Viimeaikaiset edistysaskeleet edullisissa kaksitaajuisissa GNSS-vastaanottimissa ja edulliset GNSS-signaalinkorjauspalvelut ovat luoneet uusia mahdollisuuksia massamarkkinoiden tarkkojen sijaintiperusteisten sovellusten kehittämiselle. Mainittuihin teknologioihin kuuluvat GNSS, ultralaajakaista (UWB), Bluetooth, 5G, tutka, inertianavigointijärjestelmät (INS), näköön perustuva paikannus ja erilaiset GNSS-korjauspalvelut. Koneoppimisen hyödyntämistä GNSS-sovelluksissa tarkasteltiin systemaattisesti. Katsaukseen sisältyi sarja kokeellisia tutkimuksia kaupallisen luokan GNSS-vastaanottimilla, mukaan lukien älypuhelinpohjaisilla sensoreilla, joita käytettiin kustannustehokkaiden ja luotettavien paikannusjärjestelmien kehittämiseen. Näitä ovat: (i) pelkästään GNSS:ään perustuva järjestelmä, jota on parannettu koneoppimiseen perustuvalla adaptiivisella paikannuksella (MAP), (ii) GNSS/IMU-ratkaisu, joka hyödyntää Rauch–Tung–Striebel (RTS) -tasoitusta tarkkuuden parantamiseksi kaupunkiympäristöissä, ja (iii) saumaton sisä- ja ulkotilojen paikannusjärjestelmä, joka yhdistää GNSS-, UWB- ja IMU-anturit. Ehdotetut menetelmät osoittivat kilpailukykyistä suorituskykyä globaaleissa vertailuissa, kuten Google Smartphone Decimeter Challengessa. Lisäksi tutkitaan vaihtoehtoista GNSS/IMU-fuusiomenetelmää, joka käyttää tekijägraafioptimointia (FGO), ja jolla saavutetaan parempi suorituskyky signaalin heikkenemisen tilanteissa.

Väitöskirja ulottuu myös avaruussovelluksiin käyttämällä koneoppimismalleja matalan Maan kiertoradan (LEO) satelliittien tarkkaan kiertoradan määritykseen (POD), mikä osoittaa polynomiregression edun epälineaarisen kiertoradan dynamiikan mallintamisessa. Lisäksi tutkitaan Edge AI -sovelluksia ottamalla käyttöön TinyML-malleja laitteen paikannukseen, korostaen toteutettavuutta rajoitetuissa ympäristöissä ja ottaen samalla huomioon energiatehokkuuden, yksityisyyden ja reaaliaikaisen sopeutumiskyvyn. Tekoälyn/koneoppimisen eettisiä vaikutuksia lokalisoinnissa tarkastellaan EU:n tekoälylain kontekstissa painottaen oikeudenmukaisuutta, läpinäkyvyyttä ja vastuullista tekoälyn hallintaa. Yhdessä nämä tutkimukset osoittavat koneoppimisen ja TinyML:n käytännöllisen, skaalautuvan ja eettisen käytön tarkkaan ja luotettavaan lokalisointiin erilaisissa käyttötapauksissa ja ympäristöissä.

Avainsanat: globaalit satelliittipaikannusjärjestelmät (GNSS), edulliset anturit, saumaton navigointi, koneoppiminen (ML), tekijägraafien optimointi (FGO), tarkka kiertoradan määritys (POD), TinyML, eettinen tekoäly (AI)

ABSTRACT

This dissertation investigates how machine learning (ML) techniques can enhance positioning accuracy when using low-cost sensors. Recent advances in affordable dual-frequency GNSS receivers and accessible GNSS correction services have opened new possibilities for developing accurate, mass-market location-based applications. These technologies include dual-frequency GNSS, Ultra-Wideband (UWB), Bluetooth direction finding, 5G, radar, inertial navigation systems (INS), image-based localization, and various GNSS correction services. A systematic review was conducted on the integration of ML in GNSS-related application, highlighting key trends, algorithmic performance, and associated challenges. A series of experimental studies were performed using commercial-grade GNSS receivers—including smartphone-based sensors—to develop cost-effective and robust positioning systems. These include: (i) a GNSS-only system enhanced with ML-based adaptive positioning (MAP), (ii) a GNSS/IMU solution utilizing Rauch–Tung Striebel (RTS) smoothing for improved accuracy in urban environments, and (iii) a seamless indoor–outdoor localization system combining GNSS, UWB, and IMU sensors. The proposed methods demonstrated competitive performance in global benchmarks, such as the Google Smartphone Decimeter Challenge. Additionally, an alternative GNSS/IMU fusion approach using factor graph optimization (FGO) is explored, achieving better performance in signal-degraded scenarios.

The work also extends to space applications with the use of ML models for precise orbit determination (POD) of low-Earth-orbit (LEO) satellites, presenting polynomial regression's advantage in modeling nonlinear orbital dynamics. Furthermore, Edge AI applications are explored through the deployment of TinyML models for on-device localization, emphasizing feasibility in constrained environments while addressing energy efficiency, privacy, and real-time adaptability. Ethical implications of AI/ML in localization are examined in the context of the EU AI Act, emphasizing fairness, transparency, and responsible AI governance. Together, these contributions demonstrate the practical, scalable, and ethical use of ML and TinyML for accurate and reliable localization across diverse use cases and environments.

Keywords: global positioning satellite systems (GNSS), low-cost sensors, seamless navigation, machine learning (ML), factor graph optimization (FGO), precise orbit determination (POD), TinyML, ethical artificial intelligence (AI)

ACKNOWLEDGEMENTS

This doctoral dissertation was developed during my PhD studies at the University of Vaasa, Finland, between 2019 and 2025. The work presented in this article-based dissertation was financially supported by several research projects hosted by the Digital Economy Research Platform of the University of Vaasa, including TULEVA, TosiPaikka, @PACE, RESILIENT and AI2Business. I also gratefully acknowledges the private grant support from the University of Vaasa Graduate School (2019).

I would like to express my deep gratitude to Professor Petri Välisuo for his unwavering support and guidance throughout my academic journey, beginning with my MSc studies in 2016. His scientific mindset and professionalism helped me navigate numerous research challenges. I have learned a lot from him and had the pleasure of working on several projects under his mentorship.

To my manager and co-supervisor, Professor Heidi Kuusniemi thank you for your steadfast support and belief in my potential. I truly appreciate your professionalism, mentorship, and kindness. I am proud of the time I spent within your research team and will always cherish those years.

To my supervisor and wise mentor, Professor Mohammed Elmusrati, words cannot fully express my gratitude. Since starting my MSc in 2016, you have been a source of strength and guidance. You have shaped not only my academic path but also my personal growth. Thank you for your wisdom, patience, and the countless times you answered my calls, offered advice, and smoothed the road ahead.

Special thanks are extended to the esteemed reviewers, Naser Tarhuni, Associate Professor, and Joan Solà Ortega, as well as to the respected opponent, Professor Johan Lilius, for their valuable contributions and participation. I also sincerely thank the Dean, Professor Raine Hermans, for his continued support throughout this work.

To my colleagues, friends, and acquaintances in Finland, Nigeria, and beyond, thank you for your companionship and heartfelt support. I am especially grateful to the SDA church in Vaasa, whose warmth and proximity make me feel truly at home. I am especially grateful to Pastor Kenth Stolpe, who welcomed me warmly with open arms on my very first day in Vaasa after nothing more than a phone call

I made to introduce myself while still in Nigeria.

To my beloved mother, the pillar of our family—thank you for being my greatest supporter, my source of strength, and inspiration. To my six wonderful siblings, thank you for keeping me grounded, happy, and strong. To my in-laws, I deeply appreciate your kindness, respect, and encouragement. To all of you: thank you—I love you dearly.

Finally, to my beloved wife, your unwavering support and encouragement during the final years of this PhD journey made all the difference. You have been my anchor through both challenges and triumphs. Your patience, positivity, and grace continue to inspire me. I could not have done this without you. Thank you, my precious love.

I hope that readers find this book insightful and useful and that it meets or exceeds your expectations. Thank you for reading!

Vaasa, 6th of December 2025
Akpojoto Akporido Siemuri

CONTENTS

List of Figures **XIII**

List of Tables **XV**

1 INTRODUCTION 1

1.1 Localization and positioning 2

1.2 Machine learning applications in localization 3

1.3 Embedded systems and edge devices for localization 3

1.4 Aim and objectives 4

1.5 Main research questions 5

1.6 Summary of publications 7

1.7 Scientific contribution 10

1.8 Outline of the dissertation 11

2 Localization and positioning systems 13

2.1 Importance of localization and positioning 13

2.2 Location estimation algorithms 15

2.3 Local and global positioning systems 22

2.4 Low Earth Orbit navigation satellites 34

2.5 Mixed localization systems 36

2.6 Positioning system requirements 40

2.7 Machine learning 42

2.8 Factor graph optimization 44

3 Precise positioning with low-cost GNSS devices using ML algorithms 46
3.1 GNSS position estimation and performance 46
3.2 Challenges of GNSS in indoor & outdoor environment . . . 48
3.3 Machine learning techniques overview 49
3.4 Machine learning utilization in GNSS 58
3.5 GNSS datasets used by ML models 64
3.6 Precise positioning with low-cost GNSS devices 66
3.7 Machine learning and sensor fusion 72
3.8 Factor graph optimization application to GNSS 87
3.9 Keyframe Policy and Graph Management in the Implemented FGO 90
4 LEO Satellites: Orbit determination and positioning 95
4.1 Methods and datasets 95
4.2 ML models implemented 99
4.3 Results 100
5 Embedded ML applications for localization 104
5.1 Training AI/ML models on the edge 104
5.2 Cloud-Based training and edge deployment for AI/ML models 106
5.3 Federated Learning Approaches 107
5.4 Applications of TinyML in localization 108
5.5 Advantages of edge ML/TinyML 109
5.6 Challenges of edge ML/TinyML 110
6 Ethical concerns of AI/ML 112
6.1 European Union's Artificial Intelligence Act 113
6.2 Ethical concerns of AI/ML in localization applications . . . 114
7 Conclusion 116
7.1 Main findings 116

7.2 Scientific contributions 118
7.3 Novel Contributions 120
7.4 Limitations 121
7.5 Future research 121
References 123
Publications 146

List of Figures

1 Key scientific elements of publications [PI–PVIII] 7

2 Integration of localization applications and location-aware RRM in wireless systems 16

3 Main steps of Kalman filter algorithm 20

4 Typical GNSS constellation segments, adapted from M. Y. Yang, Rosenhahn, and Murino (2019) 28

5 GNSS frequencies in the L band, adapted from *Europa* (2021) . . . 30

6 GNSS errors, adapted from Biswas (2017) 33

7 Satellite footprint comparison in LEO and MEO 35

8 Multiple technologies as input for seamless positioning 37

9 Loosely coupled GNSS/IMU integration 38

10 Tightly coupled GNSS/IMU integration 40

11 Factor graph for positioning using factors for *a priori* location, ψ^p, IMU sensor ψ^{IMU}, and GNSS pseudorange observations ψ_i^ρ. The states in epoch, i, x_i, are estimated by the FG. The factors are described as influence functions, ψ, which return the probability of the current observation in state x_i and how the x_i should be updated to maximize the probability 44

12 Schematic diagram of multipath, line-of-sight (LOS), and NLOS signals 50

13 Systematic review process 62

14 Stages in the systematic review process 64

15 Flowchart of ML implementation pipeline. GNSS data is processed by BR, LR, and NN models. Their outputs are integrated using a weighted average (SWA), producing the final positioning output . . 71

16 Fusion-based positioning framework 72

17 Driving trajectories for collected data provided for GSDC 2022 . . . 74

18 Stages of applying the proposed LC-GNSS/IMU MAP-PPK algorithm 77

19 Comparison of positioning error between the WLS baseline and proposed method for each driving paths 79
20 Framework for indoor/outdoor detection. The three processes in this framework include Data Input, Training, and Testing. In the Data Input process, GNSS observation data is recorded, and features are extracted. The data are classified in the Training phase. The classifier is applied to detect the user's current environment in the Testing phase 81
21 Floor plan of Technobothnia laboratory with the planned robot trajectory 82
22 Comparison of C/No and number of available satellites for two test trips along the same navigation path 83
23 Classification accuracy using different ML classifier algorithms . . . 84
24 Conceptual pipeline for Indoor–Outdoor detection. GNSS data is classified by ML models (kNN, NB, NN) into indoor or outdoor categories. Based on the classification, the system selects GNSS/IMU positioning for outdoor environments or UWB/IMU positioning for indoor environments 87
25 Flowchart of the proposed FGO-based GNSS/IMU integration . . . 89
26 Comparison of positioning error between the selected phones (Mi8) using WLS baseline and proposed FGO-based method for a specific driving path 92
27 Comparison of positioning error between the selected phones (Pixel4) using WLS baseline and proposed FGO-based method for a specific driving path 93
28 Interpolating points between the observed data points for LEO SWARM-A satellite data using Lagrange interpolation method . . . 97
29 Position RMSE of LEO SWARM-A satellite for implemented ML models across various prediction time intervals 101
30 Velocity RMSE of LEO SWARM-A satellite for implemented ML models across various prediction time intervals 101
31 Position RMSE of MEO GPS satellite PRN-14 for implemented ML models across various prediction time intervals 102
32 Velocity RMSE of MEO GPS satellite PRN-14 for implemented ML models across various prediction time intervals 102
33 Machine learning model on edge device, adapted from Subramani and Araujo (2022) 105
34 Deploying machine learning at the edge, adapted from Subramani and Araujo (2022) 107
35 Human-centric AI, adapted from Lepri, Oliver, and Pentland (2021) 112

List of Tables

2 EKF for nonlinear systems and RTS algorithms. The EKF operation is explained in subsection 2.2 and the RTS algorithm in subsection 2.2 21
3 GNSS frequencies and signals 31
4 GNSS error sources, their properties, and correction methods 32
5 Major GNSS positioning modes 34
6 Comparison of LEO and MEO systems for navigation 35
7 Quality Assessment Questions 63
8 Publication Type Distribution 63
9 Results from evaluation of test data prediction on kaggle.com 72
10 Confusion matrix for NB classifier 84
11 Position estimation accuracy 94

ABBREVIATIONS

AI	Artificial Intelligence
AGV	Automated Guided Vehicle
CNN	Convolutional Neural Network
DL	Deep Learning
DoS	Denial-of-Service
EKF	Extended Kalman Filter
EGNOS	European Geostationary Navigation Overlay Service
EU	European Union
FGO	Factor Graph Optimization
GSDC	Google Smartphone Decimeter Challenge
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSA	European Global Navigation Satellite Systems Agency
GDPR	General Data Protection Regulation
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
ION	Institute of Navigation
IoT	Internet of Things
IPS	Indoor Positioning System
JCAS	Joint Communication and Sensing
KF	Kalman Filter
kNN	K-Nearest Neighbor
LC	Loosely Coupled
LEO	Low Earth Orbit
LLMs	Large Language Models
LPS	Local Positioning System
LR	Logistic Regression
LSTM	Long Short-Term Memory
MAP	ML-based Adaptive Positioning
MEMS	Micro-Electromechanical System
MEO	Medium Earth Orbit
ML	Machine Learning
NB	Naive Bayesian
NLOS	Non-Line-of-Sight
NavIC	Navigation with Indian Constellation
NN	Neural Network
PNT	Positioning, Navigation, and Timing
POD	Precise Orbit Determination
PPP	Precise Point Positioning
PPK	Post-Processed Kinematic
QZSS	Quasi-Zenith Satellite System

RAIM	Receiver Autonomous Integrity Monitoring
RF	Random Forest
RMSE	Root-Mean-Square Error
RTK	Real-Time Kinematic
RTS	Rauch–Tung–Striebel
SBAS	Satellite-Based Augmentation System
SNR	Signal-to-Noise Ratio
SVC	Support Vector Classifier
SVM	Support Vector Machine
TinyML	Tiny Machine Learning
TDOA	Time Difference of Arrival
TOA	Time of Arrival
TTFF	Time to First Fix
UAVs	Unmanned Aerial Vehicles
UWB	Ultra-Wideband
V2X	Vehicle-to-Everything
WAAS	Wide Area Augmentation System

LIST OF PUBLICATIONS

This doctoral dissertation is the culmination of research work done between 2021 and 2025 and has been compiled based on the following publications.

I. **Siemuri, Akpojoto**, H. Kuusniemi, M. S. Elmusrati, P. Välisuo and A. Shamsuzzoha, ”Machine Learning Utilization in GNSS—Use Cases, Challenges, and Future Applications,” 2021 International Conference on Localization and GNSS (ICL-GNSS), 2021, pp. 1-6, doi: 10.1109/ICL-GNSS51451.2021.9452295.[1]

II. **Siemuri, Akpojoto**; Selvan, Kannan; Kuusniemi, Heidi; Välisuo, Petri; Elmusrati, Mohammed S., ”Improving Precision GNSS Positioning and Navigation Accuracy on Smartphones using Machine Learning,” Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), St. Louis, Missouri, September 2021, pp. 3081-3093.
https://doi.org/10.33012/2021.18004.[2]

III. **Siemuri, Akpojoto**; Elsanhoury, Mahmoud; Välisuo, Petri; Kuusniemi, Heidi; Elmusrati, Mohammed S., ”Application of Machine Learning to GNSS/IMU Integration for High Precision Positioning on Smartphones,” Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022), Denver, Colorado, September 2022, pp. 2256-2264.
https://doi.org/10.33012/2022.18375. [2]

IV. **Siemuri, Akpojoto**, K. Selvan, H. Kuusniemi, P. Valisuo and M. S. Elmusrati, ”A Systematic Review of Machine Learning Techniques for GNSS Use Cases,” in IEEE Transactions on Aerospace and Electronic Systems, vol. 58, no. 6, pp. 5043-5077, Dec. 2022, doi: 10.1109/TAES.2022.3219366. [1]

V. **Siemuri, Akpojoto**; Elsanhoury, Mahmoud; Selvan, Kannan; Välisuo, Petri; Kuusniemi, Heidi; Elmusrati, Mohammed S., ”Seamless Navigation for Indoor-Outdoor Positioning Using GNSS-Aided UWB/WiFi/IMU System,” Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023), Denver, Colorado, September 2023, pp. 2616-2623.
https://doi.org/10.33012/2023.19323. [2]

VI. **Siemuri, Akpojoto**, Ahmadi, Elham, Elsanhoury, Mahmoud, Selvan, Kannan, Välisuo, Petri, Kuusniemi, Heidi, Elmusrati, Mohammed S., ”Optimal

Robust Positioning Using Factor Graph," Proceedings of the 37th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2024), Baltimore, Maryland, September 2024, pp. 2684-2690. https://doi.org/10.33012/2024.19909. [2]

VII. Selvan, Kannan, **Siemuri, Akpojoto**, Prol, Fabricio S., Välisuo, Petri, Kuusniemi, Heidi, "Machine Learning for LEO and MEO Satellite Orbit Prediction," Proceedings of the 37th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2024), Baltimore, Maryland, September 2024, pp. 3556-3571. https://doi.org/10.33012/2024.19765. [2]

VIII. **Siemuri, Akpojoto**; Elmusrati, Mohammed S., "Federated Learning and TinyML for Localization: Benefits, Applications, and Challenges". *Submitted for publication on 10.12.2025 to* Journal of Technology Research. https://jtr.cit.edu.ly/ojs/index.php/jtr/en/index. [3]

[1] Open Access articles, reprinted under a Creative Commons (CC BY) 4.0 license

[2] Reprinted with kind permissions from the copyright owners.

[3] Submitted to the publisher, currently under review.

AUTHOR'S CONTRIBUTION

Akpojoto Siemuri is the lead author in Publications I. - VI. and a coauthor in Publication VII. only. Publications on outdoor positioning systems are I. - IV., VII. and VIII. while publications on seamless navigation systems are VII. and VIII.. Detailed contributions from the author(s) are described below for each publication title.

Publication I: "Machine Learning Utilization in GNSS—Use Cases, Challenges and Future Applications"

Akpojoto Siemuri played a leading role in both publications, contributing to literature review, conceptualization, investigation, and analysis. Led the development of literature review questions, data processing, illustrations and sketching, validation, and troubleshooting. Authored the first draft, handled formatting, editing, and manuscript refinement. Presented the work at a conference and served as the corresponding author for both studies

Co-authors and Roles:

- **Ahm Shamsuzzoha**: Literature review, investigation, verification of results and analysis, writing, publishing guidance
- **Petri Välisuo**: Literature review, verification of results and analysis, illustrations, writing, editing, polishing, supervision, publishing guidance
- **Heidi Kuusniemi**: Funding, project management, supervision, mentorship, publishing guidance, polishing
- **Mohammed Elmusrati**: Supervision, mentorship, publishing guidance, conceptualization

Publication II: "Improving Precision GNSS Positioning and Navigation Accuracy on Smart- phones using Machine Learning"

Akpojoto Siemuri led the research effort, including literature review, conceptualization, investigation, algorithm development, coding, ML modeling, and data analysis. Participated in the 2021 Google Competition and received a Bronze Medal.

Contributed to illustrations, manuscript writing, editing, and final polishing, presented the work at a conference, and served as the corresponding author.

Co-authors and Roles:

- **Kannan Selvan**: participation in Google Competition 2021 (Bronze Medal), verification of results and analysis, literature review, writing, editing the article at different stages
- **Petri Välisuo**: Literature review, investigation, analysis, coding, illustrations, editing, polishing, supervision, publishing guidance
- **Heidi Kuusniemi**: Funding, project management, supervision, mentorship, publishing guidance, polishing
- **Mohammed Elmusrati**: Supervision, mentorship, publishing guidance, conceptualization

Publication III: "Application of Machine Learning to GNSS/IMU Integration for High Precision Positioning on Smartphones"

Akpojoto Siemuri led the research initiative, including literature review, conceptualization, investigation, algorithm development, ML modeling, data processing, and coding in Python and/or MATLAB. Participated in the 2022 Google Competition and earned a Bronze Medal. Oversaw validation, troubleshooting, manuscript writing, formatting, multi-stage editing, and final polishing. Presented the work at a conference and served as the corresponding author.

Co-authors and Roles:

- **Mahmoud Elsanhoury**: verification of results and analysis, participation in Google competition 2022 (Bronze Medal), literature review, conceptualization, investigation, analysis, algorithms development, data processing, Python and/or MATLAB coding, validation, troubleshooting, writing - format and editing, manuscript smoothing, editing the article at different stages
- **Petri Välisuo**: Literature review, investigation, algorithms development, analysis, data processing, illustrations, editing, polishing, supervision, publishing guidance
- **Heidi Kuusniemi**: Funding, project management, supervision, mentorship, publishing guidance, polishing the article

- **Mohammed Elmusrati**: Supervision, mentorship, publishing guidance, conceptualization

Publication IV: "A Systematic Review of Machine Learning Techniques for GNSS Use Cases"

Akpojoto Siemuri played a leading role in both publications, contributing to literature review, conceptualization, investigation, and analysis. Led the development of literature review questions, data processing, illustrations and sketching, validation, and troubleshooting. Authored the first draft, handled formatting, editing, and manuscript refinement. Presented the work at a conference and served as the corresponding author for both studies

- **Kannan Selvan**: developing the literature review questions, Data Processing, Illustrations and Sketching, Validation, verification of results and analysis, Writing - Format and Editing, Manuscript Smoothing, Corresponding Author. and served as the corresponding author.
- **Petri Välisuo**: Project coordination, analysis, verification of results and analysis, visual illustrations, writing, editing, polishing, supervision, publishing guidance, editing the article at different stages.
- **Heidi Kuusniemi**: Funding, project management, supervision, mentorship, publishing guidance, polishing
- **Mohammed Elmusrati**: Supervision, mentorship, publishing guidance, conceptualization

Publication V: "Seamless Navigation for Indoor-Outdoor Positioning Using GNSS-Aided UWB/WiFi/IMU System"

Akpojoto Siemuri led the research and development activities, including literature review, conceptualization, investigation, algorithm design, ML modeling, software development, and hardware implementation. Contributed to data collection, analysis, troubleshooting, illustration, manuscript writing, editing, and polishing, and presented the work at a conference. Also served as the corresponding author and project coordinator.

Co-authors and Roles:

- **Mahmoud Elsanhoury**: verification of results and analysis, literature review, conceptualization, investigation, analysis, data processing, validation, troubleshooting, writing - format and editing, manuscript smoothing, editing the article at different stages

- **Kannan Selvan**: verification of results and analysis, literature review, conceptualization, investigation, analysis, data processing, validation, troubleshooting, writing - format and editing, manuscript smoothing, editing the article at different stages

- **Petri Välisuo**: Literature review, investigation, analysis, data processing, illustrations, editing, polishing, supervision, publishing guidance

- **Heidi Kuusniemi**: Funding, project management, supervision, mentorship, publishing guidance, polishing

- **Mohammed Elmusrati**: Supervision, mentorship, publishing guidance, conceptualization

Publication VI: "Optimal Robust Positioning Using Factor Graph"

Akpojoto Siemuri led the project through all phases, including literature review, conceptualization, investigation, algorithm and software development, troubleshooting, data collection and analysis, and preparation of illustrations. Participated in the Google competition 2023-2024 and received a Silver Medal. Contributed to writing, editing, and polishing the manuscript, presented findings at a conference, and served as both the corresponding author and project coordinator.

Co-authors and Roles:

- **Elham Ahmadi**: Literature review, investigation, participation in Google competition 2023-2024 and received a Silver Medal, algorithm/software development, troubleshooting, analysis, writing, verification of results and analysis, data processing, validation, troubleshooting, writing - format and editing, manuscript smoothing, editing the article at different stages

- **Mahmoud Elsanhoury**: Literature review, investigation, data processing, analysis, writing, participation in Google competition 2023-2024 and received a Silver Medal, editing the article at different stages

- **Petri Välisuo**: Literature review, investigation, analysis, data processing, illustrations, editing, polishing, supervision, publishing guidance

- **Heidi Kuusniemi**: Funding, project management, supervision, mentorship, publishing guidance, polishing
- **Mohammed Elmusrati**: Supervision, mentorship, publishing guidance, conceptualization

Publication VII: "Machine Learning for LEO and MEO Satellite Orbit Prediction"

Akpojoto Siemuri contributed to investigation, algorithm and software development, Supported writing through formatting, multi-stage editing, and final manuscript polishing. Contributed to literature review, conceptualization, and investigation, including the development of review questions and data processing. Supported troubleshooting and created illustrations to enhance clarity. Played a key role in writing, editing, and polishing the manuscript.

Co-authors and Roles:

- **Kannan Selvan**: Contributed to investigation, algorithm and software development, troubleshooting, and data processing. Played a key role in developing literature review questions, creating illustrations and sketches, and validating and verifying results. Played a key role in writing, editing, and polishing the manuscript, presented the work at a conference, and served as the corresponding author
- **Fabricio Prol**: Contributed to investigation, algorithm and software development, Literature review, editing, polishing, supervision, publishing guidance, editing the article at different stages, and polishing the manuscript
- **Petri Välisuo**: Literature review, investigation, analysis, illustrations, editing, polishing, supervision, publishing guidance
- **Heidi Kuusniemi**: Funding, project management, supervision, mentorship, publishing guidance, polishing

Publication VIII: "Federated Learning and TinyML for Localization: Benefits, Applications, and Challenges"

Akpojoto Siemuri led the research activities which included investigation, algorithm and software development, troubleshooting, coding, and data processing. Addi-

tional contributions involved creating illustrations and sketches, formulating literature review questions, validating and verifying results, analyzing data, and refining the manuscript through formatting, editing, and final polishing.

Co-author and Role:

- **Mohammed Elmusrati**: Conceptualization, literature review, editing, polishing, supervision, mentorship, publishing guidance

OTHER PUBLICATIONS (NOT INCLUDED IN THE DISSERTATION)

These are a list of other peer-reviewed publications, which are not included in this doctoral dissertation.

I. Elsanhoury, Mahmoud, **Siemuri, Akpojoto**, Nieminen, Jyri, Välisuo, Petri, Koljonen, Janne, Kuusniemi, Heidi, Elmusrati, Mohammed S., ”Emerging Wireless Technologies for Reliable Indoor Navigation in Industrial Environments,” Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023), Denver, Colorado, September 2023, pp. 1706-1714. https://doi.org/10.33012/2023.19225

II. Elsanhoury, Mahmoud and Nieminen, Jyri and Välisuo, Petri and **Siemuri, Akpojoto** and Koljonen, Janne and Elmusrati, Mohammed and Kuusniemi, Heidi., ”Precise Indoor Positioning System for Mobile Robots via Smoothed UWB/IMU Sensor Fusion,” 2023 13th International Conference on Indoor Positioning and Indoor Navigation (IPIN), Nuremberg, Germany, 2023, pp. 1-6, doi: 10.1109/IPIN57070.2023.10332542.

III. Elsanhoury, Mahmoud, Jyri Nieminen, Petri Välisuo, **Siemuri, Akpojoto**, Janne Koljonen, Mohammed S. Elmusrati, and Heidi Kuusniemi. ”Indoor Asset Tracking in Dense Industrial Environments Using Low-cost Wireless Technologies.” In WIPHAL. 2023. https://osuva.uwasa.fi/handle/10024/16671.

IV. Selvan, K., **Siemuri, Akpojoto**, Prol, F.S., Välisuo, P., Bhuiyan, M.Z.H., Kuusniemi H. Precise orbit determination of LEO satellites: a systematic review. GPS Solut 27, 178 (2023). https://doi.org/10.1007/s10291-023-01520-7.

V. **Siemuri, Akpojoto**, Glocker, T., Mekkanen, M., Kauhaniemi, K., Mantere, T., Elmusrati, M.: Design and implementation of a wireless CAN module for marine engines using ZigBee protocol. IET Commun. 17, 1541–1552 (2023). https://doi.org/10.1049/cmu2.12640

1 INTRODUCTION

The most common positioning method used for outdoors is the Global Navigation Satellite System (GNSS), whereas for indoors, there is no clear leader, and various positioning technologies are often employed. Stand-alone GNSS and many local positioning systems provide accuracy within a few meters, which meets the needs of many users. However, many applications would benefit from decimeter-level or better positioning accuracy. Precise positioning methods, such as GNSS's real-time kinematic (RTK) technology, have been used for decades in applications like surveying and precision agriculture. Until recently, these methods were relatively expensive. However, recent developments in low-cost GNSS receivers and correction services have opened new opportunities for precise positioning, even in mass-market applications (Jackson, Davis, and Gebre-Egziabher (2018); Mahato et al. (2019)).

There are several correction methods used to improve the accuracy of position estimation from GNSS devices. The most widely used GNSS correction methods are RTK (Real-Time Kinematic) and PPP (Precise Point Positioning), both capable of providing centimeter-level accuracy. Based on the convergence time of the positioning solutions, PPP is less suited for mass-market applications as it has a longer convergence time. RTK has been extensively used for high-accuracy vehicle navigation, but its requirement for bidirectional communication in network RTK limits its applicability in mass-market scenarios. According to a European Global Navigation Satellite Systems Agency (GSA) report, the hybridization of RTK and PPP, known as PPP-RTK, shows promise for mass-market precision positioning applications (European GNSS Agency (2019)).

There is also growing interest in positioning technologies that are independent of GNSS. These technologies, often designed to operate within a limited geographical area, are known as local positioning systems (LPS). Indoor positioning systems (IPS) are commonly referred to as LPS. However, in some cases where GNSS is deployed to cover only a limited geographical area, it could also be referred to as LPS. In recent years, significant advancements have been made in LPSs based on various technologies such as Ultra Wideband (UWB) (Elsanhoury et al. (2022); Zafari, Gkelias, and Leung (2019)), Bluetooth direction finding (Sambu and Won (2022)), and 5G (Dwivedi et al. (2021)). Additionally, technologies commonly used to complement other positioning systems, such as radar, inertial navigation, and image-based localization, have also seen notable development (Mohamed et al. (2019)).

The growing complexity and dependency on GNSS technologies have increased the need for high-performance GNSS solutions in terms of accuracy, availability, continuity, and integrity at lower costs. Precise position information is crucial for many applications, including intelligent transport services, automated guided vehi-

cles (AGVs), and the Internet of Things (IoT). These applications may operate indoors, outdoors, or both. Most modern positioning systems rely on telecommunication infrastructures for range measurements or transmitting correction data. Ideally, the same infrastructure can simultaneously handle positioning and communications.

With the increase in the use of machine learning (ML) for data-driven applications, there have also been a growing interest in the utilization of ML in several GNSS use cases (Siemuri, Kuusniemi, Elmusrati, Välisuo, and Shamsuzzoha (2021)).

1.1 Localization and positioning

Recent advancements in low-cost dual-frequency GNSS receivers and affordable correction services have created new opportunities for precision mass-market location-based applications. These low-cost precise positioning technologies include low-cost GNSS receivers and smartphones, UWB. Bluetooth direction finding, 5G, radar, inertial navigation, image-based localization, and GNSS with different correction services. Additionally, with the growing interest in positioning technologies independent of GNSS, these technologies can be used both indoors and outdoors, either alone or in conjunction with other technologies in hybrid positioning systems. To provide accurate location information when transitioning between indoor and outdoor environments, a seamless outdoor-indoor positioning system is required. In seamless positioning, the integration of various positioning technologies is performed to provide uninterrupted location information as a body/user moves from one environment to another. Several integration methods have been proposed and implemented by various studies. In (W. Jiang, Cao, Cai, Li, and Wang (2021)), the authors proposed a tightly-coupled integration navigation method using UWB, GNSS and inertial navigation technologies for seamless positioning. The use of 5G positioning signals together with GNSS to provide a seamless navigation system has also been proposed by authors in (Keating, Säily, Hulkkonen, and Karjalainen (2019)).

This research work focuses mostly on the use of three (3) of these technologies: low-cost GNSS receivers, inertial measurement unit (IMU), and UWB. It covers both indoor and outdoor positioning technologies, as well as hybrid positioning methods that combine various approaches. Additionally, the essential observables and location estimation algorithms are discussed.

1.2 Machine learning applications in localization

Machine learning techniques enable the identification of complex dependencies in data that exploratory analysis might not reveal. The goal of using ML is not to derive an explicit formula for the data distribution but to train an algorithm to understand the relationship between input features and output. ML also facilitates the study of inter-relationships between dataset features. For instance, if the features are continuous variables, covariance can be used to measure their dependence on one another. This approach allows for more flexibility than many traditional statistical methodologies.

The use of ML in the context of GNSS has gained significant interest across various industries. For example, Google's DeepMind artificial intelligence (AI) has learned to navigate cities without a map, recognizing objects and their types in relation to the physical environment at various scales and distances. Google has also developed a new method to update its maps by combining deep learning with Street View data, using location information from Street View cars' GNSS combined with address information and business names extracted from imagery. This technique can effectively map an entire city without prior knowledge of its layout or nomenclature.

In February 2020, Apple applied to the Federal Communications Commission for a license to install GPS testing equipment at its headquarters, a move likely related to its August 2019 patent application for "Machine Learning Assisted Satellite Based Positioning."

1.3 Embedded systems and edge devices for localization

In the context of IoT localization, ML has been applied in the analysis of data from several sources (Shahbazian, Macrina, Scalzo, and Guerriero (2023)). IoT applications are becoming the main source for training data-gready machine learning models. In the past, IoT applications were mostly seen as a means of retrieving huge amount of data from the physical world and sending them to the Cloud. Spatial services use self-organizing services to retrieve, spread, or aggregate data collected by these IoT applications. In the modern world, spatial information includes the digital connection between location, people and activities.

The potential benefit of ML/DL (deep learning) has made it useful in many fields, and IoT localization and spatial services are not left out. In traditional ML, large servers are often used for processing heaps of data collected from the several in-

ternet sources to provide some benefit, like predicting what movie to watch next or to label a cat video automatically or predict the weather, etc. In the context of IoT localization, we can produce location based predictions faster and without the need for transmitting large amounts of raw data across a network by running ML algorithms on edge devices such as laptops, smartphones, and embedded systems found in smartwatches, washing machines, cars, manufacturing robots, etc.

It sometimes feels that our network and cloud infrastructure can scale and withstand almost any load, however, in reality, and for many reasons such as cost and latency, etc, we cannot send all the raw data to the cloud for processing. Edge machine learning (Edge ML) is important for many scenarios requiring the collection of raw data from sources that are far from the cloud. These scenarios may also have specific requirements or restrictions such as low-latency, real-time predictions, poor or non-existing connectivity to the cloud, legal restrictions that hinder sending data to external services, and large datasets requiring pre-processing locally before sending responses to the cloud. Therefore, the use of edge computing is essential. Edge ML also known as edge artificial intelligence or edge AI is defined as the process of running ML algorithms on computing devices at the periphery of a network to make decisions and predictions as close as possible to the originating source of data (Filho et al. (2022)). This can be done in a distributed way. Distributed edge intelligence is a disruptive research area that allows the execution of ML/DL algorithms close to the source where the data is generated (Murshed et al. (2021)).

1.4 Aim and objectives

The purpose of this study was to investigate and improve the accuracy of position estimation and localization using machine learning (ML) techniques. Additionally, the study explored the potential of distributed ML algorithms—such as Tiny Machine Learning (TinyML) for deployment on IoT and edge devices. The specific objectives of this research are as follows:

1. Conduct a comprehensive review of ML utilization in GNSS, including:
    - Common GNSS use cases where ML algorithms have been applied,
    - Datasets utilized in ML-based GNSS positioning,
    - Comparative performance of ML versus traditional GNSS algorithms,
    - Examination of ML evaluation and validation methods in GNSS applications,
    - Risks and limitations associated with the application of ML in GNSS,
    - Strengths and weaknesses of ML models for GNSS use cases.

2. Apply ML algorithms to low-cost GNSS devices to improve the accuracy of their position estimates.

3. Investigate how factor graph optimization (FGO) can be integrated with ML algorithms to improve the positioning accuracy of low-cost GNSS devices and smartphones.

4. Study the benefits of deploying TinyML models on IoT edge devices for localization and real-time navigation.

1.5 Main research questions

This dissertation is compiled based on publications I. - VIII. and is guided by the following primary research questions (RQ1–RQ4):

1. **RQ1 Application and Evaluation of ML in GNSS Use Cases:** How have machine learning (ML) techniques been applied in Global Navigation Satellite System (GNSS) use cases, and what are the associated strengths, limitations, and risks in terms of positioning accuracy, reliability, and data integrity?

   Traditional GNSS algorithms provide reliable positioning, navigation, and timing (PNT) under good signal conditions but often struggle in degraded environments. To address this, researchers have explored ML-based GNSS models since the 1990s. In Publications I. and IV., we present a systematic review of studies from 2000 to 2021, evaluating the application, performance, and limitations of ML techniques in GNSS use cases.

   A notable application of ML in the GNSS domain is presented in publication VII., where the use of ML for precise orbit determination (POD) of Low Earth Orbit (LEO) satellites is explored.

   ML techniques have demonstrated significant potential for improving orbit prediction of satellites in both LEO and Medium Earth Orbit (MEO). By learning complex, non-linear relationships in satellite motion, ML models can uncover patterns that are often difficult to capture with conventional approaches. Leveraging historical satellite data, these models can enhance the estimation of position and velocity, thereby reducing orbit prediction errors and improving the overall reliability of satellite-based systems.

2. **RQ2 Enhancement of Low-Cost GNSS Positioning:** To what extent can ML algorithms improve the positioning accuracy of low-cost GNSS-enabled devices, especially in complex and constrained environments such as urban canyons?

ML can significantly enhance the accuracy of smartphone positioning by analyzing and correcting for errors in GNSS and IMU data. ML algorithms can learn complex relationships and patterns in the data that traditional methods might miss, leading to more precise location estimates, particularly in challenging environments like urban areas with signal interference. These are addressed in publications II., III.. While in publication V., seamless navigation is studied. The demand for seamless indoor-outdoor navigation is increasing across applications like factories, military, and emergency response. Multi-sensor fusion, combined with GNSS, and recent advances in AI and ML, offer effective solutions for enhancing localization accuracy and reliability.

By leveraging the capabilities of ML, researchers and developers are pushing the boundaries of smartphone-based location technology, enabling new applications in areas like navigation, augmented reality, and location-based services.

3. **RQ3 Integration of Factor Graph Optimization with ML:** How can factor graph optimization (FGO) be effectively integrated with ML techniques to enhance position estimation accuracy in GNSS/IMU fusion systems for low-cost receivers and smartphones?

   Smartphone GNSS positioning is often limited by low-cost antennas and chips, making it vulnerable to NLOS and multipath effects, especially in urban environments. Traditional methods like EKF struggle under these conditions. Factor Graph Optimization (FGO) has shown promise by leveraging multi-epoch GNSS measurements for robust positioning. The study in publication VI. implements an FGO-based GNSS/IMU integration using the Google Smartphone Decimeter Challenge 2023 dataset. Unlike filtering methods, FGO utilizes all past measurements for improved state estimation. The loosely coupled approach directly combines GNSS and IMU data, offering a practical and resilient positioning solution suitable for both hardware and software implementation.

4. **RQ4 Deployment of TinyML Models for Edge Localization:** What are the benefits and challenges of deploying TinyML models on IoT edge devices for real-time localization, considering constraints such as computational resources, energy efficiency, and data privacy?

   In recent literature, TinyML has rapidly matured as a practical paradigm for executing machine learning models on resource-constrained embedded devices, making it particularly well suited for implementing ML-based GNSS functions on smartphones and low-cost GNSS receivers. Publication VIII. discusses TinyML applications for localization and examines the benefits and challenges of deploying TinyML models on IoT edge devices, including trade-offs among model size, latency, energy consumption, and positioning accuracy. The study emphasizes engineering techniques—quantization, pruning, architecture search, and efficient inference runtimes—that enable

feasible on-device GNSS tasks such as multipath detection, indoor/outdoor classification, and lightweight position refinement, and it highlights the need for careful evaluation under realistic hardware and power constraints.

5. **RQ5 Ethical Considerations of AI/ML/LLM in Localization:** How can privacy-preserving technologies be effectively integrated into AI/ML/LLM-based localization systems to ensure compliance with the ethical standards and regulatory requirements outlined in the 2024 EU AI Act?

   The integration of AI, ML, and large language models (LLMs) into localization systems introduces complex ethical challenges, particularly around privacy, security, and regulatory compliance. Chapter 6 of this dissertation addresses the benefits and challenges of deploying TinyML models on IoT edge devices.

1.6 Summary of publications

This section provides a brief summary of all publications [I–VIII] included in this compilation dissertation. An overview illustrating the scientific contributions of each publication is presented in Figure 1.

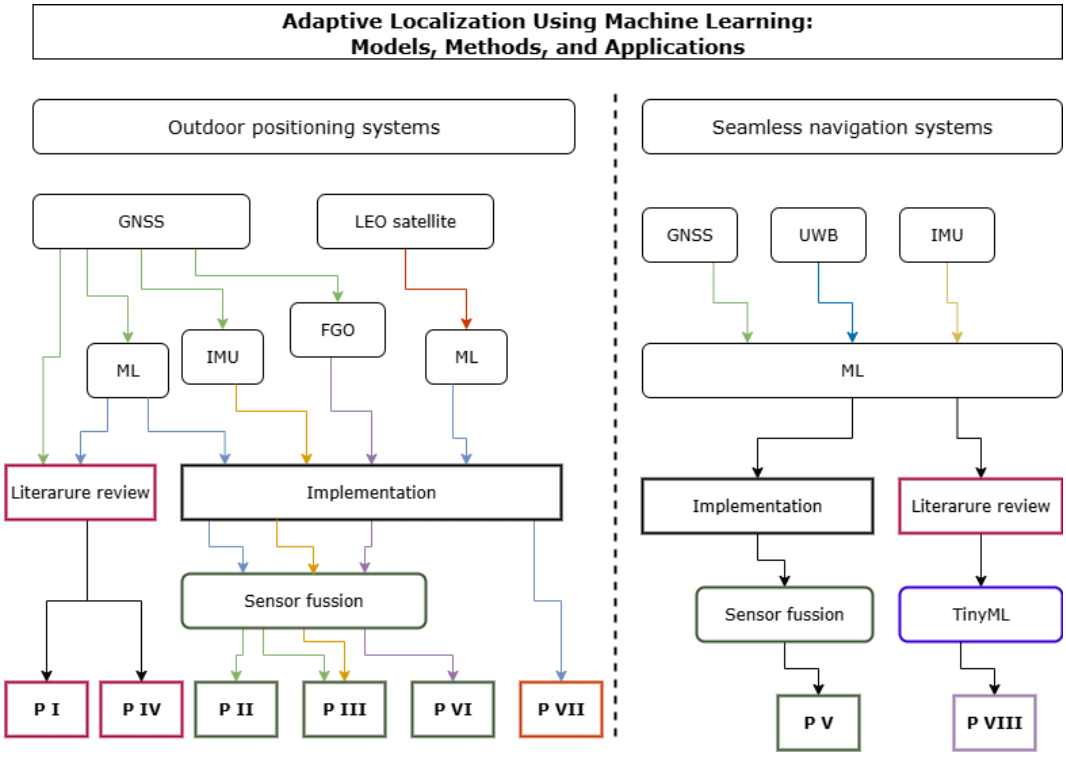


Figure 1. Key scientific elements of publications [PI–PVIII].

Publication I.: *Machine Learning Utilization in GNSS—Use Cases, Challenges and Future Applications*
Publication IV.: *A Systematic Review of Machine Learning Techniques for GNSS Use Cases*

These two complementary publications investigate the application of machine learning (ML) techniques in Global Navigation Satellite Systems (GNSS) from 2000 to 2021. Publication I focuses on practical use cases where ML has enhanced GNSS performance, particularly in positioning accuracy, reliability, and integrity. It provides a technical and conceptual overview of how ML has been integrated into GNSS workflows and highlights emerging opportunities for intelligent navigation systems.

Publication IV builds on this foundation by offering a systematic review of ML techniques used in GNSS. It identifies key trends, research gaps, and evaluates the strengths and limitations of ML approaches in GNSS positioning. This work delivers a critical assessment of ML's performance, summarizing its benefits, risks, and future potential, thereby guiding researchers toward impactful directions in GNSS innovation.

Publication II.: *Improving Precision GNSS Positioning and Navigation Accuracy on Smartphones using Machine Learning*

This study applies machine learning (ML) algorithms to enhance the positioning accuracy of low-cost GNSS devices, particularly smartphones, in challenging environments such as urban canyons. By correcting GNSS measurement errors, the ML models developed in this work outperform traditional positioning methods and demonstrate the potential for cost-effective, high-precision navigation solutions.

Publication III.: *Application of Machine Learning to GNSS/IMU Integration for High Precision Positioning on Smartphones*

Building upon the findings of Publication II, this research integrates GNSS and Inertial Measurement Unit (IMU) data using machine learning (ML) techniques to further enhance positioning accuracy on smartphones. The study demonstrates that sensor fusion—combining GNSS and IMU inputs—significantly improves performance in complex urban environments where GNSS signals are often degraded or obstructed. ML algorithms are employed to process and correct sensor data, resulting in robust and precise positioning outcomes suitable for real-world navigation applications.

Publication V.: *Seamless Navigation for Indoor-Outdoor Positioning Using GNSS-Aided UWB/WiFi/IMU System*

This publication addresses the growing need for seamless localization across indoor

and outdoor environments. It demonstrates how machine learning (ML), combined with GNSS and multi-sensor fusion technologies such as Ultra-Wideband (UWB), WiFi, and Inertial Measurement Units (IMU), can enable reliable and continuous positioning. The proposed system is particularly suited for critical applications including emergency response, military operations, and industrial automation, where robust and uninterrupted navigation is essential.

Publication VI.: *Optimal Robust Positioning Using Factor Graph*

This study explores the application of Factor Graph Optimization (FGO) for integrating GNSS and IMU data to achieve high-precision positioning on smartphones. Leveraging the Google Smartphone Decimeter Challenge 2023 dataset, the research demonstrates that FGO, when combined with machine learning (ML) techniques, significantly outperforms traditional filtering methods. The results highlight the potential of FGO-based approaches to deliver resilient and accurate positioning even with low-cost consumer-grade hardware, making it suitable for scalable deployment in real-world navigation systems.

Publication VII.: *Machine Learning for LEO and MEO Satellite Orbit Prediction*

This work presents a specialized application of machine learning (ML) for precise orbit determination (POD) of low Earth orbit (LEO) and middle Earth orbit (MEO) satellites. By learning complex, non-linear motion patterns from historical satellite data, the ML models developed in this study significantly improve orbit prediction accuracy. The enhanced predictive capability contributes to greater reliability in space-based GNSS services, with implications for satellite navigation, timing, and Earth observation systems.

Publication VIII.: *Federated Learning and TinyML for Localization: Benefits, Applications, and Challenges*

This article investigates methods for tailoring and engineering machine learning models to run on resource constrained embedded hardware. Focusing on Tiny Machine Learning (TinyML) for ultra-low-power inference on microcontrollers and sensors, it also investigates how Federated Learning (FL) complements on-device deployment by enabling collaborative training without centralizing sensitive data. The review covers end-to-end workflows from cloud training to edge deployment, optimization techniques such as quantization and pruning, and training strategies suitable for constrained platforms, and it surveys concrete localization use cases. We evaluate the advantages—low-latency inference, improved privacy, and reduced energy consumption—alongside practical limitations, including tight computational and memory budgets, interoperability challenges across heterogeneous devices, and security and robustness concerns.

1.7 Scientific contribution

The scientific contributions are briefly outlined below as follows:

1. **Comprehensive Literature Review on ML in GNSS (Publications I & IV)**:
   A systematic review of ML applications in GNSS use cases, highlighting research trends, strengths, limitations, and risks. The review serves as both an academic reference and a practical guide for researchers and practitioners interested in GNSS and ML integration.

2. **Low-Cost Outdoor GNSS/IMU/ML Positioning (Publications II & III)**:
   Development of a precise, low-cost positioning system for outdoor environments, particularly urban areas, through the integration of GNSS, IMU, and ML techniques. The approach achieved improved accuracy and ranked Bronze in the Google Decimeter Challenge 2022.

3. **Seamless Indoor-Outdoor Positioning with GNSS/IMU/UWB/ML (Publication V)**:
   Integration of GNSS, UWB, IMU, and ML techniques for seamless, low-cost positioning across indoor–outdoor environments. This approach enables reliable real-time localization for smart logistics and location-aware services.

4. **FGO-Based GNSS/IMU Integration for Precise Positioning (Publication VI)**:
   An alternative GNSS/IMU integration approach using Factor Graph Optimization (FGO) for robust and high-accuracy positioning in urban environments. The method outperformed traditional filtering techniques and achieved a Silver ranking in the Google Decimeter Challenge 2023.

5. **ML-Based Precise Orbit Determination of LEO Satellites (Publication VII)**:
   Application of ML models for precise orbit determination (POD) of LEO satellites. The study demonstrated significant improvements in orbit prediction accuracy compared to conventional approaches, contributing to advancements in satellite navigation and space services.

6. **TinyML for Localization in Resource-Constrained Devices (Publications VIII)**:
   Demonstration of the feasibility of TinyML for real-time localization on IoT and low-power devices supporting indoor tracking, zone classification, and GNSS/IMU/UWB integration. The study also addresses resource constraints and privacy challenges associated with the application of TinyML on edge devices.

7. **Ethical Considerations of AI/ML/LLM in Localization (Chapter 6)**: Discussion of privacy, security, and ethical risks associated with applying AI, ML, and LLMs to localization. Emphasis is placed on the need for privacy-preserving technologies and responsible AI development, particularly in light of the 2024 EU AI Act.

1.8 Outline of the dissertation

The rest of the dissertation chapters are organized as follows:

Chapter 2 examines the key observables and location estimation algorithms utilized in radio navigation for position determination. The primary emphasis is on observables and positioning methods that enable direct location estimation. Notable examples of such systems include GNSS, UWB, and LEO satellite systems.

Chapter 3 focuses on the use of ML and other technologies to achieve precise positioning with low-cost GNSS devices. It presents a systematic review of ML applications in GNSS, including successful use cases, common algorithms, challenges, and research opportunities. The chapter covers the development of precise outdoor positioning using ML in GNSS and GNSS/IMU integration, followed by a seamless indoor-outdoor system combining GNSS, UWB, and IMU. It also highlights the use of factor graph optimization (FGO) as an alternative to the extended Kalman filter (EKF) for GNSS-IMU integration. FGO has been increasingly applied in GNSS research, offering robust solutions for sensor fusion and positioning challenges. The application of these methods were done in the Google Smartphone Decimeter Challenge (2021 - 2023) to enhance smartphone positioning capabilities.

Chapter 4 discusses the critical role of GNSS-based precise orbit determination (POD) for low Earth orbit (LEO) satellites in supporting space applications such as navigation, telecommunications, remote sensing, and Earth observation. These applications rely on accurate satellite orbit tracking enabled by onboard GNSS receivers. Recent advancements have led to the development and deployment of GNSS receivers specifically designed to meet the high-precision requirements of satellite POD, tailored to various mission needs. It assess the contribution of ML for LEO and MEO Satellite Orbit Prediction, which aims to enhance the accuracy of orbit predictions for LEO and MEO satellites.

Chapter 5 explores the benefits and challenges of deploying ML and deep learning (DL) models on resource-constrained edge devices in a distributed manner. It discusses strategies for adapting, training, caching, inference, and offloading ML models at the edge. Special attention is given to TinyML, an emerging approach that enables AI techniques to run directly on ultra-low-power embedded devices,

offering a promising solution for on-device tracking and localization. It also addresses the challenges such as computational limitations, energy constraints, model accuracy, interoperability, and security and privacy concerns.

Chapter 6 addresses the issue of ethical AI and ML focusing on ensuring that AI and ML technologies are designed and applied in ways that are fair, transparent, accountable, and aligned with human values. As AI systems play an increasing role in critical areas like healthcare, finance, criminal justice, and localization, addressing concerns such as algorithmic bias, data privacy, accountability, and explainability becomes essential.

Chapter 7 presents the conclusions, key results, and insights from all publications, along with a summary of the scientific contributions and novel findings of this dissertation. It also outlines potential directions for future research building upon the outcomes of this work.

2 LOCALIZATION AND POSITIONING SYSTEMS

This chapter discusses various observables and location estimation algorithms used in radio navigation for position estimation. Our primary focus is on the observables and positioning algorithms that estimate an object's location directly. Examples of systems that use direct positioning include GNSS and UWB.

Some positioning technologies, such as inertial navigation systems (INS), do not provide observables related to an absolute position. Instead, these systems use data from motion sensors to integrate the position and orientation starting from a known location, a process known as dead reckoning. Dead reckoning is a method of estimating the current position of an object based on its previously determined position, and accounting for known or estimated speeds over elapsed time, course, and distance traveled. This technique is used when no direct positional information is available, instead relying on motion sensors or other data sources to infer movement. Dead reckoning is often used in conjunction with other positioning methods to improve accuracy, especially in environments where GNSS signals are weak or unavailable (e.g., indoors, underground or in urban canyons).

We also discuss the trend in the use of machine learning (ML) and factor graph optimization (FGO) techniques for the improvement of location estimation. However, before proceeding, we can discuss the importance of localization and positioning in daily human life.

2.1 Importance of localization and positioning

Localization and positioning play a crucial role in daily human life, influencing various applications ranging from navigation to safety, healthcare, and resource management. These systems enable services such as GPS navigation for travelers, location-based marketing for businesses, and emergency response systems for public safety.

2.1.1 Applications of localization and positioning

Localization and positioning technologies have become integral to modern life, enabling a wide range of applications in various fields. From navigation and transportation to health care and smart city initiatives, these technologies provide precise location information to improve efficiency, safety, and user experiences. They play a pivotal role in industries such as retail, agriculture, and robotics, while also support-

ing critical services such as emergency response and military operations (Elliott D. and Christopher J. (2017); X. Guo and Ansari (2020); Zafari, Papapanagiotou, and Christidis (2016)). Machine learning (ML) is increasingly being integrated into localization systems to improve accuracy, adapt to dynamic environments, and optimize energy consumption, offering enhanced capabilities in various applications (Carrera Villacrés, Zhao, Braun, and Li (2019); Shahbazian et al. (2023); Siemuri, Kuusniemi, et al. (2021)). Using advancements in GNSS and other local positioning systems, localization continues to transform how people and devices interact with their environments.

- **Navigation and Transportation:** Enabling non-GNSS-based and GNSS-based navigation for vehicles, pedestrians, and autonomous systems.
- **Smart Cities:** Supporting urban planning, traffic management, and public safety through precise localization.
- **Healthcare:** Tracking medical equipment, patients, and staff within hospitals for efficient operations.
- **Retail and Marketing:** Deliver location-based promotions and track customer behavior in stores.
- **Emergency Services:** Locate individuals in need during disasters or accidents for rapid response.
- **Agriculture:** Facilitating precision farming through real-time tracking of machinery and resources.
- **Military and Defense:** Provide situational awareness and navigation in critical operations.
- **Robotics and Automation:** Supporting robotic navigation and operation in warehouses and industrial environments.
- **Sports and Entertainment:** Enhancing fan experiences with augmented reality and tracking athlete' performance.
- **Internet of Things (IoT):** Enabling device tracking in smart homes and industrial IoT environments.

2.1.2 Location-aware radio resource management in wireless communication

Recent advances in wireless communication, particularly in 5G and upcoming 6G systems, highlight the importance of *location-aware radio resource management*

(RRM). Using the precise positions of the transmitter (Tx) and receiver (Rx), radio resources such as spectrum, power, time slots, and beams can be allocated more efficiently. Unlike conventional methods that rely solely on channel-state information (CSI), location knowledge provides an additional geometric and spatial dimension for optimization (Wymeersch et al. (2014)).

Key applications include: (i) beamforming and beam management, where accurate positions reduce beam search overhead and improve spectral efficiency; (ii) interference management, enabling predictive coordination in dense urban deployments; (iii) proactive handover optimization, reducing latency and dropped connections; (iv) radio resource scheduling in vehicle-to-everything (V2X) networks, ensuring ultra-reliable low-latency communication (URLLC); and (v) joint communication and sensing (JCAS) in 6G, where localization is integrated into the communication process.

Despite these opportunities, there are challenges to achieve sufficient positioning accuracy in urban and indoor environments, ensuring scalability for large user populations, and addressing privacy concerns associated with continuous location tracking. However, location-sensitive RRM is widely considered a key enabler of future 6G networks, integrating communication, positioning, and sensing for more efficient, reliable, and adaptive wireless systems (Bourdoux et al. (2020)).

2.1.3 Conceptual integration of localization and radio resource management

To better illustrate the relationship between real-world localization applications and technical advances in RRM, Figure 2 presents a conceptual overview that bridges Sections 2.1.1 and 2.1.2. The diagram highlights how diverse application domains, from smart cities and healthcare to autonomous systems and industrial IoT—rely on a variety of localization technologies. These technologies, in turn, serve as foundational inputs for optimizing wireless communication resources in modern and future networks. By mapping this flow from application to localization to RRM, the figure underscores the central role of positioning in enabling intelligent, adaptive, and efficient wireless ecosystems.

2.2 Location estimation algorithms

Position estimation can be achieved through various techniques. When location is calculated based solely on measurements taken at a single time instance, it is termed static positioning. An example of this approach is the iterative least squares

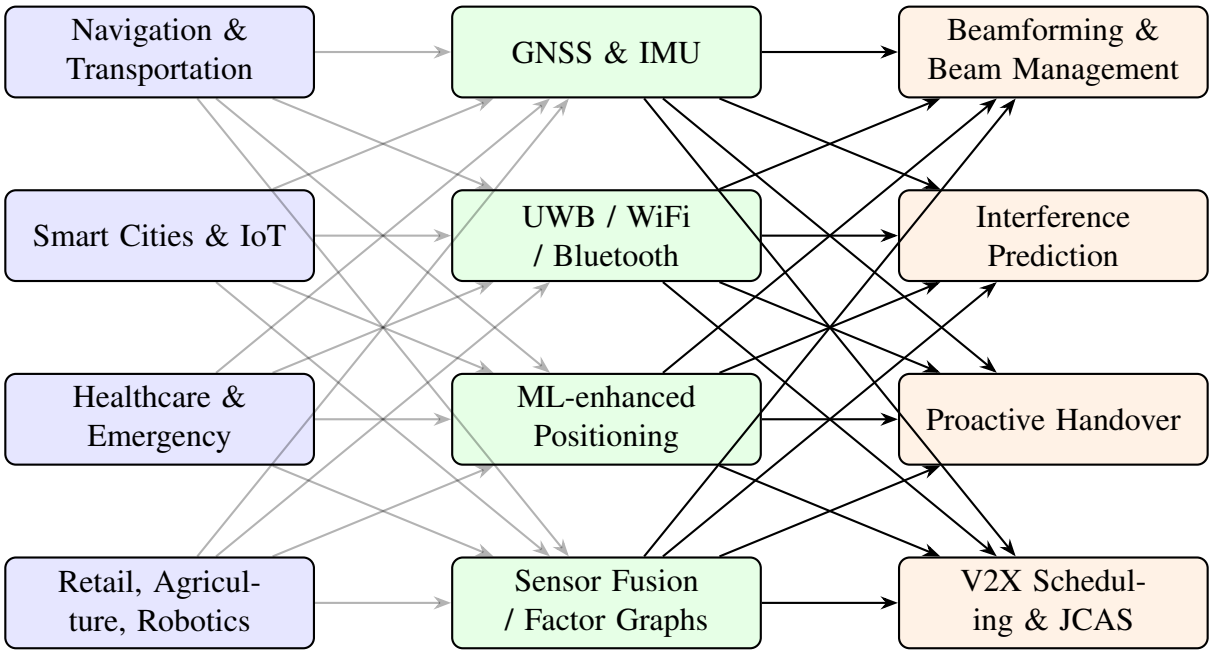


Figure 2. Integration of localization applications and location-aware RRM in wireless systems.

method, which solves nonlinear equations involving TOA (Time of Arrival) and TDOA (Time Difference of Arrival). However, static positioning does not account for the movement or changing state of the target. To address this limitation, filters such as Kalman and particle filters are often employed, as they enhance accuracy by incorporating information about the previous state to predict the current one. This section provides an overview of the methods utilized in this study.

Time of arrival (TOA) and time difference of arrival (TDOA): In Global Navigation Satellite Systems (GNSS), positioning can be derived using either the *Time of Arrival (TOA)* or the *Time Difference of Arrival (TDOA)* of signals.

Time of arrival (TOA): TOA measures the absolute travel time of a signal from a satellite (transmitter, Tx) to a receiver (Rx). The pseudorange is obtained as

$$\rho = c \cdot (t_{rx} - t_{tx}), \tag{2.1}$$

where c is the speed of light, t_{rx} is the reception time, and t_{tx} is the satellite transmission time. Precise synchronization between the satellite and receiver clocks is required. In practice, satellite clocks are synchronized using atomic references, while the receiver estimates its own clock bias. Standard GNSS positioning is based on TOA-derived pseudoranges from at least four satellites.

Time difference of arrival (TDOA): TDOA measures the difference in signal arrival times either between multiple receivers or between pairs of satellites. The time difference can be expressed as

$$\Delta t_{ij} = (t_{rx,i} - t_{tx}) - (t_{rx,j} - t_{tx}) = t_{rx,i} - t_{rx,j}, \tag{2.2}$$

where $t_{rx,i}$ and $t_{rx,j}$ are the reception times on the receivers i and j. Here, the common transmission time t_{tx} is canceled, making TDOA less sensitive to satellite clock errors. This technique is particularly useful in differential GNSS, cooperative positioning, or multilateration systems (e.g, UWB or cellular networks), where relative timing accuracy is more critical than absolute positioning.

Key Difference.

- TOA measures **absolute travel time**, requires clock bias correction, and supports standard GNSS pseudorange positioning.
- TDOA measures **relative arrival time differences**, cancels common clock errors, and is widely applied in relative or network-based positioning.

Least squares: Although the Gauss-Newton algorithm is a common method for solving overdetermined and nonlinear systems of equations—such as those encountered in iterative position estimation—it is computationally inefficient for real-time applications. Therefore, we introduce the Least Mean Squares (LMS) algorithm as a more efficient alternative to solve such systems.

In localization using an anchor and tag, the actual distance between an anchor point *i* and a tag is calculated as specified in Equation (2.3):

$$r_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} \tag{2.3}$$

where (x_i, y_i, z_i) is the position of the i^{th} anchor and (x_u, y_u, z_u) is the position of the tag. If (x_v, y_v, z_v) is the initial approximate position, let $x_u = x_v + \delta_x$, $y_u = y_v + \delta_y$ and $z_u = z_v + \delta_z$. By linearizing Equation (2.3) using Taylor-series expansion and omitting second-order and higher terms as in Equations (2.4)–(2.8), (C.-S. Chen (2017); Elsanhoury et al. (2022); Shen, Zetik, and Thoma (2008)) we have:

$$H\delta = b \tag{2.4}$$

where

$$\mathbf{b} = \begin{bmatrix} r_1 - r_{v1} \\ r_2 - r_{v2} \\ . \\ . \\ r_n - r_{vn} \end{bmatrix}, \mathbf{H} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} \\ a_{x2} & a_{y2} & a_{z2} \\ . & . & . \\ . & . & . \\ a_{xn} & a_{yn} & a_{zn} \end{bmatrix}, \tag{2.5}$$

$$\delta = [\delta x_u, \delta y_u, \delta z_u]^T , \tag{2.6}$$

$$a_{xi} = \frac{x_i - x_v}{r_{vi}}, a_{yi} = \frac{y_i - y_v}{r_{vi}}, a_{zi} = \frac{z_i - z_v}{r_{vi}} \tag{2.7}$$

and

$$r_{vi} = \sqrt{(x_i - x_v)^2 + (y_i - y_v)^2 + (z_i - z_v)^2} \tag{2.8}$$

The Least Squares (LS) solution for the position estimation problem is derived using Equation (2.9) (C.-S. Chen (2017)).

$$\delta = (\mathbf{H^T H})^{-1} \mathbf{H^T b} \tag{2.9}$$

The object's position (x_u, y_u, z_u) is determined iteratively. Starting with an initial guess (x_v, y_v, z_v) the process continues until the solution converges (C.-S. Chen (2017)).

The Weighted Least Squares (WLS) solution can be computed using Equation (2.10):

$$\mathbf{\Delta x} = (\mathbf{H^T W H})^{-1} \mathbf{H^T W \Delta p} \tag{2.10}$$

The elements of Equation (2.10) are defined as follows:

- $\mathbf{\Delta x}$: The estimated correction vector for unknown parameters (e.g, receiver position and clock bias). It represents the adjustment applied to the current state estimate to minimize the residual error between the observed and modeled measurements.
- $\mathbf{H}$: The design matrix (also called the geometry or the Jacobian matrix). Each row of $\mathbf{H}$ contains the partial derivatives of the measurement equations with respect to the unknown parameters. In GNSS positioning, it encodes the satellite–receiver geometry and relates small changes in the estimated parameters

to the corresponding changes in the measurements.

- $\mathbf{W}$: The weighting matrix, typically a diagonal matrix, containing the statistical weights assigned to each measurement. These weights are usually chosen as the inverse of the measurement variances:

$$\mathbf{W} = \text{diag}\left(\frac{1}{\sigma_1^2}, \frac{1}{\sigma_2^2}, \ldots, \frac{1}{\sigma_m^2}\right),$$

  where σ_i^2 is the variance of the i^{th} measurement. This ensures that more reliable (low-variance) measurements contribute more strongly to the solution.

- $\mathbf{\Delta p}$: The observation residual vector, representing the difference between the actual measured values and the predicted measurements based on the current estimate of the unknown parameters. It indicates how far the current solution is from fitting the observations.

- $(\mathbf{H^T W H})^{-1}$: The inverse of the normal matrix. This term combines the geometry of the system and the weights of the observations to determine the sensitivity and stability of the solution. Its inverse exists when the system is well-conditioned (i.e., the measurements provide sufficient and non-redundant information).

In summary, Equation (2.10) updates the estimate of the unknown parameters by minimizing a weighted sum of squared residuals, where measurements with higher accuracy (lower variance) have a greater influence on the solution.

The iterative least squares method described above for TOA can also be applied for TDOA observables starting from Equation 2.11.

$$\begin{aligned} r_{i,1} = cd_{i,1} = r_i - r_1 = &\sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} \\ &- \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} \end{aligned} \tag{2.11}$$

Here, c represents the speed of signal propagation, $r_{i,1}$ is the distance difference between the first base station and the i^{th} base station, r_1 denotes the distance from the first base station to the mobile terminal and $d_{i,1}$ corresponds to the measured TDOA between the first and the i^{th} base station. Equation (2.11) describes a set of nonlinear hyperbolic equations, the solution of which provides the three-dimensional coordinates of the mobile terminal.

In addition to iterative methods, closed-form (non-iterative) approaches are available for solving overdetermined nonlinear systems of equations. These methods have been discussed in works such as (Cheng and Zhou (2019); X. Li, Deng, Rauchenstein, and Carlson (2016); Shen et al. (2008)).

Kalman filters: The Kalman filter is an iterative and recursive estimation algorithm designed to predict optimal states in linear state-space systems, accounting for additive white Gaussian noise (Kalman (1960)). This algorithm integrates prior knowledge to derive posterior states and computes the Kalman gain and innovation, producing optimal state and covariance estimates (Darbellay (1999); Elsanhoury et al. (2022); Faragher (2012); Hartikainen, Solin, and Särkkä (2011); Kalman (1960); Schalk and Leuthardt (2011)).

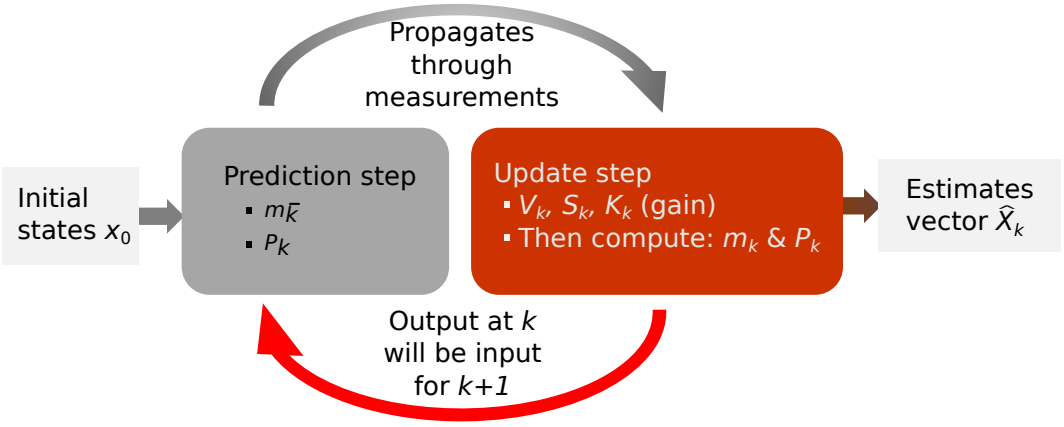


Figure 3. Main steps of Kalman filter algorithm.

A. Extended kalman filter (EKF): The Extended Kalman Filter (EKF) extends the capabilities of the standard Kalman filter to nonlinear systems (Y. Bar-Shalom (2001)). All variants of Kalman filters operate in two phases, as illustrated in Figure 3.

- **Prediction step:** The system's future state is predicted based on prior information.
- **Update step:** The predicted state is corrected using the calculated innovation and the Kalman gain (Bar-Shalom (1989); Hartikainen et al. (2011)).

The EKF algorithm, outlined in Table 2, fulfills the state-space estimation equations through these iterative steps:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1},\ k-1) + \mathbf{q}_{k-1},$$
$$\mathbf{y}_k = h(\mathbf{x}_k,\ k) + \mathbf{r}_k,$$

In this algorithm:

$\mathbf{x}_k$: State vector at time step k,
$\mathbf{y}_k$: Measurement vector at time step k,
$\mathbf{q}_{k-1}$: Process noise at time step $k-1$, $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$,
$\mathbf{r}_k$: Measurement noise at time step k, $\mathbf{r}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$,
$f(\cdot)$: Nonlinear system dynamics function,
$h(\cdot)$: Nonlinear measurement model function.

Table 2. EKF for nonlinear systems and RTS algorithms. The EKF operation is explained in subsection 2.2 and the RTS algorithm in subsection 2.2.

Step	**EKF**	**RTS**
0 Initialization	for $k = 0$ set $\hat{\mathbf{X}}_\mathbf{0}, \mathbf{P}_0^-, \mathbf{Q}_0, \mathbf{R}_0$	$\mathbf{x_k}$: State estimate at time step k $\mathbf{P_k}$: Covariance of the state estimate at time step k $\mathbf{A}$: State transition matrix **Smoothing step:** k: Current time step in the backward recursion $k+1$: Subsequent time step
1 Prediction step	Prior estimate of the state: $\mathbf{m}_k^- = f(\mathbf{m}_{k-1}, k-1)$ Prior estimate of the covariance: $\mathbf{P}_k^- = \mathbf{F}_x(\mathbf{m}_{k-1}, k-1)\mathbf{P}_{k-1}\mathbf{F}_x^T(\mathbf{m}_{k-1}, k-1) + \mathbf{Q}_{k-1}$	Prior estimate of the particle positions: $\mathbf{x}_k^i := f(\mathbf{x}_{k-1}^i, k-1)$
2 Update step	Measurement residual update: $\mathbf{V}_k = \mathbf{y}_k - h(\mathbf{m}_k^-, k)$ Measurement covariance update: $\mathbf{S}_k = \mathbf{H}_x(\mathbf{m}_k^-, k)\, \mathbf{P}_k^-\, \mathbf{H}_x^T(\mathbf{m}_k^-, k) + \mathbf{R}_k$ Kalman gain calculation: $\mathbf{K}_k = \mathbf{P}_k^-\, \mathbf{H}_x^T(\mathbf{m}_k^-, k)\, \mathbf{S}_k^{-1}$ Updating the posterior state: $\mathbf{m}_k = \mathbf{m}_k^- + \mathbf{K}_k\mathbf{V}_k$ Updating the posterior covariance: $\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T$	Compute the smoothing gain: $\mathbf{G_k} = \mathbf{P_k}\mathbf{A}^\top\mathbf{P_{k+1}^{-1}}$ Update the smoothed state estimate: $\mathbf{x_k^s} = \mathbf{x_k} + \mathbf{G_k}(\mathbf{x_{k+1}^s} - \mathbf{A}\mathbf{x_k})$ Update the smoothed covariance: $\mathbf{P_k^s} = \mathbf{P_k} + \mathbf{G_k}(\mathbf{P_{k+1}^s} - \mathbf{P_{k+1}})\mathbf{G_k^\top}$
	Return to step 1, repeat for k positioning steps.	**Return to step 1, repeat for k positioning steps.**
Output	Estimated state vector: $\hat{\mathbf{X}}$	Smoothed state estimate: $\mathbf{x_k^s}$ Smoothed covariance estimate: $\mathbf{P_k^s}$

In EKF, the state transition matrix ***F*** and the measurement matrix ***H***, which are utilized in the linear Kalman filter, are replaced by the nonlinear state transition function $f(.)$ and the nonlinear measurement function $h(.)$, respectively. These nonlinear mappings enable the algorithm to process Gaussian distributions in nonlinear system conditions effectively.

In the Kalman filtering process depicted in Figure 3, $m_{\bar{k}}$ and $P_{\bar{k}}$ represent the predicted mean and covariance of the system state at time step *k*, prior to incorporating the measurement, m_k and P_k are the updated mean and covariance estimates after incorporating the measurement at time step *k*. y_k is the measurements vector of the system at time step *k*. S_k is the measurement prediction covariance at time step *k*, reflecting the expected uncertainty in the measurements at time step. K_k known as

the filter gain, is the correction coefficient applied to refine the predictions at time step k.

B. Rauch-Tung-Striebel Smoother (RTS): The Rauch-Tung-Striebel (RTS) smoother is a recursive smoothing algorithm that refines estimates from a Kalman filter by applying backward recursion, which uses future observations to improve state estimates. The RTS smoother uses both past and future data to refine the state estimates, yielding more accurate results than the forward-only Kalman filter, especially in the presence of noise and uncertainty.

Farrell and Barth (Farrell and Barth (1999)) present RTS smoother as an effective post-processing algorithm that follows a forward pass of the Kalman filter, which estimates position and velocity based only on current and past measurements. The RTS backward pass then revises these estimates by using future measurements, which significantly improves the precision and stability of the position estimates, particularly in environments where GNSS data may be intermittent or noisy (e.g., urban areas or GNSS-denied settings). Integration of GNSS and INS benefits substantially from RTS smoothing, as inertial sensors alone can accumulate errors over time. By smoothing, the system can reduce this drift in positioning, making it valuable for applications such as aircraft navigation, autonomous vehicle guidance, and other critical uses where precise positioning is required (Farrell and Barth (1999)).

2.3 Local and global positioning systems

Local positioning systems (LPS) offer a range of solutions: UWB, radar, and laser-based methods provide high-accuracy distance measurements using radio, sound, or light waves, making them effective in real-time tracking and obstacle detection in complex environments. Systems such as INS and image-based localization use motion sensors or visual data for continuous tracking, although INS may suffer from drift without external updates, while image-based methods are advantageous in visually rich settings (Zafari et al. (2019)).

These LPS deliver high-accuracy distance measurements that support real-time tracking and obstacle detection in complex environments. Global positioning systems, such as GPS, Galileo, GLONASS, and BeiDou, provide broad-area outdoor coverage but can integrate with local systems or use methods such as INS, UWB, and image-based localization for seamless indoor-outdoor transitions, addressing limitations in specific environments and improving accuracy (Farrell and Barth (1999)).

In this section, we have discussed only the positioning systems used in this research thesis, including UWB, INS, and GNSS.

2.3.1 Local positioning systems

In this section, we discuss UWB and INS in more detail. Ultra-Wideband (UWB) offers precise, interference-resistant indoor positioning by calculating time-of-arrival (TOA) or time-difference-of-arrival (TDOA) across a wide frequency spectrum, making it effective for complex environments and real-time tracking. In contrast, Inertial Navigation Systems (INS) rely on accelerometers and gyroscopes to provide continuous tracking through dead reckoning, but they can accumulate errors over time without periodic corrections, often from external sources like GNSS.

A. Ultra-Wideband (UWB)
Ultra-Wideband (UWB) is a radio-based positioning technology that transmits short pulses over a broad frequency spectrum (Wilzeck, Guirao, and Dimitrov (2018)). It offers high precision for indoor applications by utilizing time-of-arrival (TOA) or time-difference-of-arrival (TDOA) measurements. UWB is highly resistant to interference and is well-suited for real-time tracking in complex environments. The wide bandwidth of UWB offers several advantages for positioning (Alarifi et al. (2016); Nekoogar (2005)). Its short pulse duration allows for exceptional range accuracy, achieving a precision of approximately 2 centimeters in ideal conditions. In typical indoor environments with multipath fading, the accuracy ranges from 10 to 30 centimeters. Moreover, UWB technology enables low-power operation for tags, particularly when employing the TDOA method.

A key advantage of UWB lies in its robustness in indoor settings, where multipath propagation is a significant source of error. Unlike narrowband systems such as GPS, where reflected signals can distort the direct path and hinder accurate timing, UWB can better differentiate the direct path signal from reflections. This capability simplifies pulse timing and improves accuracy. Despite its benefits, UWB has some limitations. The technology may interfere with, or be affected by, other systems operating in the same vicinity. Although UWB is designed for coexistence, such interference is still possible (Alarifi et al. (2016); Miller (2003)). Additionally, the low transmission power of the UWB can pose challenges in large indoor spaces (Santhanam (2011)). Its relatively short signal range necessitates a higher number of anchors, increasing the system's complexity. Developing outdoor positioning systems with UWB is also challenging due to regulatory restrictions on fixed outdoor transmitters (Standard (2016)).

Like most positioning systems that utilize RF signals, UWB encounters challenges in non-line-of-sight (NLOS) conditions. In such scenarios, the direct path of the signal is obstructed, causing the receiver to rely on the reflected signals. This reliance on reflected signals can introduce significant errors in position estimation. To address this, various strategies have been developed for NLOS identification and error mitigation.

Some approaches focus on analyzing range estimations and channel statistics. By examining variances in range measurements or channel parameters—such as kurtosis, mean delay, excess delay, amplitude, and signal-to-noise ratio (SNR)—NLOS signals can be detected and distinguished from line-of-sight (LOS) signals (Guvenc, Chong, and Watanabe (2007); Khodjaev, Park, and Malik (2009)). In addition, techniques such as residual analysis (S. Yang and Wang (2017)) and receiver autonomous integrity monitoring (RAIM) (Xhafa, del Peral-Rosado, Seco-Granados, and Lopez-Salcedo (2021)) have been employed to identify NLOS signals. Once identified, these signals can either be assigned smaller weights or excluded from the position estimation process.

Recently, machine learning (ML) approaches have gained prominence in addressing NLOS challenges. Techniques such as long-short-term memory networks (LSTM) (Poulose and Han (2020); H. Wang, Wang, Xue, and Jiang (2020)), support vector machines (SVM) (Lei et al. (2020); Rana et al. (2017); X. Yang, Zhao, and Chen (2018)), and convolutional neural networks (CNN) (Lei et al. (2020); Park, Nam, Choi, Ko, and Ko (2020)) have been effectively applied to NLOS identification and error mitigation tasks.

B. Inertial Navigation Systems (INS)

Inertial Navigation Systems (INS) determine an object's position by integrating data from accelerometers and gyroscopes, relying on the dead reckoning principle. Although INS offers continuous tracking capabilities, it suffers from cumulative errors over time due to drift, which necessitates external corrections to maintain accuracy.

Inertial navigation operates as a dead reckoning method by observing the kinematic state through accelerometers and gyroscopes. Starting from a known initial orientation and velocity, it tracks the object's motion relative to the starting point. Although it can serve as the primary navigation method, drift-induced errors often require INS with other technologies, such as GNSS and UWB, to improve long-term accuracy.

I. Inertial measurement unit (IMU): An inertial measurement unit (IMU) typically consists of three orthogonal gyroscopes and three orthogonal accelerometers. These components measure angular velocity and linear acceleration along three coordinate axes, respectively (Woodman (2007)). By integrating the acceleration twice and the angular velocities once, the IMU can estimate the object's position and orientation. However, due to error accumulation during integration, most low-cost IMUs are limited to determining the attitude reliably, whereas position estimates are accurate only for short durations.

Accelerometers: Mechanical accelerometers determine acceleration by measuring the force F acting on a proof mass m, using Newton's second law: $a = F/m$ (Bao, Li, Qiao, and Rauschenbach (2020); Grewal, Weill, and Andrews (2007)). Solid-state accelerometers, such as those based on surface acoustic waves, detect acceleration through frequency shifts. MEMS accelerometers combine principles from

both mechanical and solid-state designs. Accelerometers are commonly classified into three types: (1) mechanical, (2) solid-state, and (3) micro-electromechanical systems (MEMS). Solid-state variants include surface types of acoustic wave, vibratory, silicon, and quartz (El-Sheimy and Youssef (2020); Woodman (2007)).

Gyroscopes: Gyroscopes are used to measure the angular velocity of a moving platform (Titterton, Weston, and Weston (2004); Woodman (2007)). Mechanical gyroscopes employ spinning wheels mounted on gimbals to determine orientation. MEMS-based gyroscopes, along with optical gyroscopes, use the Coriolis effect and light interference, respectively, to measure angular velocity. The precision and performance of gyroscopes are strongly influenced by their cost, as well as environmental factors, which may introduce undesirable effects (El-Sheimy and Youssef (2020)). Various types of gyroscope, including dynamically tuned, vibratory, optical, and fluid-based angular rate sensors, each operate using distinct technologies and principles. Additional details on these types of gyroscopes and accelerometers can be found in (El-Sheimy and Youssef (2020); Grewal et al. (2007); Titterton et al. (2004); Woodman (2007)).

II. Applications of IMUs: IMUs are frequently used to enhance other navigation systems such as GNSS, improve accuracy, and ensure continuous navigation even when GNSS signals are obstructed. One of the key benefits of using an IMU is that, once initialized, it operates independently of external references and is immune to interference. However, like other dead-reckoning systems, IMUs are susceptible to integration drift. Small errors in acceleration accumulate over time, leading to linearly increasing errors in velocity and quadratically increasing errors in position (Ali and Ullah Baig Mirza (2010)).

Beyond inertial navigation, IMU sensor data is also useful for pedestrian dead reckoning (PDR). PDR systems use accelerometer data to estimate the number of steps taken and the distance traveled (Groves, Pulford, Littlefield, Nash, and Mather (2007)).

2.3.2 Global Navigation Satellite Systems (GNSS)

Global Navigation Satellite Systems (GNSS) are satellite-based systems that provide positioning, navigation, and timing (PNT) data to Earth receivers. These are Medium Earth Orbit (MEO) satellites. MEO satellites typically operate at altitudes ranging from 2,000 to 35,786 km above the Earth's surface. These orbits are widely used in navigation satellite systems, such as the GPS, GLONASS, and Galileo constellations, because of their ability to offer a good balance between coverage, orbital speed, and signal strength. They allow users to determine their precise location (latitude, longitude, and altitude) using a GNSS enabled device. GNSS technology has a wide range of applications, including in fields such as transportation, agriculture,

aviation, environmental monitoring, and emergency response.

Working Concept of GNSS - Overview

GNSS, such as GPS, Galileo, GLONASS, and BeiDou, operate by transmitting radio signals from satellites to receivers on Earth. These signals are sent over specific frequency bands—primarily within the **L-band** (1–2 GHz)—which are ideal for penetrating the atmosphere and minimizing signal degradation.

1. Signal Transmission

- Each GNSS satellite continuously broadcasts signals on **multiple frequencies** (e.g., L1, L2, L5).
- These signals contain:
  - **Satellite position and time information**
  - **Navigation message** for orbital data
  - **Pseudorandom codes** for identifying satellites and measuring signal travel time

2. Positioning Mechanism

- A GNSS receiver picks up signals from **at least four satellites**.
- By calculating the **time delay** between transmission and reception, the receiver determines its **distance** from each satellite.
- Using **trilateration**, it computes its precise location (latitude, longitude, altitude) and time.

3. Role of Frequency Bands

- Different frequencies serve different purposes:
  - **L1**: Common for civilian use; carries coarse acquisition (C/A) code.
  - **L2**: Used for ionospheric correction and military signals.
  - **L5**: Designed for safety-critical applications with greater accuracy and robustness.

- Multi-frequency reception allows receivers to **correct for atmospheric errors**, especially ionospheric delay, improving accuracy and reliability.

4. Interference Sensitivity

- GNSS signals are **extremely weak** when they reach Earth.
- They are vulnerable to the following:
  - **Unintentional interference** from nearby electronics
  - **Intentional jamming or spoofing**
  - **Multipath effects**, where signals bounce off surfaces before reaching the receiver

Understanding the frequency structure is crucial for designing GNSS receivers that can **filter noise**, **mitigate interference**, and **enhance positioning performance** in diverse environments.

The important concepts in GNSS are discussed in the following:

A. GNSS constellation:
A GNSS constellation refers to a group of satellites that provide global navigation and positioning services. The main GNSS systems are:

1. GPS (Global Positioning System): Operated by the United States, GPS was the first widely available satellite navigation system and remains one of the most widely used.
2. GLONASS (Globalnaya Navigazionnaya Sputnikovaya Sistema): Operated by Russia, GLONASS provides global coverage similar to GPS.
3. Galileo: Operated by the European Union, Galileo aims to offer high accuracy and reliability, independent of GPS or GLONASS.
4. BeiDou: Operated by China, BeiDou also provides global navigation coverage, with a focus on the Asia-Pacific region.

In addition to global navigation satellite systems, there are regional systems such as Japan's Quasi-Zenith Satellite System (QZSS) and the Navigation with Indian Constellation (NavIC). Regional satellite-based augmentation systems (SBAS) such as the European Geostationary Navigation Overlay Service (EGNOS) and the U.S.

Wide Area Augmentation System (WAAS) enhance GNSS accuracy and reliability (Elliott D. and Christopher J. (2017)).

Each GNSS system is composed of three key segments: space, control, and user, as illustrated in Figure 4, adapted from (M. Y. Yang et al. (2019)). The space segment includes a constellation of satellites that broadcast radio signals to users. The control segment is made up of a global network of monitoring and control stations that track the satellites, supervise their transmissions, and send updates and commands. The user segment consists of GNSS receivers, which determine the user's position and time by processing the satellite signals.

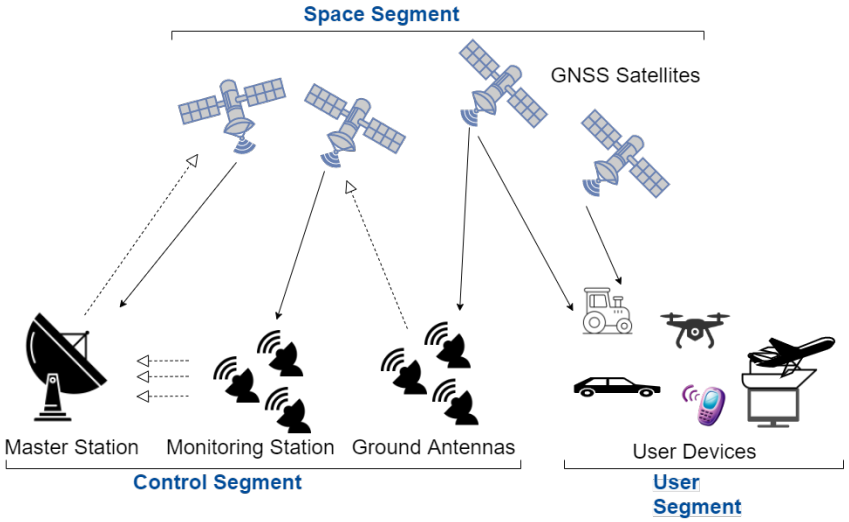


Figure 4. Typical GNSS constellation segments, adapted from M. Y. Yang et al. (2019).

I. Galileo Galileo, a European Global Navigation Satellite System (GNSS), offers free positioning and timing services. The Galileo constellation consists of 30 satellites, with 24 operational satellites and six spares (European Space Agency (2021)). The Galileo navigation signals are broadcasted across frequency bands including E5a, E5b, E6, and E1 (European Space Agency (2011)) (See Figure 5).

Galileo will provide the following services (EUSPA (2022)):

- Open service (OS): A free, open service that offers positioning and timing.
- High-accuracy service (HAS): A free service that provides high-accuracy corrections via the Galileo signal (E6-B) and the internet.
- Public regulated service (PRS): A service for government-authorized users only.
- Search and Rescue Service (SAR): A global satellite-based system for search, rescue, and distress alert detection.

- Commercial authentication service (CAS): A service that complements the OS, providing controlled access and authentication functionality.

II. GPS Introduced in the 1970s, the Global Positioning System (GPS) offers two main services: the Precise Positioning Service (PPS) and the Standard Positioning Service (SPS). PPS is a restricted, encrypted service available only to the United States military, its allies, and certain government agencies. SPS, on the other hand, is a free service that provides positioning data to civil users globally (NOAA (2022)). Both services deliver navigation signals to determine position, velocity, and time. GPS satellites transmit signals across L1, L2, and L5 frequencies. The United States ensures the availability of at least 24 operational GPS satellites 95% of the time, with the current constellation consisting of 31 operational satellites.

III. BeiDou The BeiDou Navigation Satellite System (BDS) is China's global navigation system. Similar to GPS, GLONASS, and Galileo, BeiDou is designed to be compatible and interoperable with these systems. The BDS space segment includes satellites in geostationary Earth orbit, inclined geosynchronous orbit, and medium Earth orbit (MEO) (European Space Agency (2018)). The open service, which is similar to GPS and Galileo, is free and accessible worldwide, while the authorized service is global but restricted in access.

IV. GLONASS The Russian Global Navigation Satellite System (GLONASS) offers multifrequency services for positioning, navigation, and timing, for both military and civilian use. GLONASS satellites currently use frequency division multiple access (FDMA) for signal transmission (European Space Agency (2019)). Each GLONASS satellite transmits P-code on the L1 and L2 frequencies, while all satellites broadcast the C/A code on L1, with most also on L2. The upcoming GLONASS-K satellites will feature additional signals. GLONASS-K1 will transmit a CDMA signal on a new L3 frequency, and GLONASS-K2 will also use CDMA signals on the L1 and L2 frequencies.

B. GNSS Frequencies and Signals

GNSS satellites transmit navigation signals across multiple frequency bands, primarily within the L-band spectrum. These signals are used for positioning, navigation, and timing across various applications. Figure 5 illustrates the frequency allocations used by major GNSS constellations, and Table 3, summarizes the civil signals available for positioning.

Due to their low power levels when received on Earth, GNSS signals are highly susceptible to interference. Common sources include unintentional emissions from electronic devices, overlapping transmissions from other radio services, and environmental effects such as multipath propagation; intensional jamming or spoofing done by malicious entities and multipath effects, where signals bounce off surfaces before reaching the receiver. Interference can degrade signal quality, reduce po-

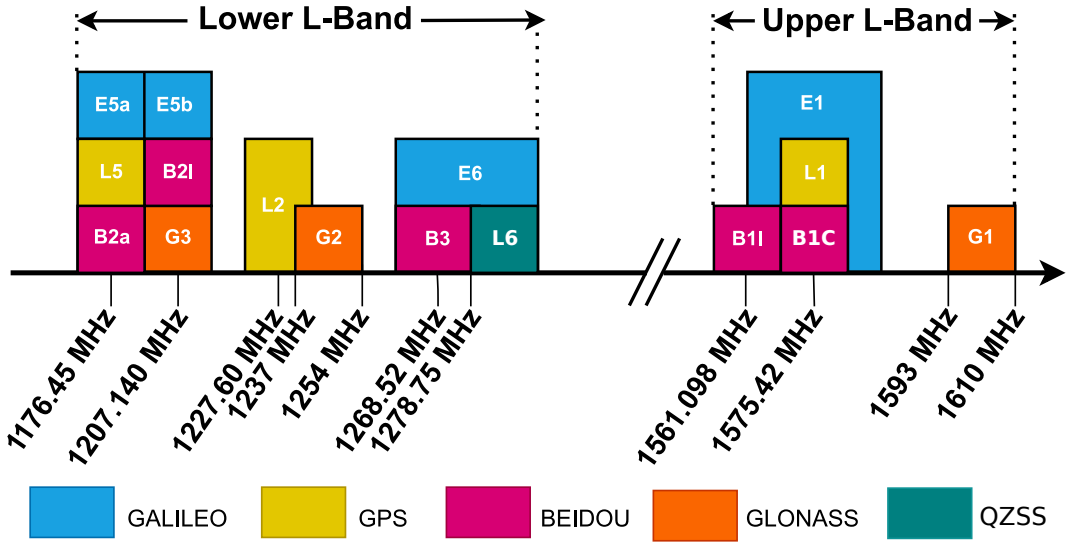


Figure 5. GNSS frequencies in the L band, adapted from *Europa* (2021).

sitioning accuracy, or cause complete signal loss. Understanding the frequency structure of GNSS systems is essential for designing resilient receivers and implementing effective mitigation techniques.

C. Multi-constellation GNSS receivers

Multi-constellation GNSS receivers can receive signals from multiple satellite constellations, such as GPS, Galileo, BeiDou, and GLONASS. By leveraging signals from several constellations, these receivers improve the spatial distribution of visible satellites and reduce the geometric dilution of precision (GDOP), which enhances both the accuracy and reliability of the positioning, especially in environments with limited sky visibility (Shabnam, Chowdhury, Tushar, Sultana, and Hossam-E-Haider (2017)). Additionally, multi-constellation receivers offer increased resistance to spoofing attacks due to the larger number of redundant measurements (Zhang and Papadimitratos (2019)). When utilizing two constellations, only one additional satellite is required to estimate the time offset between them.

D. Multi-frequency GNSS receivers

Previously, GNSS receivers in consumer devices, such as smartphones and automobiles, used a single frequency for positioning, with only high-end professional devices employing multiple frequencies. Today, however, low-cost multifrequency GNSS receivers are capable of achieving centimeter-level accuracy (Dabove and Pietra (2022)). One major advantage of multi-frequency GNSS receivers is the compensation of ionospheric delay. Since the ionospheric delay varies with frequency, it can be mitigated by comparing measurements from two frequencies. Furthermore, the new wideband signals, such as those in the L5 and E5a bands, inherently provide noise and multipath mitigation capabilities (Hexagon (2021)). These receivers also offer better immunity to interference. If one frequency experiences interference, the

Table 3. GNSS frequencies and signals.

System	Frequency	Description
GPS	L1	C/A: First (legacy) civil signal. L1C: Fourth civil signal designed for interoperability with other GNSS. The first GPS satellite featuring L1C launched in December 2018.
	L2	L2C: Second civil signal. Available since April 2014 (preoperational).
	L5	Third civil signal, which features higher power, greater bandwidth, and an advanced signal design. Available since April 2014 (preoperational).
Galileo	E1	Open service. Open service navigation message authentication (OSNMA).
	E5a	Open positioning service.
	E5b	Open positioning service.
	E6	Galileo HAS and commercial augmentation service (CAS).
GLONASS	G1	C/A code for civil users (FDMA). GLONASS-K2 will transmit also CDMA signals.
	G2	C/A code for civil users (FDMA). GLONASS-K2 will transmit also CDMA signals.
	G3	GLONASS-K1 will transmit CDMA signals.
BeiDou	B1I	Open positioning service.
	B1C	Open positioning service transmitted by the BDS-3M satellites.
	B2I	Open positioning service.
	B2a	Open positioning service transmitted by the BDS-3M satellites.
	B3I	Open positioning service signal transmitted by the BDS-2 and BDS-3 satellites.
QZSS	L1	Similar open service signals as in GPS L1.
	L2	Similar open service signals as in GPS L2.
	L5	Similar open service signals as in GPS L5.
	L6	L6 signal provides centimeter-level augmentation service (CLAS).
NavIC	L5	Open positioning service.
	S	Open positioning service.

receiver can still track the remaining frequency bands.

E. Low-cost GNSS receivers for mass-market applications

Historically, most consumer-grade GNSS receivers could only track the GPS L1 signal, with an accuracy of around 4 meters in stand-alone mode and 1 meter with differential corrections. However, with the advent of low-cost precise-positioning GNSS receivers, this has changed. These receivers can now track frequencies from multiple GNSS constellations, similar to high-end survey-grade GNSS devices. Low-cost, multi-constellation, and multifrequency GNSS receivers are now capable of providing centimeter-level accuracy with relatively high reliability, especially in static scenarios (Biagi, Grec, and Negretti (2016); Dabove and Pietra (2022)). Additionally, the quality of the position solution is influenced by the GNSS antenna's performance. Currently, affordable, multifrequency GNSS antennas are available that offer performance on par with geodetic antennas (Hamza, Stopar, and Sterle (2021)). While low-cost GNSS instruments may not match the performance of high-end geodetic systems, they still provide sufficient precision for a variety of applications, particularly given their cost-effectiveness.

F. GNSS errors

The major sources of errors in the measured pseudorange for GNSS systems are primarily due to various environmental and system-related factors. These errors, as illustrated in Figure 6 and summarized in Table 4, typically include:

Table 4. GNSS error sources, their properties, and correction methods.

Error	Locality	Correction method
Satellite clock error	Global	SBAS, DGNSS, RTK, PPP, HAS
Ephemeris error	Global	SBAS, DGNSS, RTK, PPP, HAS
Ionospheric error	Regional	SBAS, DGNSS, RTK, PPP, HAS*
Tropospheric error	Regional	DGNSS, RTK, PPP, HAS*
Receiver noise and resolution	Local	Sensor fusion
Multipath error	Local	Sensor fusion

* = *HAS SL2 in Europe*

These sources of errors can be mitigated through various correction methods such as differential GNSS (DGNSS), carrier-phase measurements, or using augmentation systems like WAAS or EGNOS, as discussed in section 2.3.3. For high-accuracy applications, combining multiple sources of data, such as using multi-frequency and multi-constellation receivers, helps to reduce the impact of these errors (Elliott D. and Christopher J. (2017)). The total error affecting the pseudorange between a satellite and a user's receiver is summarized as the User Equivalent Range Error (UERE). This error impacts the position accuracy, which is influenced by both satellite geometry (as represented by Position Dilution of Precision, or PDOP) and the pseudorange error (UERE).

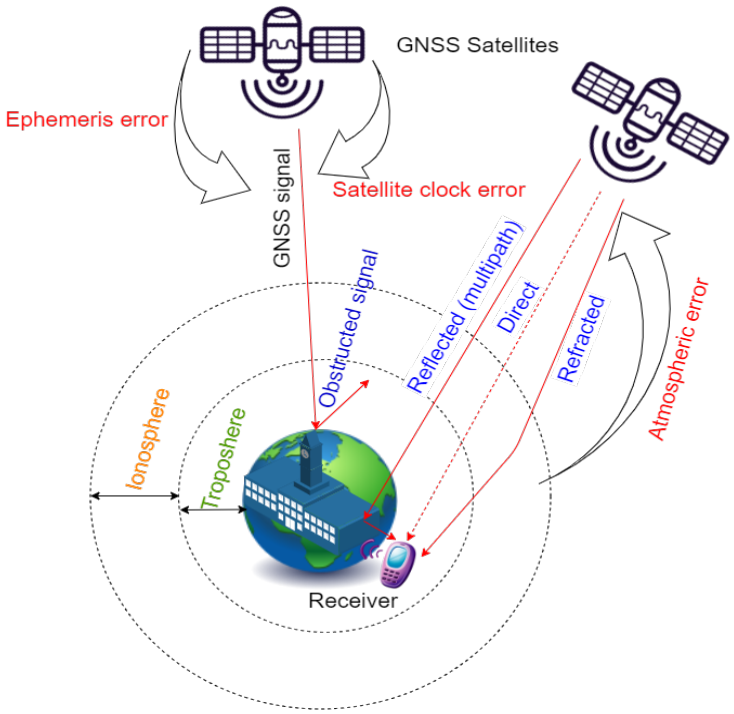


Figure 6. GNSS errors, adapted from Biswas (2017).

2.3.3 GNSS positioning modes

Various GNSS positioning modes are used to estimate positions, each offering different levels of complexity and accuracy (P. J. Teunissen and Montenbruck (2017)). These range from basic Standard Point Positioning (SPP) typically used in consumer devices to more advanced methods like carrier-phase-based techniques that can achieve centimeter-level accuracy. One such technique is Precise Point Positioning (PPP), and although Galileo's High-Accuracy Service (HAS) is based on PPP, it is often treated as a distinct positioning mode due to its specific characteristics. The properties and differences among these positioning modes are summarized in Table 5, adapted from (European GNSS Agency (2020); European Union Agency for the Space Programme (2020)).

The most common positioning modes are:

- Standard point positioning (SPP)
- Differential GNSS (DGNSS)
- Satellite-based augmentation system (SBAS)
- Real-time kinematic positioning (RTK)

Table 5. Major GNSS positioning modes.

Method*	**SPP**	**DGNSS**	**SBAS**	**RTK**	**PPP-RTK**	**PPP**	**HAS**
Observable	Code	Code	Code	Code/Carrier	Code/Carrier	Code/Carrier	Code/Carrier
Positioning	Absolute	Relative	Relative	Relative	Absolute	Absolute	Absolute
Correction format	–	OSR	SSR	OSR	SSR	SSR	SSR
Communication link for corrections	No	External	GNSS-like SBAS signal	External	External	External	Galileo E6B or Internet
TTFF / Convergence time	Rx TTFF	SPP + correction delay	As DGNSS	DGNSS + ambiguity fixing	Faster than PPP	Slower than RTK	Similar to PPP-RTK
Horizontal accuracy	5 m	1–5 m	1 m	1 cm + 1 ppm	10 cm	10 cm–1 m	20 cm
Coverage	Worldwide	$\leq$100 km	$\leq$1000 km	$\leq$10 km	Regional	Worldwide	Worldwide

** Acronyms are defined in Subsection 2.3.3*

- Precise point positioning (PPP)
- Precise point positioning–real-time kinematic positioning (PPP-RTK)
- Galileo high-accuracy service (HAS)

Galileo HAS is one implementation of PPP, but it is considered a separate positioning mode in this chapter. The properties of the positioning modes are summarized in Table 5.

2.4 Low Earth Orbit navigation satellites

Low-Earth Orbit (LEO) refers to an orbital range relatively close to Earth's surface, typically at altitudes between 160 km and 2,000 km. Satellites in LEO travel at high speeds, approximately 7.8 km per second, completing an orbit around the Earth in about 90 minutes. While traditionally used for telecommunications, LEO is gaining attention for its potential in navigation applications, with plans for LEO-based satellite constellations for global navigation services. These satellites' rapid movement across the sky presents challenges for continuous coverage, which can be addressed by using large constellations of satellites. The advantages of LEO, such as lower latency and reduced signal attenuation, offer significant benefits for applications like global communications and precise navigation (ESA (2020)).

Satellites in Low Earth Orbit (LEO) and Medium Earth Orbit (MEO) each offer distinct advantages and limitations in navigation applications, as summarized in Table 6, adapted from (Lawrence et al. (2017)). Ground-based receivers experience reduced signal attenuation and stronger signal strength when communicating with LEO satellites compared to MEO satellites.

This difference arises because LEO satellites are positioned closer to Earth, resulting in lower signal spreading or space loss. The typical signal loss for LEO satel-

Table 6. Comparison of LEO and MEO systems for navigation.

Characteristic	**MEO**	**LEO**	**LEO:MEO Ratio**
System	*GNSS*	*Iridium*	-
Altitude, 10^3 km	20	0.78	1:25
Zenith loss, dB	-97	-69	28
Footprint area, 10^7 km^2	17	1.9	1:9
Footprint radius, km	7900	2500	1:3
Mean motion, °/s	0.008	0.06	7:1
Orbital period, h	12	1.7	1:7
Multipath decorrelation time, min	10	1	10:1
RTK initialization time, min	30	5	6:1

lites ranges between -140 and -130 dB, whereas for GNSS signals transmitted from MEO satellites, it is approximately -160 dB. Another key distinction lies in satellite speed. LEO satellites complete an overhead pass in about 100 minutes due to their higher angular velocity, whereas MEO satellites require around 12 hours. The rapid movement of LEO satellites can enhance the handling of tropospheric delay variations by improving the spatial-temporal decorrelation of these delays (Prol et al. (2022)).

Coverage is another significant factor. LEO satellites have a footprint only about one-ninth the size of that of MEO and GEO satellites, as depicted in Figure 7. Consequently, a larger constellation of LEO satellites is necessary to achieve coverage comparable to that of smaller MEO constellations (Lawrence et al. (2017)).

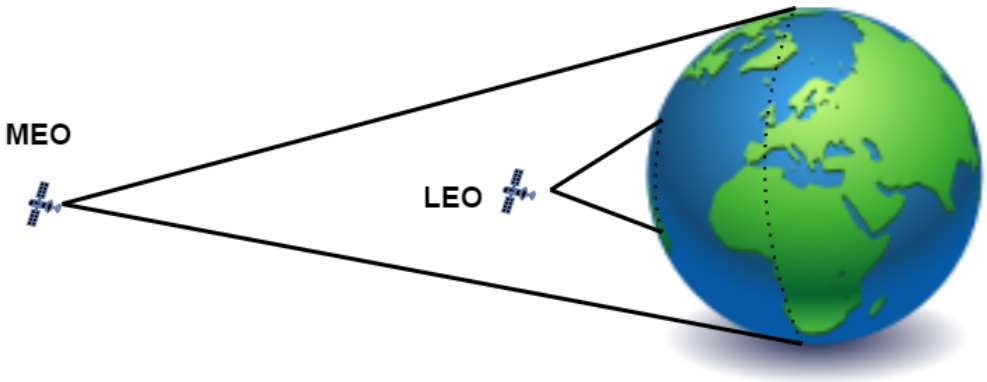


Figure 7. Satellite footprint comparison in LEO and MEO.

The stronger signal strength, pronounced Doppler effect, and higher angular velocity of LEO satellites present significant opportunities for the development of more robust and cost-effective positioning receivers. These characteristics can enable the design of receivers that are not only reliable but also better suited for challenging

environments, such as indoor use, in certain scenarios. The higher Doppler shift inherent in LEO signals provides additional information that can be leveraged for improved velocity estimation and navigation accuracy. Furthermore, the increased angular velocity facilitates faster updates, making the system more responsive to changes in position and reducing the impact of multipath effects in some cases (Selvan et al. (2023)). As LEO satellites traverse the sky more rapidly than their MEO or GEO counterparts, the relative geometry between the satellite and receiver changes more quickly. This dynamic geometry allows receivers to collect more frequent and diverse measurements, which improves the temporal resolution of position estimates. In turn, this heightened update rate helps the system adapt swiftly to user movement and environmental changes, while also enabling better discrimination between direct and reflected signals—thereby mitigating the adverse effects of multipath propagation, especially in urban or indoor settings.

2.5 Mixed localization systems

No single positioning technology can consistently deliver accurate and dependable location information across all environments. Each technology has specific strengths and limitations depending on the operational context. For instance, GNSS offers exceptional global performance in open-sky scenarios but struggles indoors or within urban canyons. Conversely, local positioning systems like UWB, Wi-Fi, and Bluetooth function well indoors, but their coverage is constrained by the anchor network. Additionally, all RF-based navigation systems are susceptible to challenges such as multipath effects and non-line-of-sight (NLOS) conditions. In contrast, inertial measurement units (IMUs) are resistant to external interference but experience integration drift, similar to other dead reckoning methods. Integrating data from multiple positioning technologies can enhance the accuracy, availability, and reliability of position estimation (X. Guo et al. (2020a)). These hybrid positioning system provides better performance than a stand-alone technologies. Figure 8 illustrates various positioning technologies applicable in sensor fusion.

Hybrid positioning systems leverage various combinations of positioning technologies to achieve enhanced performance. For instance, inertial measurement units (IMUs) are often integrated with GNSS (Tuan, Hongping, Xiaoji, and Zhouzheng (2017)), UWB (J. Liu, Pu, Sun, and He (2019a); Y. Wang and Li (2017); Zhao, Huang, and Liu (2016)), or Wi-Fi (G. Guo et al. (2022)) positioning systems. Beyond IMUs, other dead reckoning techniques, such as wheel odometry, laser sensors, radar, and visual odometry, are frequently combined with GNSS or UWB for improved accuracy and robustness (Mohamed et al. (2019); Mostafa, Zahran, Moussa, El-Sheimy, and Sesay (2018)). In intelligent vehicles, data fusion from multiple positioning systems is standard practice. For example, Zhu *et al.* (B. Zhu et al. (2020)) explored an integrated positioning strategy that combines GNSS, UWB,

and IMU data with a visual map-matching algorithm for more precise location estimation.

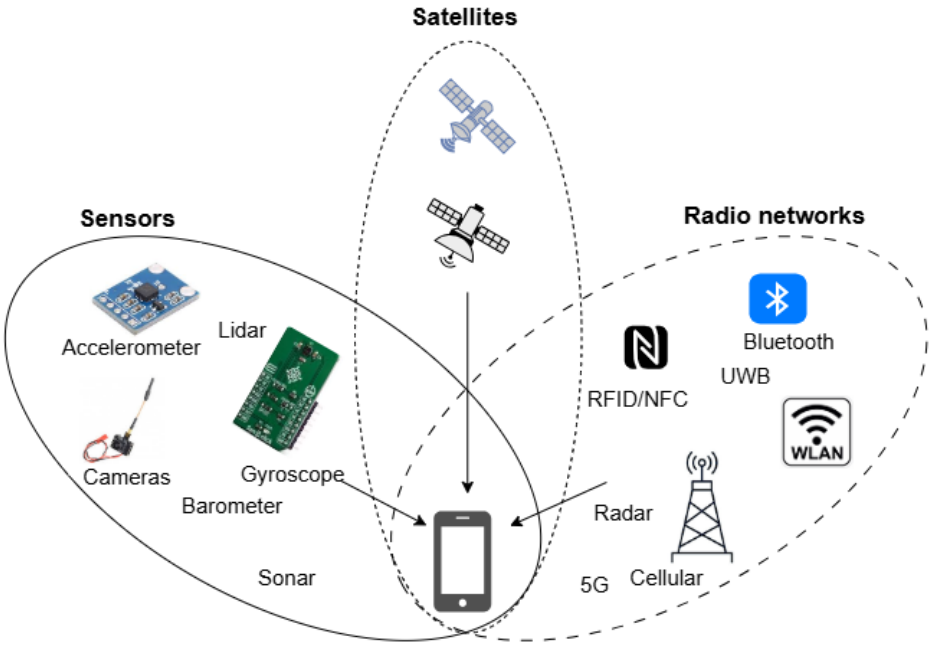


Figure 8. Multiple technologies as input for seamless positioning.

Positioning techniques can be integrated either at the position level or the measurement level. In position-level integration, the location estimated by an individual system is combined with data from other motion sensors. For example, in GNSS-IMU integration, the position derived from GNSS observations is enhanced using IMU data to improve both accuracy and reliability. Conversely, in measurement-level integration, raw sensor measurements are jointly processed to estimate position. This approach is particularly advantageous when there are insufficient TOA/TDOA or AOA/AOD measurements for a standalone position fix. Position-level integration is commonly referred to as loose coupling, whereas measurement-level integration is known as tight coupling.

Hybrid positioning involves combining data from multiple sensors using advanced algorithms. Common sensor fusion methods include the least squares method, Kalman filters (standard, extended, and federated), particle filters, and various machine learning techniques (X. Guo et al. (2020a)). Additionally, the factor graph algorithm is widely applied in sensor fusion and SLAM applications (Carlone, Tron, Daniilidis, and Dellaert (2015); Indelman, Williams, Kaess, and Dellaert (2012)). A detailed explanation of commonly used algorithms can be found in sections 2.5.1, 2.5.2 and 2.5.4.

2.5.1 Loosely coupled GNSS/IMU integration

In principle, orientation can be determined using GNSS by employing multiple antennas (Grimm (2008)), though this is less common in consumer applications. For scenarios where attitude information is critical for control and guidance, GNSS, UWB, Wi-Fi, and Bluetooth DF signals are often integrated with inertial systems.

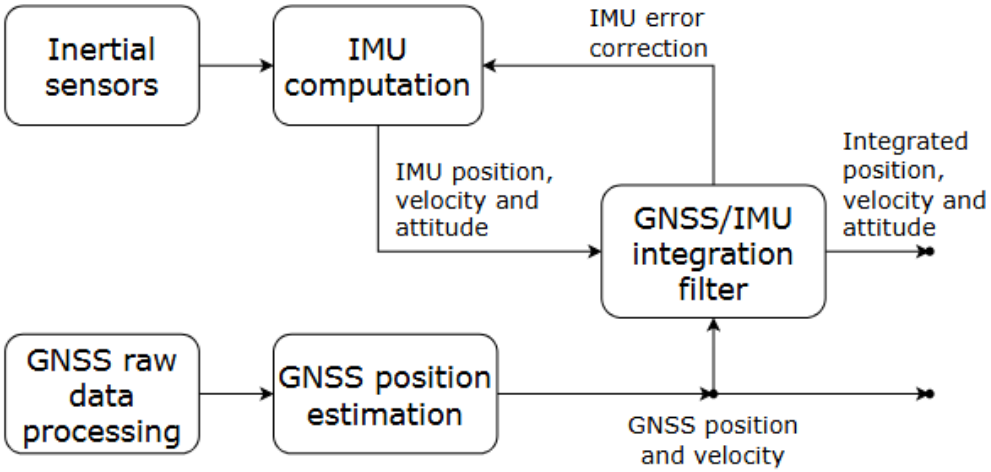


Figure 9. Loosely coupled GNSS/IMU integration.

Another motivation for integrating IMUs arises from the fact that signals used for direct positioning are not always accessible. For example, GNSS signals can be severely impaired or unavailable in indoor environments and urban canyons. Similarly, indoor positioning systems can face issues such as signal reflection, leading to unreliable or unavailable data. As a result, additional technologies are needed to support position estimation in challenging signal conditions. Integration with dead reckoning methods, such as IMUs, can improve the performance of direct position estimation methods.

IMU and direct positioning systems complement each other in terms of the data they provide. IMUs can generate navigation state estimates at data rates exceeding 100 Hz, while most GNSS receivers operate at 1-20 Hz (Ryu, Gankhuyag, and Chong (2016)). The high data rate of IMUs is crucial for guidance, control, and dynamic applications where capturing rapid motion is necessary. Furthermore, IMUs and direct positioning systems offer complementary error profiles: IMU errors tend to be time-correlated and can grow without bounds, whereas direct positioning systems generate position estimates with bounded errors.

IMU data can enhance the resilience of GNSS receivers against jamming or RF interference (Wu, Zhang, Yang, Liang, and Liu (2020)), and GNSS position and velocity information can be used to estimate and mitigate IMU errors (Cao, Guo,

and Chen (2018); Z. Wang, Li, Zhu, Li, and Fang (2022)). IMUs provide high-rate attitude, position, and velocity estimates essential for vehicle guidance and control, while also enabling operation through short-term GNSS signal dropouts (Ryu et al. (2016)).

In loose-coupled integration, the position and velocity from GNSS are combined with the IMU navigation solution using an integration filter, as depicted in Figure 9. The integration filter's estimated error states are fed back into the IMU computation to correct for IMU integration drift. For loose coupling to be effective, a sufficient number of GNSS measurements must be available (Falco, Pini, and Marucco (2017)).

2.5.2 Tightly coupled GNSS/IMU integration

Tightly coupled integration, on the other hand, incorporates raw GNSS data with IMU measurements to estimate position, velocity, and attitude. This method, illustrated in Figure 10, computes predicted ranges and range rates based on satellite and IMU motion information (Zhou, Zhang, Li, and Li (2015)). The integration filter uses the discrepancies between GNSS-observed and IMU-predicted ranges and range rates as inputs, refining the estimation and correcting the IMU output in a feedback loop. A key advantage of this approach is its ability to use raw GNSS data to constrain inertial drift, eliminating the need for a standalone GNSS position solution.

2.5.3 Comparison and application context of loosely and tightly coupled integration

The choice between loosely and tightly coupled integration depends on the operational environment and system requirements. Tightly coupled architectures are generally preferred in high-dynamic or obstructed environments, whereas loosely coupled systems may suffice in open-sky conditions with consistent satellite visibility.

2.5.4 Other GNSS/IMU integration schemes

The third integration scheme is ultra-tight coupled integration. In ultra-tight coupling, the IMU measurements directly assist in signal tracking by removing dynamic effects from the incoming GNSS signal. This tighter integration allows improved signal tracking and noise suppression.

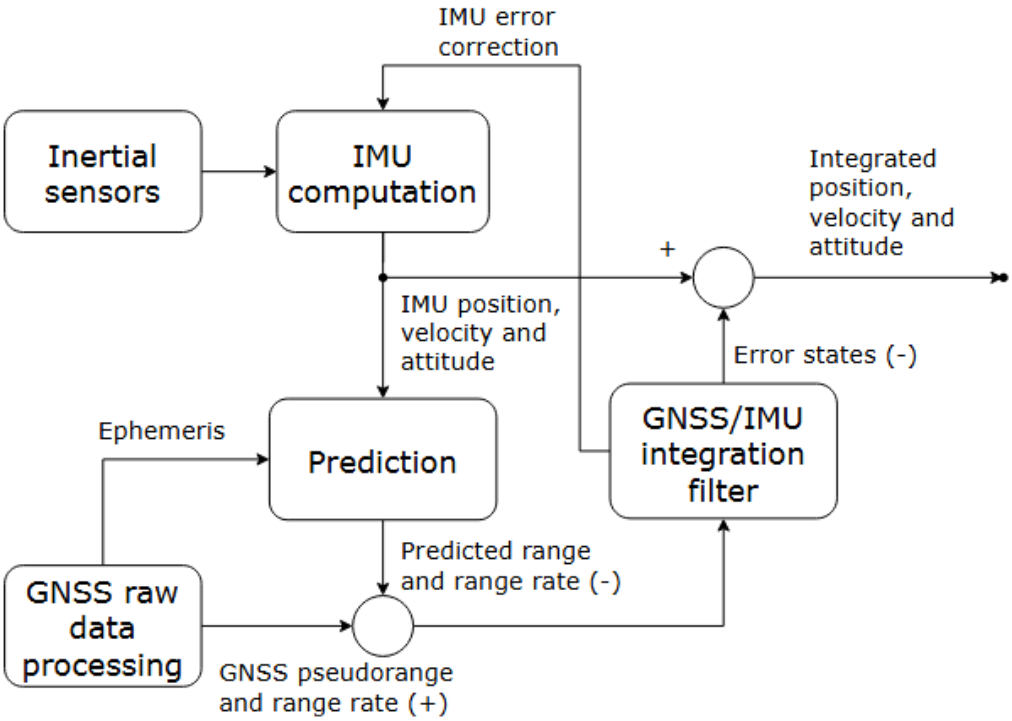


Figure 10. Tightly coupled GNSS/IMU integration.

Kalman filters, in various forms, are commonly employed in GNSS/IMU integration (Falco et al. (2017); Zhou et al. (2015)). Zhang *et al.* (Wen, Pfeifer, Bai, and Hsu (2021)) introduced a loosely coupled GNSS/INS method integrated with a factor graph model, enabling the use of historical measurements for state estimation and prediction of GNSS observations during interruptions. RTK-based positioning also frequently incorporates IMU integration to enhance precision (S. Liu, Li, Zheng, and Fu (2022); Tuan et al. (2017)).

UWB and IMU integration follows a similar principle but lacks Doppler-based velocity observables provided by GNSS. In UWB systems, IMUs are often employed to mitigate non-line-of-sight (NLOS) conditions and improve positioning accuracy (Elsanhoury et al. (2022); LI, XU, SHEN, and BI (2019); J. Liu, Pu, Sun, and He (2019b)). This complementary relationship makes UWB/IMU fusion effective in challenging environments.

2.6 Positioning system requirements

Users of GNSS systems require high accuracy, availability, low cost, and security and privacy assurances. These aspects collectively determine a positioning system's

suitability for various applications, from consumer navigation to industrial monitoring. Each of these requirements has distinct considerations, outlined below: Below, we discuss these important parameters related to the requirements of positioning systems. This breakdown highlights the critical considerations across positioning system requirements, ensuring that systems are designed to meet diverse application needs effectively.

2.6.1 Accuracy

The accuracy of positioning systems varies by application, with high-precision tasks like autonomous vehicles and surveying requiring decimeter- to centimeter-level accuracy, while general navigation can tolerate meter-level precision; GNSS performance is influenced by factors such as multipath and atmospheric conditions, with techniques like RTK used for precision-critical tasks.

The European GNSS Agency highlights the growing demand for affordable precision technologies, noting that for applications such as autonomous vehicles and drones, decimeter-level accuracy is often sufficient, as onboard sensors like lidar manage the detailed navigation tasks. While indoor positioning systems prioritize symbolic locations over exact coordinates for user navigation needs (European GNSS Agency (2019); Tegou, Kalamaras, Votis, and Tzovaras (2018)).

2.6.2 Availability

A dependable GNSS system must provide positioning signals across diverse environments, including indoor, outdoor, and remote areas. However, availability is often hindered by obstructions like buildings or natural terrain. To address this, hybrid positioning solutions, combining GNSS with technologies like cellular networks or Wi-Fi, are frequently used to enhance performance in complex environments.

The growing demand for location-based services that function seamlessly across indoor and outdoor spaces underscores the importance of such hybrid systems. Since no single technology currently excels in both environments and their transitions, integrating multiple positioning technologies is vital for a unified, reliable location solution.

2.6.3 Cost

The cost of GNSS devices varies significantly based on their precision, features, and applications. High-precision tools like RTK GNSS receivers are expensive, often costing thousands of euros, while low-cost GNSS chips in smartphones are under a euro and are enhanced by additional sensors like IMUs, Wi-Fi, and UWB.

Similarly, in local positioning systems, the infrastructure is a major expense, with the number of base stations directly affecting the cost. For example, while a TDOA-based UWB system enables low-cost tags, its infrastructure can be complex and expensive to deploy.

2.6.4 Security and privacy

When tracking individuals or sensitive assets with GNSS systems, privacy is crucial, requiring secure data handling, anonymization, and strict access controls to prevent unauthorized access. Privacy-preserving positioning methods aim to balance accurate location tracking with data protection, addressing the risk of location data being used as quasi-identifiers or revealing sensitive personal details when combined with public information (H. Jiang et al. (2021); Mäkelä (2008)).

Techniques like anonymization, obfuscation, and encryption help safeguard privacy, though practical implementation is still evolving (B. Liu, Zhou, Zhu, Gao, and Xiang (2018)). GNSS self-positioning systems are less privacy-invasive than network-based systems like RFID or fingerprinting, as they calculate location locally, avoiding external storage. Network-based systems, however, generate and store data externally, introducing potential vulnerabilities (Sartayeva and Chan (2023)).

2.7 Machine learning

Research into machine learning (ML) methods for positioning is expanding, driven by advances in ML algorithms and improved computational capabilities of positioning hardware. ML techniques are particularly useful when conventional multilateration assumptions do not hold, such as in the presence of long-tailed or skewed noise distributions or measurement outliers. They enhance robustness by handling outliers, integrating diverse information sources, and leveraging unconventional observations for positioning.

ML-based approaches in positioning include the use of algorithms to exclude noisy measurements, integrate sensor data with contextual information, and adaptively

learn environmental features for enhanced accuracy. These techniques address challenges in environments where traditional methods fall short, improving reliability and precision.

Some examples of ML in positioning are:

- Enhancing the robustness of positioning in the presence of outlying observations by using advanced variable selection algorithms (W. Liu et al. (2019)).
- Implementing an adaptive positioning method that adjusts model parameters based on changes in operating conditions. For example, the positioning method or observables can switch when an agent moves from outdoors to indoors or when an NLOS condition is detected (Carrera Villacrés et al. (2019); Xia and Weitnauer (2019)).
- Collaborative positioning can be employed by sharing information between agents. In such cases, the conditions for standard positioning methods might not be met due to skewed noise distributions and non-convexity. Machine learning-based methods, such as cognitive particle filters, may assist in finding solutions (Minetto, Gurrieri, and Dovis (2020)).
- Machine learning can autonomously learn the model from the problem domain, enabling model-free positioning, which is especially useful when ad hoc information is used. An example of this is Wi-Fi fingerprinting-based positioning (L. Li, Guo, Ansari, and Li (2019)), which utilizes methods like nonlinear ensemble regression (Marthala (2020)).
- Intelligent variable selection from heterogeneous information sources using machine learning can be beneficial when leveraging ad hoc information or addressing NLOS issues in standard positioning infrastructures (Vandermeeren, Van de Velde, Bruneel, and Steendam (2018)). Algorithms commonly used for feature selection include nearest-neighbors, k-means, support vector machines, and outlier detection methods.
- In publication II., the study focuses on enhancing GNSS positioning and navigation accuracy on smartphones by using machine learning (ML). It demonstrates how ML can mitigate errors caused by environmental challenges like multipath interference and atmospheric disturbances. The approach was validated through real-world testing to assess its effectiveness in mobile devices (Siemuri, Selvan, Kuusniemi, Välisuo, and Elmusrati (2021)).
- In publication III., the paper explores the integration of GNSS and IMU data for high-precision positioning on smartphones using machine learning (ML). It highlights how combining these sensor measurements can greatly improve positioning accuracy, particularly in dynamic environments, and introduces

a machine learning model to optimally integrate both data sources, reducing errors and enhancing overall performance (Siemuri, Elsanhoury, Välisuo, Kuusniemi, and Elmusrati (2022)).

- In publication V., this paper focuses on seamless navigation for both indoor and outdoor positioning by integrating GNSS, UWB, Wi-Fi, and IMU systems. It demonstrates how a hybrid positioning approach can provide continuous and accurate location tracking across diverse environments, emphasizing the advantages of combining multiple sensor technologies to overcome the limitations of GNSS in challenging indoor or urban canyon scenarios (Siemuri et al. (2023)).

- In publication VII., this study explores the use of machine learning algorithms to improve the accuracy of orbit prediction for both low Earth orbit (LEO) and medium Earth orbit (MEO) satellites, demonstrating their potential to enhance the reliability of satellite navigation and space-based services.

2.8 Factor graph optimization

Factor graphs (FG) are a relatively new technique in the field of Bayesian filtering and have gained significant attention in the simultaneous localization and mapping (SLAM) domain (e.g., Carlone et al. (2015); Cunningham, Paluri, and Dellaert (2010)). Early studies on the application of factor graphs in positioning focused on addressing multipath effects (Sünderhauf and Protzel (2012)) and integrating data from GPS, IMU, and stereo vision sensors for sensor fusion (Indelman et al. (2012)).

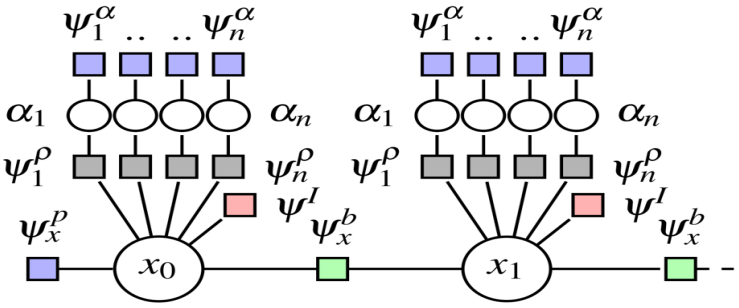


Figure 11. Factor graph for positioning using factors for *a priori* location, ψ^p, IMU sensor ψ^{IMU}, and GNSS pseudorange observations ψ_i^ρ. The states in epoch, i, x_i, are estimated by the FG. The factors are described as influence functions, ψ, which return the probability of the current observation in state x_i and how the x_i should be updated to maximize the probability.

FG is a bipartite graphical model consisting of variables, x_i, and constraint functions or factors, ψ^i, see Figure 11. Each constraint is connected to one or more variables, while each variable can have multiple constraints. This sparse graph represents the factorization of a full probability density function, which is solved iteratively through a message-passing (MP) algorithm (Pearl (1987)). FG provides a general, intuitive graphical representation, capable of modeling Kalman filter (KF) behavior. Moreover, it extends KF by incorporating historical measurements with arbitrary batch sizes, allowing for greater accuracy, particularly in non-line-of-sight (NLOS) scenarios (Pfeifer and Protzel (2019a); Wen et al. (2021)). However, FG is computationally more demanding than recursive KF variants.

FG have recently been applied to GNSS/IMU integration and have been shown to outperform extended Kalman filter (EKF) integration (Wen et al. (2021)). In publication VI., we implemented factor graph optimization to enhance the positioning accuracy of low-cost GNSS devices (Siemuri et al. (2024)).

3 PRECISE POSITIONING WITH LOW-COST GNSS DEVICES USING ML ALGORITHMS

Traditional global navigation satellite system (GNSS) algorithms and models deliver good positioning, navigation, and timing (PNT) performance under favorable signal conditions. However, ongoing research focuses on enhancing their robustness and reliability in challenging signal environments. The growing exploration of machine learning (ML) across diverse fields has spurred significant interest in its application within GNSS, transforming how navigation issues are addressed and resolved. ML is becoming integral to advancing PNT technologies, enhancing GNSS performance and usability. Therefore, it is important to examine how ML improves GNSS capabilities, discuss areas where ML has been successfully applied, compare commonly used algorithms in similar scenarios, and explore the challenges and risks of ML utilization in GNSS. This can help to highlight future opportunities where ML can be applied to improve GNSS performance, accuracy, and robustness, paving the way for innovative research directions.

However, before proceeding we will look at GNSS position estimation and ML techniques overview in section 3.1 and 3.3.

3.1 GNSS position estimation and performance

The increasing complexity and reliance on Global Navigation Satellite System (GNSS) technologies have driven the demand for high-performance GNSS solutions that offer accuracy, availability, continuity, and integrity at lower costs.

For a GNSS receiver to accurately determine its location, it requires three key sets of data: GNSS signals from multiple satellites, almanac data, and ephemeris data. *Almanac data* provides coarse orbital parameters and health information for all satellites in the constellation. It helps the receiver identify which satellites are likely to be visible and worth tracking, thereby speeding up the initial acquisition process (Dems (2010)). *Ephemeris data*, in contrast, contains precise and time-sensitive orbital information for individual satellites, including clock corrections and high-accuracy position data. This information is essential for computing the satellite's location at a specific time and is valid only for short durations (Novatel (2025)). The time it takes for a GNSS-enabled device to acquire sufficient satellite signals and compute its initial position after being powered on is known as the *Time to First Fix* (TTFF). TTFF is a crucial performance metric, particularly for applications that require fast and reliable positioning.

Several factors influence TTFF, including the receiver's current location, satellite

signal quality, processing power, and whether it has prior information such as the last known position, almanac, and ephemeris data. TTFF generally falls into one of three categories:

1. Cold Start: The receiver has no prior information about its location or the satellite constellation, resulting in the longest TTFF, which is typically between 30 seconds and several minutes depending on signal conditions, satellite visibility, and receiver sensitivity.

2. Warm Start: The receiver has some information, such as the last known position and time, but needs to acquire new satellite data, leading to a moderate TTFF, which is typically between 5 and 45 seconds depending on the accuracy of stored data and signal availability.

3. Hot Start: The receiver has recent information about its position and the satellite constellation, allowing for the shortest TTFF, which is typically between 1 and 10 seconds depending on the time elapsed since the last fix and signal strength. In this case, the receiver has both accurate ephemeris and information on frequency offset as well as an accurate initial solution.

TTFF is important for users who need rapid and accurate positioning, such as in navigation systems, emergency response, and various mobile applications. Estimating GNSS performance before system deployment is essential for assessing navigation accuracy and minimizing maintenance efforts and downtime. This means that early detection of faults and errors enables timely corrections, ensuring reliable system operation.

Various performance parameters and error/fault data has been addressed in the literature and these can be used to develop and evaluate models for detecting and correcting GNSS errors. These models can estimate and predict the GNSS performance required by applications relying on GNSS systems to function adequately. Significant sources of errors in satellite-based positioning include ionospheric and tropospheric effects, multipath propagation, clock drift, receiver noise, spoofing, jamming interferences, and hardware biases. Ionospheric content interferes with GNSS signals, inducing errors in position calculations. Additionally, GNSS, designed for ideal line-of-sight conditions, is highly influenced by multipath propagation in areas with high reflection or refraction, such as urban environments. GNSS signals are also vulnerable to radio frequency (RF) interference due to their low power at the Earth's surface. In GNSS-denied environments, signals are unavailable, hindering position calculations. These factors severely degrade receiver performance.

3.2 Challenges of GNSS in indoor & outdoor environment

Global Navigation Satellite Systems (GNSS) are essential for positioning, navigation, and timing (PNT) applications. However, they are highly vulnerable to intentional and unintentional interference, particularly spoofing (deceptive signal injection) and jamming (signal blocking). The challenges differ based on whether the GNSS receiver operates in indoor or outdoor environments.

Outdoor GNSS receivers rely on direct signals from satellites, making them susceptible to large-scale spoofing and jamming attacks.

3.2.1 Outdoor spoofing

- **Signal Authenticity:** GNSS signals are unencrypted in civilian applications (e.g., GPS L1), making them susceptible to spoofing attacks where a fake signal mimics legitimate satellite signals.
- **Coherent Spoofing Attacks:** Advanced attackers can generate counterfeit GNSS signals synchronized with real ones, gradually deceiving the receiver without immediate detection.
- **Meaconing:** Adversaries record genuine GNSS signals and rebroadcast them with a delay, misleading receivers into incorrect positioning.
- **Sophisticated Spoofing with SDRs:** Software-defined radios (SDRs) enable attackers to generate realistic spoofed signals, making detection harder.

3.2.2 Outdoor jamming

- **Low GNSS Signal Power:** GNSS signals are weak when they reach Earth (-130 dBm), making them easy targets for high-power jammers.
- **Wide-area Impact:** Jamming can affect multiple GNSS receivers over a large area, including critical applications like aviation, maritime navigation, and autonomous systems.
- **Unintentional Interference:** Nearby communication systems (e.g., 4G/5G, military radars) can unintentionally interfere with GNSS signals.
- **Denial-of-Service (DoS) Attacks:** High-power jammers can completely disable GNSS receivers, rendering them useless for navigation.

Indoor GNSS applications (e.g., warehouses, urban canyons, smart buildings) face additional challenges due to weak signal reception.

Machine learning (ML) is applied to address GNSS performance degradation, leveraging the vast quantities of data GNSS provides. ML has been used in numerous research studies to propose and provide solutions for GNSS challenges, including addressing sources of errors like ionospheric effects, multipath propagation, spoofing, jamming interferences, receiver noise, satellite clock errors, and hardware biases.

3.3 Machine learning techniques overview

Machine learning (ML) techniques are well-suited for solving real-world problems, as they are inherently tolerant of imprecise, partially incorrect, incomplete, or uncertain data (Siemuri, Kuusniemi, et al. (2021)). ML performs well at handling time-series data by learning time-dependent patterns across multiple models, making it particularly useful for uncovering hidden or unknown information in GNSS datasets. The availability of vast amounts of data, affordable storage, and increasingly powerful, cost-effective processing has significantly driven the adoption of ML. ML can uncover complex dependencies in datasets where exploratory analysis fails to define the underlying model's structure. Instead of providing explicit formulas for data distribution, ML trains algorithms to identify relationships within data directly. This approach circumvents many assumptions inherent in traditional statistical methods. For example, Figure 12 illustrates an ML model trained to classify GNSS signals as line-of-sight (LOS), multipath, or non-line-of-sight (NLOS).

Supervised learning involves mapping inputs to known outputs using labeled training data, with prediction errors used to refine model accuracy, while semi-supervised learning applies when only some data are labeled or when only the statistical properties of the data are known, combining elements of supervised and unsupervised methods (Alabi (2021)).

Unsupervised learning is a branch of machine learning in which the training data are neither labeled nor categorized, and no explicit output is provided during the learning process (Alabi (2021)). Rather than relying on predefined targets, the algorithm identifies inherent structures within the data by analyzing patterns, correlations, and similarities. This often results in the grouping or clustering of input data based on similarity means, enabling the discovery of hidden relationships and data-driven insights.

Reinforcement Learning (RL) is neither purely supervised nor unsupervised, but rather falls into its own category of online learning. RL learns from rewards (which

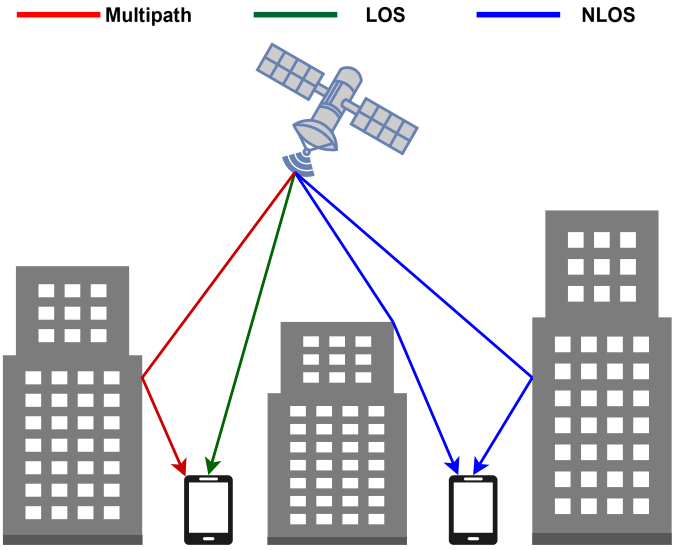


Figure 12. Schematic diagram of multipath, line-of-sight (LOS), and NLOS signals.

provide some supervision) but without explicit labeled data for every action. RL is a machine learning approach that enables interaction with the environment to learn online by taking actions and receiving feedback in the form of errors or rewards (Elmusrati, 2020). Actions are categorized as correct or incorrect based on the environment's response, making trial and error a fundamental aspect of this methodology (Alabi (2021)). The model autonomously determines the optimal behavior to maximize performance within a specific context, using a reinforcement signal as reward feedback to identify the most effective actions.

3.3.1 Tasks of Machine Learning

Machine learning (ML) encompasses a variety of tasks depending on the nature of the data, the problem domain, and the desired output. In the context of GNSS, ML techniques are increasingly employed to enhance positioning accuracy, signal integrity, and system robustness. The main tasks in machine learning include:

1. **Classification**: Predicting the category or class of input data. The output variables are used to classify the input variables into one of the possible output classes. Examples include spam email detection, medical diagnosis, and image recognition. In GNSS, classification tasks can be used to detect spoofing or jamming attacks, classify signal types (e.g., GPS vs. Galileo), or identify environmental conditions affecting signal quality (Heublein et al. (2025); L. Li, Elhajj, Feng, and Ochieng (2023a)).

2. **Regression**: Predicting continuous values or quantities. The objective of the regression task is to learn the observed input-output relations to find an accurate model. Examples include predicting house prices, stock prices, or temperature. In GNSS, regression is applied to estimate satellite clock errors, ionospheric delays, or precise receiver coordinates based on raw signal measurements (Jia, Yang, and Li (2024); Kiani (2020)).

3. **Clustering**: Grouping input data into clusters of similar items without predefined labels. Examples include customer segmentation, image segmentation, and anomaly detection. In GNSS, clustering can be used to group multipath-affected signals, segment geographic regions based on signal reliability, or identify patterns in signal degradation across urban environments (Gutierrez et al. (2024); Takahashi, Yano, and Kano (2025)).

4. **Dimensionality Reduction**: Reducing the number of features or dimensions in the data while preserving important information. Examples include PCA (Principal Component Analysis) or t-SNE for visualizing high-dimensional data. In GNSS, dimensionality reduction helps simplify complex sensor fusion inputs (e.g., GNSS + IMU + barometer) and extract dominant features for real-time positioning or signal integrity monitoring (Gite, Solankar, Surase, and Kale (2019); Jolliffe (1986); van der Maaten and Hinton (2008)).

5. **Anomaly Detection**: Identifying unusual or outlier data points that deviate significantly from the majority of the data. Applications include fraud detection, network security, and system health monitoring. In GNSS, anomaly detection is used to identify spoofed signals, sudden satellite malfunctions, or unexpected deviations in receiver behavior due to environmental interference (Savolainen (2022)).

6. **Reinforcement Learning**: Learning optimal actions through trial and error, where an agent interacts with the environment and receives feedback (rewards or penalties) based on actions. Examples include robot navigation, game playing (e.g., AlphaGo), and autonomous driving. In GNSS-integrated navigation systems, reinforcement learning can optimize satellite selection strategies, adaptively fuse GNSS and inertial data, or guide autonomous drones in GPS-denied environments (Dasgupta, Ghosh, and Rahman (2021a); Tang et al. (2024a)).

7. **Recommendation Systems**: Predicting what items a user might be interested in based on past behavior or preferences. Examples include movie recommendations, online shopping, and music suggestions. In GNSS, recommendation systems can assist in adaptive receiver configuration, suggesting optimal signal processing parameters or satellite subsets based on historical performance and environmental context (Mohanty and Gao (2024b); P. J. G. Teunissen and Montenbruck (2017)).

8. **Generative Models**: Learning to generate new data samples that resemble the original dataset. Examples include generating realistic images (e.g., GANs - Generative Adversarial Networks) or generating text. In GNSS, generative models can simulate realistic signal environments for training spoofing detection algorithms, augment datasets for urban canyon scenarios, or generate synthetic satellite trajectories for testing (Jwo, Biswal, and Mir (2023a); S. Wang et al. (2024)).

9. **Natural Language Processing (NLP)**: Analyzing and understanding human language data. Examples include text classification, sentiment analysis, language translation, and question answering. In GNSS, NLP can be used to interpret satellite system logs, analyze user feedback in crowd sourced positioning platforms, or support voice-based navigation interfaces (Manjunath, Heublein, Feigl, and Ott (2025)).

10. **Transfer Learning**: Applying knowledge gained from one task to improve learning in a related but different task. This is particularly useful when there is a lack of data in the target task. Examples include adapting image recognition models to medical imaging or speech recognition across languages. In GNSS, transfer learning can be used to adapt models trained in open-sky conditions to urban or indoor environments, or generalize spoofing detection models across different satellite constellations (M. Jiang, Ye, Xiao, and Gou (2024)).

These tasks are often combined or adapted to suit specific real-world applications, such as autonomous driving, robotics, healthcare, finance, and more.

3.3.2 Data division in machine learning

In supervised machine learning such as in regression and classification tasks, the data are usually divided into three types. These are **training**, **validation**, and **testing** sets. In certain cases, the data is split into the training and testing sets, respectively. The data division is a critical step in preparing data for training, validation, and testing of models. The process typically involves splitting the available data into different subsets to ensure that the model generalizes well to unseen data. The common data division strategies include:

- **Training Set**: The largest portion of the data is used to train the model. This data helps the model learn patterns and relationships between input features and target outcomes.

- **Validation Set**: A smaller portion of the data is used during training to tune hyper-parameters and validate the model's performance on data it hasn't seen

yet. It helps in preventing over-fitting by providing an unbiased evaluation of the model during the training process.

- **Test Set**: This portion of data is kept separate from the training process and is only used after the model has been trained. It is used to assess the final model's performance and generalization ability.
- **Cross-Validation**: In cases with limited data, k-fold cross-validation is used, where the data is divided into 'k' subsets. The model is trained on 'k-1' subsets and validated on the remaining one, iterating this process 'k' times, using each subset as a validation set once.
- **Holdout Method**: A simple approach where the data is split into training and test sets, typically with a ratio of 70-80% for training and 20-30% for testing.
- **Stratified Sampling**: This ensures that each class in the target variable is represented proportionally in both the training and test sets, which is especially important for imbalanced datasets.

These strategies help ensure the model is robust, avoids overfitting, and performs well when exposed to new data.

3.3.3 Machine learning performance metrics

We present a list of common machine learning performance metrics with the ML task they are usually used in.

1. **Accuracy (Classification)**: The proportion of correctly predicted instances over the total instances. It is most useful when the data is balanced but less effective in imbalanced datasets (Sokolova and Lapalme (2009)).

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

2. **Precision (Classification)**: The proportion of true positive predictions out of all positive predictions. Precision is particularly important when the cost of false positives is high (Powers and Ailab (2011)).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

3. **Recall/Sensitivity (Classification)**: The proportion of true positive predictions out of all actual positives. Recall is critical when false negatives are

costly.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

4. **F1 Score (Classification)**: The harmonic mean of precision and recall. It balances the two metrics, especially in cases of imbalanced data.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. **Area Under the Receiver Operating Characteristic Curve (AUC-ROC) (Classification)**: Measures the ability of a model to discriminate between classes. The ROC curve plots the true positive rate (recall) against the false positive rate (Fawcett (2006)).

6. **Confusion Matrix (Classification)**: A matrix that visualizes the performance of a classification model, showing the true positives, true negatives, false positives, and false negatives.

	Predicted Positive	**Predicted Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

7. **Mean Squared Error (MSE) (Regression)**: Commonly used in regression tasks, it calculates the average of the squared differences between predicted and actual values (Hyndman and Athanasopoulos (2021)).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

8. **Root Mean Squared Error (RMSE) (Regression)**: The square root of the mean squared error. RMSE gives an idea of the error magnitude in the original units (Hyndman and Athanasopoulos (2021)).

9. **Mean Absolute Error (MAE) (Regression)**: The average of the absolute errors between predicted and actual values. It is simpler than MSE but does not heavily penalize large errors.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

10. **R-squared (R^2) (Regression)**: A statistical measure that indicates how well the model's predictions approximate the real data. It is used to evaluate the

goodness of fit in regression models.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Where:

- R^2: R-squared, also known as the coefficient of determination, measures how well the regression model's predictions approximate the actual data points. It ranges from 0 to 1, where 1 indicates a perfect fit.
- y_i: The actual observed value for the i-th data point.
- $\hat{y}_i$: The predicted value for the i-th data point, as estimated by the regression model.
- $\bar{y}$: The mean (average) of all the actual observed values.
- $\sum(y_i - \hat{y}_i)^2$: The sum of the squared differences between the actual observed values and the predicted values. This term represents the residual sum of squares (RSS), which measures the amount of variance in the data that is not explained by the model.
- $\sum(y_i - \bar{y})^2$: The sum of the squared differences between the actual observed values and the mean of the observed values. This term represents the total sum of squares (TSS), which measures the total variance in the data.

11. **Adjusted Rand Index (ARI) (Clustering)**: The Adjusted Rand Index is a widely used metric for evaluating the similarity between two data clusterings, particularly in unsupervised learning tasks. Unlike the basic Rand Index, ARI adjusts for the possibility that some agreement between clusterings may occur purely by chance (Hubert and Arabie (1985)). The ARI ranges from -1 to 1, where:

    - **1** indicates perfect agreement between the two clusterings,
    - **0** suggests that the similarity is no better than random chance,
    - **-1** reflects complete disagreement.

    The ARI is computed using the following formula:

$$\text{ARI} = \frac{\text{RI} - \text{Expected RI}}{\text{Max RI} - \text{Expected RI}}$$

    Where:

    - RI **(Rand Index)**: Represents the proportion of all possible pairs of samples that are either assigned to the same cluster in both clusterings or assigned to different clusters in both. It quantifies the overall agreement between the two clustering assignments.

- $ExpectedRI$: Denotes the expected value of the Rand Index under a random model, assuming that the cluster assignments are made independently. This term corrects for the baseline similarity that could occur by chance alone.
- $MaxRI$: Refers to the theoretical maximum value the Rand Index could attain, given the number of clusters and data points. It serves as a normalization factor to scale the ARI between -1 and 1.

By incorporating these adjustments, the ARI provides a more reliable and interpretable measure of clustering performance, especially when comparing results across datasets with varying cluster sizes or distributions.

12. **Silhouette Score (Clustering)**: A measure of how similar an object is to its own cluster compared to other clusters. A high silhouette score indicates that the points are well clustered (Rousseeuw (1987)).

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where $a(i)$ is the average distance from the i-th point to other points in the same cluster, and $b(i)$ is the minimum average distance to points in a different cluster.

13. **Davies-Bouldin Index (Clustering)**: Measures the average similarity ratio of each cluster with the cluster that is most similar to it. A lower score indicates better clustering (Davies and Bouldin (1979)).

14. **Dunn Index (Clustering)**: Evaluates clustering by comparing the ratio of the minimum inter-cluster distance to the maximum intra-cluster distance. Higher values indicate better clustering.

15. **Homogeneity, Completeness, and V-Measure (Clustering)**: Measures used to evaluate the quality of clustering:

    - **Homogeneity**: Measures if each cluster contains only members of a single class.
    - **Completeness**: Measures if all members of a given class are assigned to the same cluster.
    - **V-Measure**: A balanced score that combines both homogeneity and completeness.

Each of these metrics can provide unique insights into how well a machine learning model is performing depending on the problem at hand.

3.3.4 Over-fitting and under-fitting in machine learning methodology

In machine learning, over-fitting and under-fitting are two common issues that affect the a model's performance:

Over-fitting
Overfitting refers to a modeling error that arises when a ML algorithm captures not only the true underlying patterns in the training data but also the random noise and irrelevant fluctuations. This results in a model that is overly complex compared to the actual mapping in the data and tailored to the training set, rather than to the general data distribution. Although such a model may exhibit excellent performance on the training data, it typically fails to generalize to new, unseen data. A common symptom of overfitting is a significant discrepancy between training and test performance—manifested as low error on the training set but high error on the validation set.

Causes of over-fitting:

- Model is too complex relative to the actual mapping complexity of data.
- Many irrelevant features are included.
- Lack of proper regularization or constraints on the model.

Mitigation strategies:

- Reducing the model complexity (e.g., using fewer parameters).
- Using techniques such as cross-validation.
- Regularization methods like L1 (Lasso) or L2 (Ridge) regularization.
- Pruning decision trees or using early stopping in training.

Under-fitting
Under-fitting happens when the model is too simple compared to the actual mapping in the data and fails to capture the underlying patterns in the training data. This leads to both poor training and test performance because the model cannot learn the necessary details to make accurate predictions.

Causes of under-fitting:

- Model complexity is much lower than actual mapping or pattern complexity.

- Insufficient training time or inappropriate model selection.
- The lack of important features in the dataset.
- Not enough data.

Mitigation strategies:

- Using a more complex model with more features.
- Increasing training time or improving the quality of features.
- Trying more sophisticated algorithms.
- Increase the data points.

Balancing between over-fitting and under-fitting is crucial for building a good machine learning model. Techniques such as cross-validation, regularization, and feature selection are commonly used to address these issues.

3.4 Machine learning utilization in GNSS

3.4.1 Machine learning algorithms

Machine learning algorithm is a mathematical-based model that automatically learns from data without the need to explicitly program the algorithm. ML algorithms are the building blocks for learning from data and making predictions or decisions. When implementing machine learning algorithms, the data serve as the fundamental components (Alabi (2021)). These data are characterized by various features, each containing different types of values. Therefore, it is crucial to preprocess the data to address potential issues such as noise, outliers, missing values, hidden redundancy, and duplicates to enhance data quality. Additionally, understanding the dependencies and relationships among the features can help in selecting the most relevant features during the training of the ML model (Alabi (2021)).

The main types of machine learning algorithms include:

1. **Linear Regression**: A statistical method used for predicting a continuous target variable based on one or more input features. Commonly used in predicting house prices, stock prices, etc. In GNSS, it can be used to model satellite

clock drift or predict ionospheric delay based on temporal and spatial features (Mohanty and Gao (2024a)).

2. **Logistic Regression**: A classification algorithm used to predict the probability of a binary outcome (0 or 1). It is commonly used in tasks like spam detection and disease prediction. In GNSS, it can be applied to distinguishing between valid and spoofed signals (Liu, Lo, and Walter (2021); Shamohammadi, Pahlavani, and Sharifi (2023)).

3. **Decision Trees**: A supervised learning algorithm that creates a tree-like model of decisions and their possible consequences. It is used for both classification and regression tasks. In GNSS, decision trees can be used to select optimal satellite subsets based on signal quality metrics (L. Wang, Jiang, Ji, and Wei (2024)).

4. **Random Forests**: An ensemble learning method that combines multiple decision trees to improve accuracy and reduce overfitting. It is widely used for classification and regression problems. In GNSS, random forests can be used to predict positioning errors or detect anomalies in satellite signal behavior (L. Li, Elhajj, Feng, and Ochieng (2023b); Qi, Xu, Wang, and Hsu (2024)).

5. **Support Vector Machines (SVM)**: A supervised learning algorithm used for classification and regression tasks, which works by finding the hyperplane that best separates data into different classes. In GNSS, SVMs can be used to classify urban vs. open-sky environments based on signal features (Kondaveeti, Ratnam, and Sivavaraprasad (2022); Mohanty and Gao (2024a)).

6. **K-Nearest Neighbors (KNN)**: A non-parametric algorithm used for classification and regression tasks that makes predictions based on the majority class of the nearest neighbors to a given point. In GNSS, KNN can be used to estimate user location by comparing signal patterns to a database of known positions (M. Li, Huang, Wang, and Xie (2022); Shamohammadi et al. (2023)).

7. **Naive Bayes**: A probabilistic classification algorithm based on Bayes' theorem that assumes independence between the features. It is simple and effective for text classification tasks. In GNSS, Naive Bayes can be used for signal fault detection or predicting satellite visibility based on environmental conditions (Jwo, Biswal, and Mir (2023b); Mohanty and Gao (2024a)).

8. **K-Means Clustering**: An unsupervised learning algorithm used for clustering tasks, which divides data into K clusters by minimizing the variance within each cluster. In GNSS, it can be used to group multipath-affected signals or cluster geographic regions based on signal degradation patterns (H. Wang, Pan, Gao, Xia, and Ma (2022).).

9. **Principal Component Analysis (PCA)**: An unsupervised algorithm used for dimensionality reduction by transforming the features of the data into a

smaller set of uncorrelated variables called principal components. In GNSS, PCA helps reduce high-dimensional sensor fusion data (e.g., GNSS + IMU) for efficient processing (D. Zhu et al. (2023)).

10. **Neural Networks (NN)**: A class of algorithms inspired by the human brain, consisting of layers of nodes (neurons). Used for tasks such as image recognition, speech recognition, and more complex problems. In GNSS, NNs can be used to model satellite orbit prediction or estimate user position from raw signal features (Jwo et al. (2023b); Kanhere, Gupta, Shetty, and Gao (2022)).

11. **Deep Learning (DL)**: A subset of neural networks with multiple layers that can learn hierarchical features. It is used for complex tasks like image and speech recognition, natural language processing, and game playing. In GNSS, DL models can detect spoofing attacks, predict signal outages, or enhance positioning in urban environments (Kanhere et al. (2022); Klos, Bogusz, Bos, and Gruszczynska (2019)).

12. **Gradient Boosting Machines (GBM)**: An ensemble learning method that builds a series of weak learners (typically decision trees) sequentially to improve accuracy. Popular implementations include XGBoost, LightGBM, and CatBoost. In GNSS, GBMs can be used to predict satellite position errors or classify signal anomalies (Ozarpaci et al. (2023); Shamohammadi et al. (2023)).

13. **Reinforcement Learning (RL)**: A learning paradigm where agents learn to make decisions by interacting with an environment and receiving rewards or penalties. Examples include Q-learning, Deep Q Networks (DQN), and policy gradient methods. In GNSS-integrated navigation, RL can optimize sensor fusion strategies or guide autonomous vehicles in GPS-denied areas (Dasgupta, Ghosh, and Rahman (2021b); Tang et al. (2024b)).

14. **Association Rule Learning**: A method for discovering interesting relationships between variables in large datasets, often used in market basket analysis. The Apriori algorithm is a well-known example. In GNSS, it can uncover correlations between satellite visibility, environmental conditions, and positioning accuracy (Zheng, Xie, and Ma (2009)).

15. **Hidden Markov Models (HMM)**: A statistical model used for time-series data, where the system is assumed to be a Markov process with unobserved (hidden) states. It is used in speech recognition, bioinformatics, and more. In GNSS, HMMs can model user movement patterns or predict satellite signal transitions over time (Newson and Krumm (2009)9.

16. **Generative AI (GenAI)**: A type of artificial intelligence that creates new content, such as text, images, music, or code, based on patterns learned from its training data. In GNSS, GenAI can generate synthetic signal environments

for training spoofing detection models or simulate satellite trajectories for testing (Mohanty and Gao (2024a); Tucker (2024)).

These algorithms represent the core of machine learning and are applied in various domains including finance, healthcare, marketing, robotics, and GNSS localization and position estimation.

3.4.2 Applications of machine learning in GNSS

In the study of the utilization of ML in GNSS, several machine learning algorithms were compared based on the GNSS use case. The application of machine learning (ML) in the context of GNSS has garnered significant interest across various industries. For example, Google's DeepMind AI has demonstrated the ability to navigate cities without a map, showcasing the potential for artificial intelligence (AI) to recognize objects, classify them, and relate them to the physical environment at different scales and distances (Internet of business (2022)). Google has also innovated map updates by integrating deep learning with Street View, combining location data from a Street View car's GNSS with address information and business names extracted from imagery. This approach enables efficient mapping of entire cities without prior knowledge of their layouts or nomenclature (Manoj Joshi (2022))

In another notable development, Apple applied to the Federal Communications Commission in February 2020 for a license to install GPS testing equipment at its headquarters, potentially linked to the company's August 2019 patent application for "Machine Learning Assisted Satellite Based Positioning" (Tracy Cozzens (2020)).

Given the growing interest from both industry and academia in leveraging ML for GNSS, we conducted a systematic literature review (SLR) to analyze and compare ML algorithms, methods, and solutions used in the field. This review aims to facilitate the development of innovative and efficient ML-based GNSS solutions. Covering studies published between 2000 and 2021, the review identifies and cites relevant examples to provide a comprehensive analysis of the literature (section 3.4.3).

3.4.3 Systematic literature review of ML utilization in GNSS

The planning, conducting, and reporting of the SLR performed in Publications I. and IV., used a process adopted from (Kitchenham (2004)), as illustrated in Figure 13.

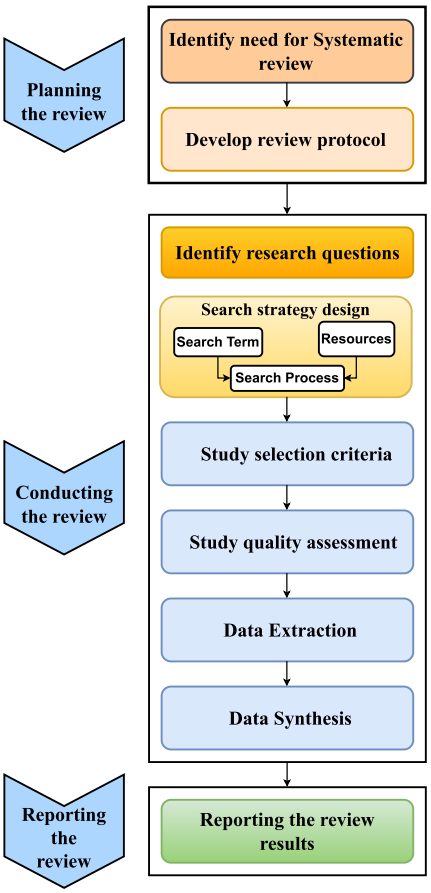


Figure 13. Systematic review process.

The review protocol was developed through frequent meetings and consultations with senior researchers and professors. This review protocol helps in reducing the possibility and risk of research bias in the SLR. Furthermore, we developed comprehensive search queries by incorporating alternative terms and synonyms with the Boolean operator 'OR' and by combining primary search terms using 'AND'. The following search terms were utilized to identify relevant literature:

GNSS AND "deep learning" OR GNSS AND "machine learning" OR GNSS AND "artificial intelligence" OR GNSS AND "random forest" OR GNSS AND "decision tree" OR GNSS AND "support vector machine" OR GNSS AND "neural network" OR GNSS AND "regression"

After identifying the search terms, relevant digital portals were selected. The following electronic databases were used for the search. IEEE Xplore, Google Scholar, ScienceDirect, Crossref, Scopus, Institute of Navigation (ION).

The flowchart for the search process is presented in Figure 14.

We formed a quality questionnaire for assessing the relevance and strength of the

relevant studies. The quality criteria were developed by considering the suggestions given in (Malhotra (2015)). Table 7 presents the quality assessment questions. The questions are ranked 1 (yes), 0.5 (partly) and 0 (no). The final score is obtained after adding the values assigned to each question. A study could have a maximum score of 6 and a minimum score of 0.

Table 7. Quality Assessment Questions.

#Q	Quality factors	Yes	Partly	No
Q1	Are the ML algorithm used in the research clearly described?			
Q2	Are the GNSS use cases to which the ML algorithms are utilized clearly defined?			
Q3	Are the data set for model construction clearly defined and are the evaluation or validation methods for the ML algorithms mentioned?			
Q4a	Are there any comparison done between non-ML vs ML?			
Q4b	Are there any comparison done between ML vs other ML?			
Q5	Are the strengths and weaknesses of the ML algorithms specified?			

The SLR study identified 213 primary studies that applied ML in GNSS use cases. These papers were published between the period 2000–2021. A total of **122 (57.55%)** papers were published in journals, **87 (41.04%)** papers were published in conference proceedings, and **3 (1.42%)** monograph **(1 Ph.D. thesis, 1 M.Sc thesis, and 1 book chapter** (Table 8).

Table 8. Publication Type Distribution.

Paper Type	Number of Studies	Percentage
Journal	123	57.75
Conference	87	40.85
PhD thesis	1	0.47
Msc thesis	1	0.47
Book-Chapter	1	0.47
Total	**213**	**100**

This SLR showed that ML has been utilized in GNSS studies as listed below. This is, however, not an exhaustive list, as new research topics and areas in the application of ML in GNSS are continually evolving.

- GNSS Signal Acquisition

- Signal Detection and Classification
- Earth Observation and Monitoring
- GNSS Navigation and Precise Positioning
- GNSS Denied Environments and Indoor Navigation
- GNSS Anomaly Detection and Atmospheric Effects
- GNSS Security: Spoofing and Jammer Attacks
- Sensor Fusion (e.g., GNSS/INS Integration)
- Satellite Selection based on Geometric Dilution of Precision (GDOP)
- LEO Satellites: Orbit Determination and Positioning

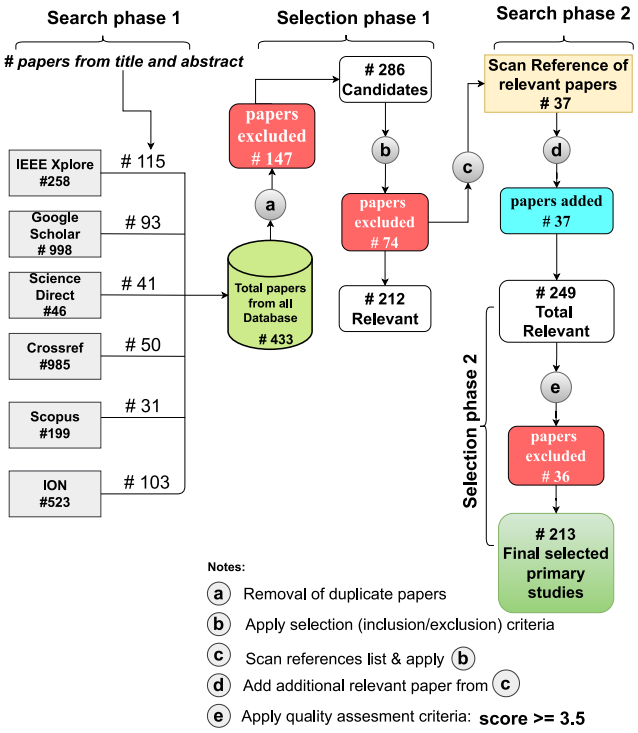


Figure 14. Stages in the systematic review process.

3.5 GNSS datasets used by ML models

A variety of data sets have been used in ML utilization in GNSS studies. Some of the data used were simulated data (Yiming (2017), Munin, Blais, and Couellan

(2020), Louis (2019)) while others were real data (Brum et al. (2019), I, Ratnam, Raman, and Sivavaraprasad (2020), Orus Perez (2019)), and publications II., III., V., and VI.. The utilized data sets can be publicly available (for free) or private in nature (not shared by researchers), in such cases, the results cannot be verified and such studies are not replicable. The major category of data sets used in ML utilization in GNSS are as follows:

- Simulated data - These data sets include, CU SeNSe Lab (Liu, Morton, and Jiao (2018)), and a customized Software Defined Radio (SDR)-based GNSS data grabber and software receiver (Linty, Farasin, Favenza, and Dovis (2019)).
- Real data - These datasets include, GNSS raw data from smartphones (it also include data collected by individuals or research organizations and companies) (Hsu (2017)), Vertical Total Electron Content (VTEC) data from the National Oceanic and Atmosphere Administration (NOAA) (Brum et al. (2019)), and GPS data over International GNSS Service (IGS) stations (I et al. (2020), Orus Perez (2019)).
- Combining Real and Simulated data - The combination of real and simulated data are used (Quan, Lau, Roberts, Meng, and Zhang (2018); Semanjski, Muls, Semanjski, and De Wilde (2019); Yiming (2017)).

Different kinds of devices were used for the collection of the data used for research. These include low-cost receivers such as u-blox, smartphones, and high-end receivers, for example, Trimble, Novatel, and Javad Triumph VS receivers. The type of research determined the location for data collection. These locations include open sky environments (LOS), urban canyon (multipath/NLOS prone), indoors (GNSS denied environment), and Laboratory generated data.

In GNSS, even with the same equipment, and the same data collection path, the data collected at different times should be regarded as different dataset due to the change in satellite's geometry. Therefore, in the GNSS field, it is rare to see researchers use the same dataset for different objectives or even more, the same objective. However, in research, replicability is important, therefore, it would be good practice to create a database where researches can store their research data for easy access in order for other researchers to be able to replicate their results. This way, the ML model used can be trained and evaluated by other researchers making use of the shared data.

In our research (publications II., III., and VI.), we have mainly used the dataset provided by Google as part of the Google Smartphone Decimeter Challenge (GSDC) in the ION GNSS+ conferences from 2021 - 2024 (ion.org).

The Google Smartphone Decimeter Challenge (GSDC), launched by the Android GPS team at Google, focuses on improving smartphone positioning accuracy, par-

ticularly in urban areas where GNSS signals are often obstructed. The challenge encourages the development of machine learning models to enhance GPS data resolution to decimeter or even centimeter-level accuracy, enabling advanced navigation features such as lane-level precision for carpool lane ETAs. Initially introduced during the Institute of Navigation (ION) GNSS conference in 2021, the competition has continued through subsequent editions in 2022, 2023, and 2024. It aims to foster research in smartphone GNSS accuracy and utilize navigation data to improve smartphone localization performance. The ultimate objective of the GSDC is to advance navigation techniques and mobile internet applications by delivering more precise geospatial information (Google (2021, 2022, 2023)).

In publication V., we used an Android smartphone with an application capable of capturing raw RINEX observables. The smartphone used was the Samsung Galaxy Note20 Ultra. The software used was the GEO++ RINEX Logger.

3.6 Precise positioning with low-cost GNSS devices

There have been an increase in the study of the application of ML to low-cost GNSS devices to enhance positioning accuracy and navigation performance, particularly by mitigating errors caused by multi-path effects, signal interference, and other environmental challenges.

Some examples of low-cost GNSS devices are listed below:

- **u-blox receivers:** u-blox receivers are popular GNSS module. u-blox NEO-M8N supports GPS, GLONASS, and QZSS, commonly used in hobbyist and IoT applications. C94-M8P by u-blox support Real-Time Kinematic (RTK) positioning. While, ZED-F9P by u-blox is a multi-band GNSS receiver offering centimeter-level accuracy, suitable for cost-effective surveying and robotics applications.
- **SkyTraq S2525F8-BD-RTK:** An affordable GNSS module capable of providing sub-meter accuracy with RTK.
- **Piksi Multi by Swift Navigation:** A compact multi-band RTK GNSS receiver designed for low-cost precision navigation.
- **Adafruit Ultimate GPS:** An easy-to-use GPS breakout module ideal for low-cost DIY projects.
- **GNSS Shield for Arduino:** A versatile module for integration with microcontroller platforms, providing basic GNSS functionality.

- **Smartphones:** Smartphones are enabled with positioning and localization devices like GNSS, Inertia measurement unit (IMU), Wi-Fi, and recently Ultra-wide band (UWB), providing basic positioning and localization functionality.

Publication II. investigates the use of machine learning (ML) techniques to enhance the accuracy and reliability of GNSS positioning on smartphones. The study addresses challenges such as signal degradation, multipath effects, and interference in urban environments, which often hinder GNSS performance on mobile devices.

Main contributions of the study include:

- **ML Integration:** Proposing machine learning-based methods to process GNSS data and correct errors caused by adverse conditions.
- **Practical Testing:** Demonstrating the effectiveness of the proposed methods through real-world testing scenarios using smartphone GNSS data.
- **Results:** Highlighting significant improvements in positioning accuracy and navigation performance compared to traditional GNSS approaches.

This work underscores the potential of ML in overcoming GNSS limitations on smartphones, paving the way for more precise location-based applications in consumer devices.

Datasets:
The dataset used was obtained from a Google Smartphone Decimeter Challenge organized in 2021 (Google (2021)). This included, GNSS raw measurements collected from commercially available **smartphone GNSS receivers**, and **Reference positioning data** obtained using high-precision GNSS equipment to establish ground truth. The data comprised of urban and suburban datasets to evaluate the model's performance in diverse environmental conditions (e.g., open-sky and multipath-rich areas) (Orendorff et al. (2021)).

The results show an improvement in the GNSS positioning and navigation accuracy when evaluated against the ground truth of the data collected using high end GNSS receivers for the same locations. The positioning accuracy obtained is in the range of 5.620 - 5.743 m (meters). This is an improvement to the WLS of 20.49 meters positioning accuracy.

The training datasets included GnssLogger files, RINEX observation files, ground truth files, and GNSS and IMU data files from the smartphone device. The test datasets contained the same types of data and adhered to the same format as the training datasets, except that the ground truth data were not included. The results

obtained from the test datasets were intended to predict the expected ground truth and were then evaluated against the actual custodial ground truth.

Methods:
For the BR and LR approaches, one-hot encoding was applied to prepare the categorical data. For the NN approach, raw features were used without one-hot encoding. Each model was trained and evaluated separately. Preliminary tests implementing multiple regression algorithms for performance analysis showed that BR, LR, and NN were the most promising methods.

Publication II. applies Machine Learning (ML) techniques to enhance GNSS positioning and navigation accuracy on smartphones. A supervised learning approach is used to model the GNSS error sources and mitigate their impact on positioning accuracy. Outlier detection and removal methods are employed to process GNSS data. Algorithms such as Linear Regression (LR), Bayesian Ridge (BR) and Neural Network (NN) are explored for error modeling and correction. A simple weighted average (SWA) which combines all the previous three ML technique was also implemented. The results showed improvement in the position accuracy with the simple weighted average (SWA) method having the best accuracy followed by Bayesian Ridge (BR), Linear Regression (LR), and then Neural Network (NN) respectively (Siemuri, Selvan, et al. (2021)).

Figure 15 illustrates the machine learning implementation pipeline, where GNSS data is processed using the BR, LR, and NN models, and their outputs are combined through a weighted average (SWA) to generate the final positioning result.

Bayesian Ridge (BR) and Logistic Regression (LR) Models

Both BR and LR models were trained using one-hot encoded data. The one-hot encoding was applied to the ”phoneName” variable (name of the phone used to collect the data) of the train and test data. The input features consisted of latitude and longitude columns, while the output was the predicted position coordinates. The dataset was split into training and validation subsets using the `scikit-learn` model selection library.

Bayesian Ridge Regression: Bayesian Ridge Regression introduces a probabilistic framework for linear regression. The model assumes:

$$y = Xw + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \alpha^{-1} I)$$

where X is the feature matrix, w are the regression coefficients, and α is the precision of the noise. A Gaussian prior is placed on w:

$$w \sim \mathcal{N}(0, \lambda^{-1} I)$$

with λ controlling the regularization. Posterior inference yields estimates of w that balance data fit and regularization.

Logistic Regression: Logistic Regression was adapted for classification of categorical features related to positioning. The probability of belonging to class c is modeled as:

$$P(y = c \mid x) = \frac{\exp(w_c^T x)}{\sum_j \exp(w_j^T x)}$$

$$P(y = 1 \mid x) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}$$

where w_c are the learned weights for class c. One-hot encoding was reversed (decoded) after training to recover phone-specific predictions.

Validation data was used to evaluate the models, with scores reported as 50th, 95th, and average percentiles for each phone. The models were then applied to the complete test dataset to predict latitude and longitude coordinates. Evaluation was performed on Kaggle as part of the Google Smartphone Decimeter competition requirements. These models were adapted from (Ramswaroop (2021)).

Neural Network (NN) Model

The NN model was implemented using the `scikit-learn` library, adapted from (J.-Y. Lee (2021)). Unlike BR and LR, one-hot encoding was not applied. Instead, aggregated features were generated for training.

Feature Creation The features included:

- Previous latitude and longitude values
- Corrected pseudoranges (correctedPrM) for each satellite, derived from Google's shared files
- Differences between training data coordinates and ground truth coordinates

These features were generated for both the train and test datasets.

Training the NN Model The NN was trained using Keras with skip connections on Tensor Processing Units (TPUs). TPUs are optimized for large batch sizes and convolutional neural networks, providing high training throughput. Binary cross-entropy loss was used, and accuracy was tracked during training, validation, and testing. The NN architecture summary is itemized below.

- Number of layers: 13 (including input, dense, batch normalization, dropout, add, and output layers)
- Total parameters: 89,730
- Trainable parameters: 88,962
- Non-trainable parameters: 768

Weighted Average of BR, LR, and NN Models

To improve performance, the results from the three ML algorithms were integrated using a Simple Weighted Average (SWA). Initially, equal weights were assigned to each model:

$$w_{BR} = w_{LR} = w_{NN} = \frac{1}{3}$$

This yielded slight improvements compared to individual models. Subsequently, weights were adjusted based on model accuracy:

$$w_i = \frac{Acc_i}{\sum_j Acc_j}, \quad \sum_i w_i = 1$$

where Acc_i is the accuracy of model i. This adaptive weighting improved results further, as models with higher accuracy contributed more strongly to the final prediction.

Results

In publication II., the evaluation criteria for the GSDC 2021 were as follows: ”Submissions are evaluated based on the mean of the 50th and 95th percentile distance errors. For each phone, the horizontal distance (in meters) between the predicted and ground truth latitude/longitude is computed once per second. These distance errors are used to generate a distribution from which the 50th and 95th percentile errors are derived (i.e., the 95th percentile error is the value, in meters, below which

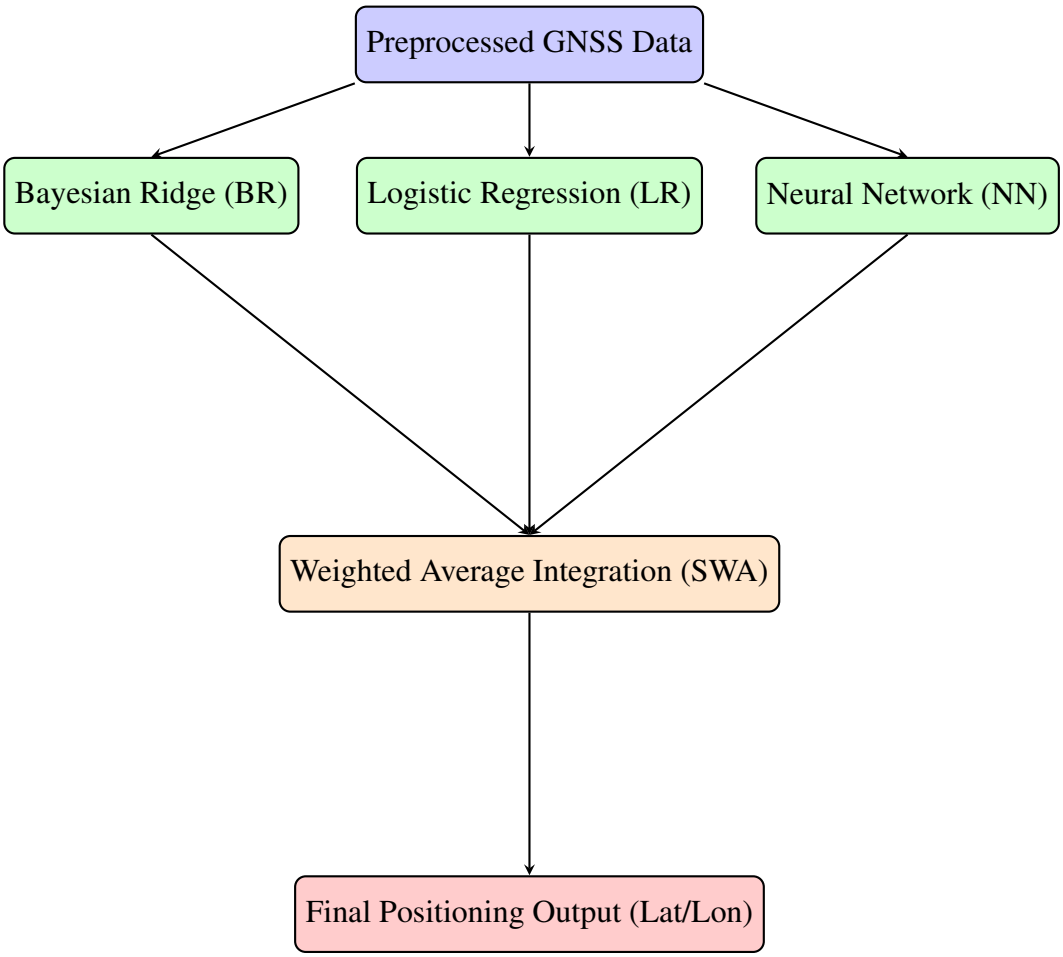


Figure 15. Flowchart of ML implementation pipeline. GNSS data is processed by BR, LR, and NN models. Their outputs are integrated using a weighted average (SWA), producing the final positioning output.

95% of the distance errors fall). The 50th and 95th percentile errors are then averaged for each phone. Finally, the mean of these averaged values is calculated across all phones in the test set."

In Table 9, the results of the preliminary evaluation done on kaggle.com for the test data baseline prediction is presented for each ML model implemented (BR, LR, NN , SWA method).

From table 9, we can see that the SWA performs better, followed by the BR, LR and NN model respectively. In the case of the SWA, when the average accuracy of a certain ML technique increases, its summation weight would increase, thereby, increasing the accuracy. This is seen in the accuracy improvement when different weights are used compared to using the same weights.

Table 9. Results from evaluation of test data prediction on kaggle.com.

Model List						
					SWA	
	WLS	Phone mean + BR	Phone mean + LR	Phone mean + NN	same weights	different weights
Accuracy in meters (m)	20.49	5.631	5.656	5.743	5.626	5.620

3.7 Machine learning and sensor fusion

Sensor fusion is a computational procedure to combine the measurements from multiple sources such that the output information after fusion is maximized (Elsanhoury et al. (2022)).

With the advancement of sensor technology and the inherent errors associated with individual sensors, multi-sensor fusion has become increasingly popular. Leveraging multiple measurement devices in positioning applications helps reduce uncertainties, leading to enhanced reliability and accuracy compared to relying on a single sensor (X. Guo et al. (2020b)). Several tracking systems, such as UWB, inertial navigation systems (INS)/IMU, dead reckoning (DR), visual map matching (VMM), and computer vision, can be integrated with a GNSS system to provide more precise and dependable estimations. The optimal positioning outcomes obtained by combining various positioning methods adhere to a unified framework, as shown in Figure 16.

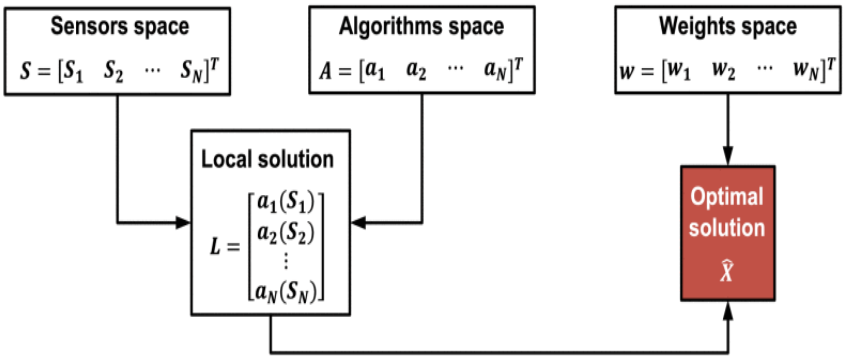


Figure 16. Fusion-based positioning framework.

The positioning scene determine the sensor fusion strategy employed. The positioning may be required outdoor, in that case an integration of GNSS/INS is appropriate. For the indoor applications, integrations schemes like GNSS/Wi-Fi, GNSS/Bluetooth, and GNSS/UWB have been studied (Elsanhoury et al. (2022)). The use of sensor fusion for seamless navigation between indoor and outdoor is important especially in the area of intelligent transportation systems (ITSs). A tightly coupled sensor-fused GPS/UWB/INS algorithm can be used to address this need (J. Wang, Gao, Li, Meng, and Hancock (2016)). This approach was enhanced by an error detection unit and a robust Kalman filter. The UWB component serves to refine noisy GNSS measurements. The results demonstrated that integrating UWB with GPS/INS significantly improved vehicle positioning accuracy, achieving submeter level precision, even with added pseudo-range gross errors across different scenarios (the maximum positioning error was 0.78 m), thereby enhancing system reliability.

3.7.1 Precise outdoor positioning with low-cost GNSS devices

In publication III., we explore the application of machine learning (ML) techniques to enhance GNSS/IMU integration for achieving high-precision positioning on smartphones. The study evaluates the performance of various ML algorithms in fusing GNSS and IMU data, particularly under challenging environments such as urban canyons, where GNSS signals are often obstructed or degraded.

Main contributions of the study include:

- Demonstrated the effectiveness of ML-driven GNSS/IMU integration in improving positioning accuracy and robustness on smartphones.
- Proposed a comparative analysis of different ML algorithms for sensor fusion, identifying their strengths and weaknesses in real-world scenarios.
- Showcased the potential of ML to address the limitations of traditional GNSS/IMU integration methods, paving the way for enhanced smartphone-based navigation systems.

Datasets used:
The dataset used in publication III. includes GNSS data collected from smartphones equipped with multi-constellation, dual-frequency GNSS receivers and IMU sensors. The dataset is provided by Google as part of the GSDC competition for 2022 and includes reference data from GNSS survey-grade receivers for validation. The raw GNSS data from smartphones, which have lower signal levels and higher noise compared to commercial GNSS receivers, were processed and used to train and validate the machine learning model (Howard et al. (2022)). The final performance of

the method was evaluated using the GSDC competition's test data and achieved a public score of 2.61 meters and a private score of 2.29 meters.

The driving paths of the GSDC provided data can be classified into three main categories: highways, tree-lined streets, or downtown areas and were typically collected in the areas shown in the map in Figure 17.

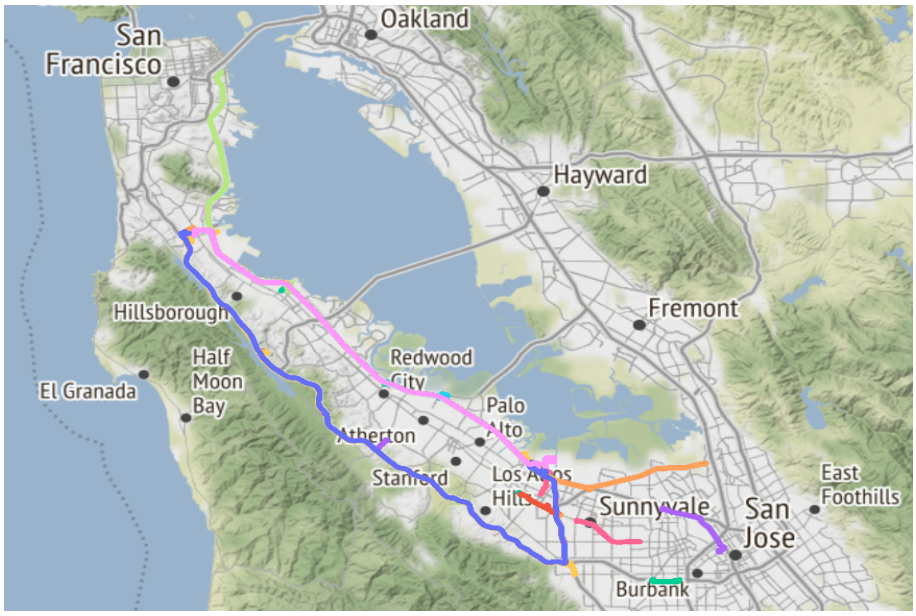


Figure 17. Driving trajectories for collected data provided for GSDC 2022.

The training datasets included GnssLogger files, RINEX observation files, ground truth files, and GNSS and IMU data files from the smartphone device. The test datasets contained the same types of data and adhered to the same format as the training datasets, except that the ground truth data were not included. The results obtained from the test datasets were intended to predict the expected ground truth and were then evaluated against the actual custodial ground truth.

Methods:
In publication III., we implemented a machine learning (ML)-based adaptive positioning approach for smartphone positioning in the Google Smartphone Decimeter Challenge (GSDC) 2022. The approach uses multi-sensor fusion by integrating GNSS data and Inertial Measurement Unit (IMU) sensor measurements. Our developed method (publication II.) termed loosely coupled-GNSS/IMU/ML (LC-GNSS/IMU/ML), integrates a ML-based adaptive positioning (MAP) algorithm with RTK post-processing for enhanced positioning estimation (MAP-PPK). Specifically, the method involves:

- **ML Model - kNN algorithm:** The kNN algorithm is utilized to predict the driving paths (e.g., highways, tree-lined streets, downtown areas), enabling

adaptation of the positioning technique. This was a classification problem.

- **Post-Processed Kinematic (PPK) Precise Positioning:** GNSS data from multi-constellation, dual-frequency receivers are processed using the PPK technique. The carrier phase measurements are used in conjunction with differential corrections from known GNSS base stations.
- **Rauch–Tung–Striebel (RTS) Smoother:** The RTS smoother combines a forward pass Kalman Filter (KF) and backward recursion smoother to integrate GNSS and IMU measurements for improved smartphone positioning accuracy.

Machine Learning Model for Environment Classification

To adapt the positioning technique to different driving environments, a machine learning model was employed to classify GNSS data traces into distinct categories. The chosen model was the *k-Nearest Neighbors (kNN)* algorithm, which is a non-parametric classifier that assigns a label to a new observation based on the majority class among its k closest neighbors in the feature space.

Data Preparation

The dataset consisted of GNSS measurements collected from 29 smartphones over multiple days. Each phone's data was associated with a specific driving environment, namely:

- Highway
- Tree-lined streets
- Downtown areas

The raw GNSS data was organized such that each phone's measurements could be grouped and labeled according to the environment in which they were collected.

The kNN model was trained to classify new data from a new phone into one of the groups or cate gories (highway, tree-line, or downtown). Afterwards, a RTKLIP PPK confgurations was used to process the phone's data based on the classifcation results. For training purposes, categorical labels were encoded numerically:

$$\text{Highway} = 0, \quad \text{Tree-line} = 1, \quad \text{Downtown} = 2$$

This encoding allowed the kNN classifier to operate on structured input data.

In the MAP-PPK method, the K-nearest neighbor (kNN) was employed in the MAP algorithm to predict the area in which smartphone data was collected. This is a classification problem, where kNN was used to categorize the data based on its collection location. The area was divided into three categories: highway, tree-line, and downtown, based on the environmental characteristics. Data from phones, identified by numbers 1 to 29, represented GNSS data collected on specific days. These data were grouped based on the area of collection (downtown, highway, or tree-line). The dataset was organized so that the phones could be classified and labeled according to their data collection location. Each phone was assigned to one of the three categories, with labels encoded for training (0 = highway, 1 = tree-line, 2 = downtown). The kNN model was then trained to classify new data from an unseen phone into one of the predefined categories. After classification, RTKLIB PPK configurations were used to process the phone's data based on the classification results.

Model Training

The kNN algorithm was trained on the labeled dataset. For each new GNSS trace, the algorithm computed the Euclidean distance in feature space to all training samples and identified the k nearest neighbors. The class label was then assigned based on majority voting among these neighbors:

$$\hat{y} = \arg \max_{c \in \{1,\ldots,C\}} \sum_{x_i \in \mathcal{N}_k(x)} \mathbb{1}(y_i = c)$$

where $\hat{y}$ is the predicted class, C is the total number of possible classes, $\mathcal{N}_k(x)$ is the set of the nearest neighbors of sample, y_i are the labels of the k nearest neighbors, and $\mathbb{1}(\cdot)$ is the indicator function (equals 1 if neighbor i belongs to class, otherwise 0).

Application to Positioning

Once the environment classification was obtained, the GNSS data from the corresponding phone was processed using an RTKLIB Post-Processed Kinematic (PPK) configuration tailored to that environment. For example:

- Highway traces were processed with parameters optimized for open-sky conditions.
- Tree-lined street traces used configurations robust to partial satellite blockage.
- Downtown traces employed settings designed to mitigate multipath and non-line-of-sight errors.

This adaptive processing pipeline ensured that the positioning algorithm was tuned

to the specific challenges of each environment, thereby improving accuracy.

Initially, the RTK-GNSS technique, which is widely used for high-precision positioning (Takasu and Yasuda (2009), Tim (2021)), was considered for implementation. However, it was found to be difficult to apply on smartphones due to the limitations of their antennas, which prevented them from resolving the integer ambiguities in the carrier phase measurements. As a result, a more robust position estimation method was developed, utilizing post-processed kinematic (PPK) data derived from smartphone GNSS observations (Takasu and Yasuda (2009)) and our ML-based adaptive positioning (MAP) algorithm. In the PPK technique, the carrier phase is used to compute the user's position by applying differential corrections based on the user's position detected by the MAP algorithm. In other words, the GNSS carrier-phase information was post-processed using the MAP-PPK method.

The integration of the GNSS results and IMU data was performed using loosely coupled integration with Kalman filter (KF) algorithm and Rauch–Tung–Striebel (RTS) smooter. We use the acceleration measured by IMU for inertial navigation calculation to get the position. The GNSS/IMU result is used as a prior observation of KF while the covariance matrix and state transition matrix gained from forward Kalman filtering process is used for the reverse RTS smoothening to the state estimate. The RTS smoother, consists of a forward pass KF and a backward recursion smoother to achieve the loosely coupled integration of GNSS and IMU measurements for positioning estimation in the smartphone. The forward classical KF is used to estimate the state of each moment, while the backward filter is to obtain more accurate state estimate on the basis of forward filter (Tian, Jiabin, Chunlei, and Huan (2017)).

The flow diagram of LC-GNSS/IMU MAP-PPK algorithm is shown in Figure 18.

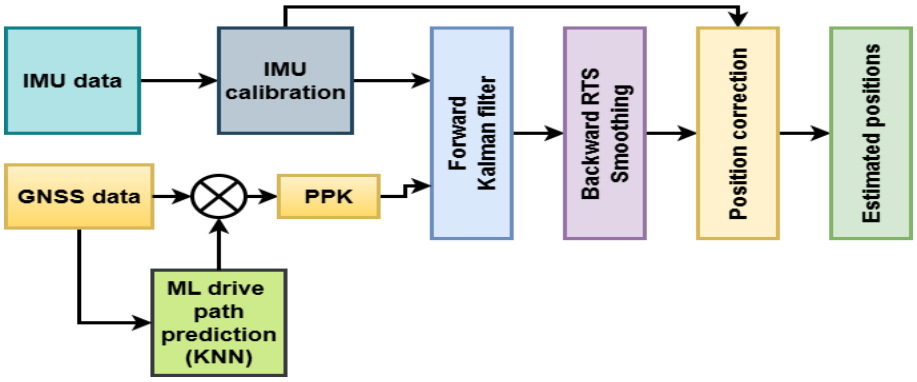


Figure 18. Stages of applying the proposed LC-GNSS/IMU MAP-PPK algorithm.

The state vector of the Kalman filter comprises both GNSS (longitute, latitude) and IMU (accelerometer) epochs, as follows in Equation 3.1:

$$\mathbf{x} = \begin{bmatrix} p^x & v_x & a_x & p^y & v_y & a_y \end{bmatrix}^T \tag{3.1}$$

where p^x, p^y are the (longitude, latitude) positions gained from GNSS. v_x, v_y are the non-measured speeds, and a_x, a_y are the IMU accelerations, respectively. Hence, the measurements are p^x, p^y and a_x, a_y.

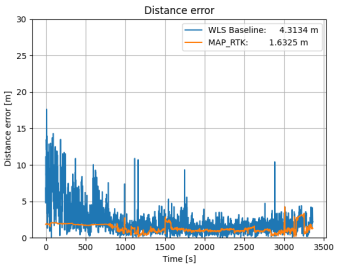
In summary, the steps involved in applying our proposed LC-GNSS/IMU/ML method are outlined as follows, as illustrated below:

1. Data analysis and pre-processing
2. Machine learning-based prediction of driving paths using kNN for MAP algorithm.
3. Application of PPK precise positioning techniques to process the GNSS carrier phase
4. Integration of GNSS/IMU multisensor fusion using KF-RTS algorithms

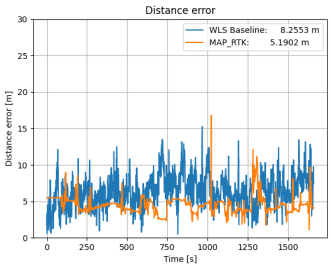
Results:
In publication III., the evaluation criteria for the GSDC were as follows: *"Submissions are evaluated using the average of the 50th and 95th percentile distance errors. For each phone, the horizontal distance error (in meters) between the predicted and ground truth latitude/longitude is calculated once per second. These errors form a distribution from which the 50th and 95th percentile values are extracted — the 95th percentile represents the distance below which 95% of the errors fall. The average of these two percentile errors is computed for each phone, and the final score is the mean of these averages across all phones in the test set."*

We estimated the smartphone's location using the carrier phase information, and the evaluation of our method was performed on the Kaggle platform by comparing the results against the locally generated ground truth. The outcomes from our proposed method demonstrated significant improvements in positioning accuracy across all road driving scenarios when compared to the given WLS solution, as depicted in Figure 19.

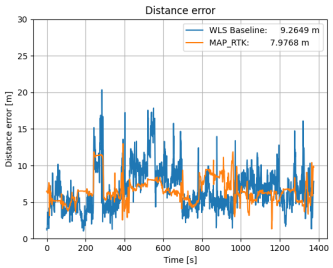
Since the actual ground truth datasets were concealed, the results were compared to the next best available alternative, which was the WLS data. Figure 31 illustrates a significant reduction in positioning errors in favor of our proposed MAP-PPK method when compared to the WLS baseline across all driving scenarios: highway, tree-lined streets, and downtown roads. By applying the ML-based adaptive PPK positioning techniques to the GSDC test datasets, the mean accuracy achieved was 2.61 meters for the public score on the Kaggle.com. However, after the final



(a) Highway.



(b) Tree-lined Street.



(c) Downtown.

Figure 19. Comparison of positioning error between the WLS baseline and proposed method for each driving paths.

evaluation, the private score improved to 2.29 meters, demonstrating a significant enhancement over the original 3-5 meter accuracy.

It was necessary to ensure that the algorithm's complexity was relatively simple. This is because GNSS low-cost receivers often have limited channels and computational resources. The proposed loosely coupled GNSS/IMU/ML method demonstrated acceptable performance in terms of accuracy, robustness, cost, timing, and integrity. However, further validation and investigation are needed to confirm the method's suitability for real-time applications.

In the final published results of GSDC 2022, our team "DigiECO UVA" ranked 69th out of 573 on the leader-board with a mean accuracy of **2.29 meters**, qualifying us for a Bronze Medal (Kaggle (2022)). This result strongly suggests that the proposed LC-GNSS/IMU/ML using MAP-PPK method can significantly improve smartphone position estimation accuracy.

3.7.2 Precise seamless navigation with low-cost GNSS devices

In publication V., we present a system for seamless navigation that combines GNSS with UWB, WiFi, and IMU data to provide accurate indoor and outdoor positioning

on mobile devices. It discusses the integration of these technologies to address the challenges posed by GNSS signal loss in indoor environments and urban canyons. The proposed hybrid system offers continuous, precise navigation across varying environments, making it suitable for real-world applications where GNSS alone may not suffice.

Main contributions of the study include:

- Introduced a hybrid navigation system that combines GNSS, UWB, WiFi, and IMU sensors for seamless indoor and outdoor positioning.
- Demonstrated the effectiveness of the system in providing accurate and reliable navigation in challenging environments, such as urban canyons and indoor spaces.
- Highlighted the advantages of using a multi-sensor fusion approach to overcome GNSS limitations in areas with poor signal reception.

Datasets used:
Publication V. uses a combination of GNSS signals, Ultra-Wideband (UWB) measurements, WiFi data, and IMU outputs to create a dataset for evaluating the proposed system. The dataset includes real-world data captured in indoor and outdoor environments, providing the necessary information for training and testing the machine learning models applied in the system. The specific details of the dataset, such as size, structure, and data collection procedures, are likely discussed in the paper itself. We used an Android smartphone with an application capable of capturing raw RINEX observables. The smartphone used was the Samsung Galaxy Note20 Ultra. The software used was the GEO++ RINEX Logger.

In this process, the Geo++ RINEX Logger uses the most recent Android API services to log the device's raw GNSS measurement data into a RINEX file. The GNSS data are collected for the test area. The outdoor-indoor positioning is performed in Technobothnia. The UWB (Decawave), Wi-Fi, and IMU (Xsense) data are collected in the Technobothnia building using their respective sensors mounted on a robot (Omron robot).

Methods:
In publication V., a seamless navigation system that integrates multiple sensors, such as GNSS, UWB, WiFi, and IMU, for indoor-outdoor positioning was implemented. The system utilizes sensor fusion techniques to combine the strengths of each sensor and mitigate their individual limitations. Machine learning algorithms have been employed to improve the system's accuracy and robustness in dynamic environments. This was achieved using an ML-based indoor-outdoor (IO) detection model to find the IO signal transition pattern. The proposed system was tested in a scenario over 1.1 km including outdoor, and indoor phases using a mobile robot.

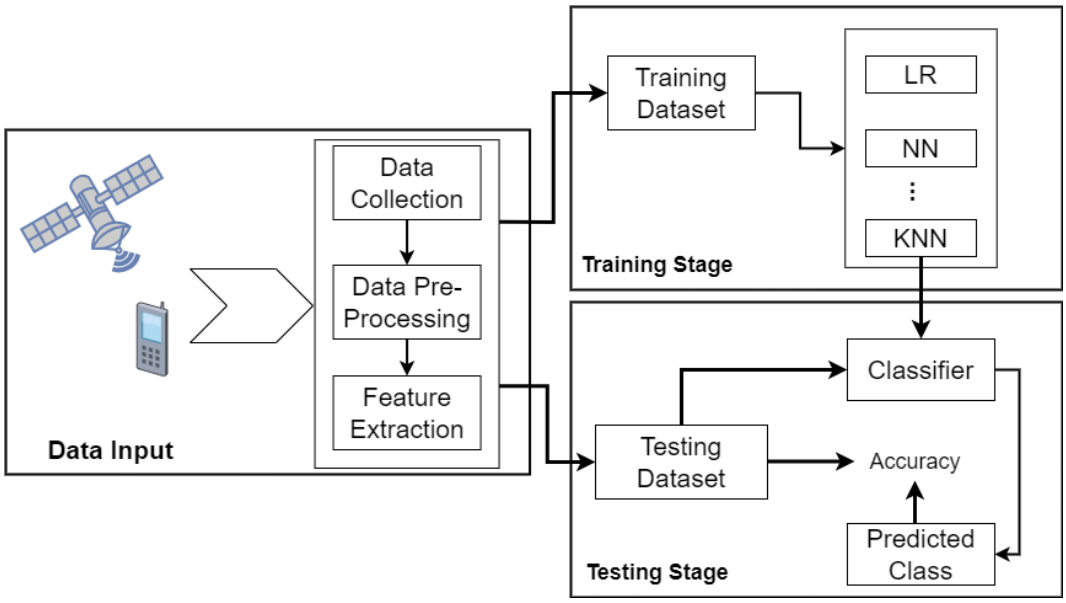


Figure 20. Framework for indoor/outdoor detection. The three processes in this framework include Data Input, Training, and Testing. In the Data Input process, GNSS observation data is recorded, and features are extracted. The data are classified in the Training phase. The classifier is applied to detect the user's current environment in the Testing phase.

The proposed algorithm is categorized into two main stages: Data Input, ML-based IO detection model training and testing. An overview of the indoor/outdoor detection process is illustrated in Figure 20.

The ML algorithms used for the ML-based indoor-outdoor (IO) detection model include Random Forest (RF), Support Vector Machines (SVM) such as support vector classifier (SVC) and Linear SVC, Logistic Regression (LR), Naive Bayesian (NB), and Neural Networks (NN). The training data consist of raw RINEX data from the GNSS sensors on the smartphone. Hyperparameters for each model were selected using grid search with cross-validation to optimize classification accuracy. For the neural network, a feedforward architecture was used with two hidden layers containing 64 and 32 neurons respectively, ReLU activation functions, and a final sigmoid output layer for binary classification. The best-performing classifier was selected for indoor/outdoor detection.

Results:
In publication V., the evaluation criteria for the GSDC were as follows: *The models prediction of the environment was evaluated against the truth data using the Accuracy (Classification) metric.*

The experiment area for the indoor environment is shown in Figure 21. The IO transition is made from outside the building to the section highlighted with green

lines for the robot's trajectory inside the building.

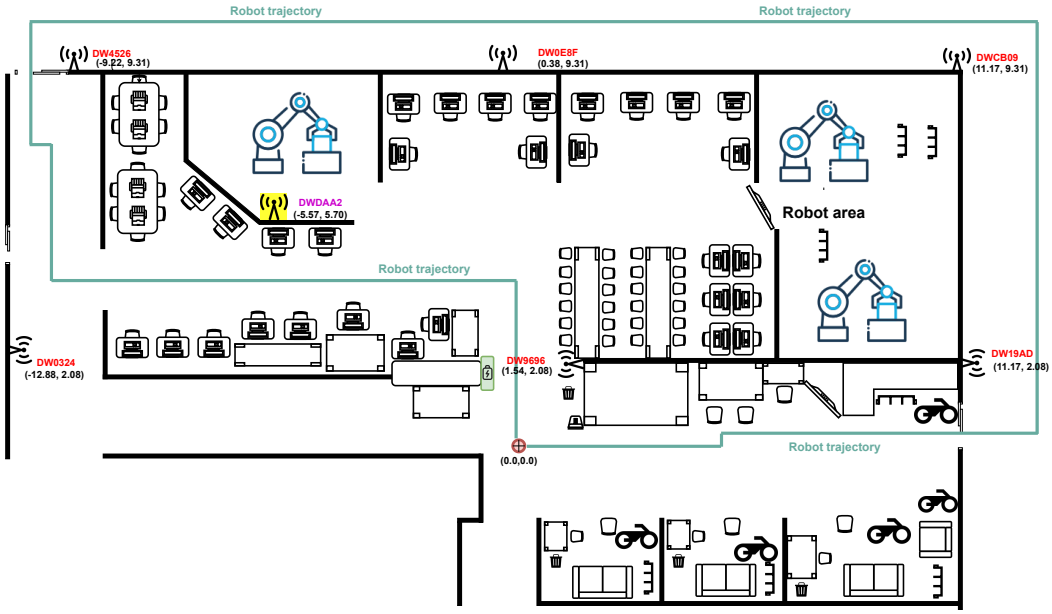


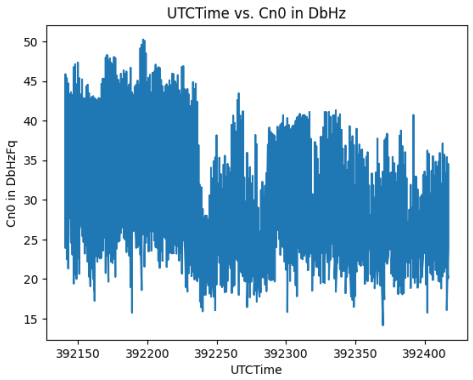
Figure 21. Floor plan of Technobothnia laboratory with the planned robot trajectory.

Accurate detection of the IO transition would lead to the right decision-making in the navigation mode to be employed in seamless positioning. In our studies, we made use of the ML model for indoor-outdoor (IO) detection using an IO signal transition pattern and the available number of satellites. This approach is more reliable than the use of the signal strength of certain sensors embedded in the location system such as a light sensor, Bluetooth, Wi-Fi, magnetometer, or GNSS signal strength. This is because the solutions based on power-hungry sensors, such as Wi-Fi, could be an issue, especially for mobile devices due to limited battery capacity.

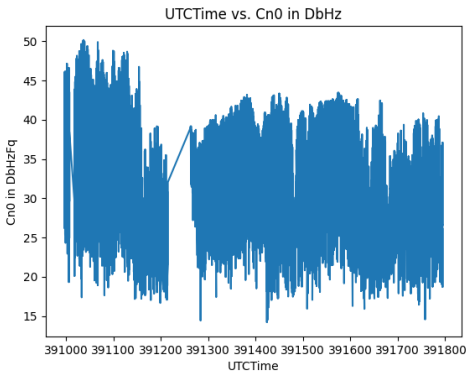
To test the IO detection algorithm, we make use of separate data (the test data) not shown to the ML-based IO detection models during training. The trained ML classifiers are used to predict the detected environment.

From Figure 22 (c) and (d), we can see the sharp drop in the number of satellites after some UTC time epoch. This drop was noticed during the transition from outdoor to indoor (light-indoor). The number of satellites kept fluctuating during the time when moving around the indoor environment. This is also seen in Figure 22 (a) and (b) for the C/NO values.

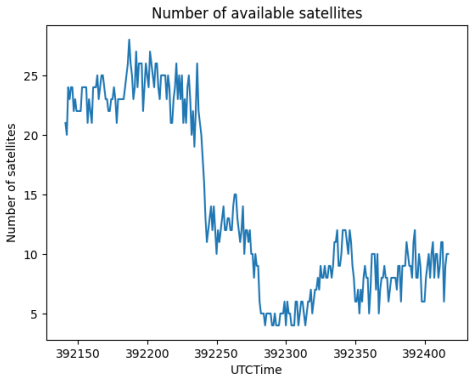
The ML IO detection model is trained to detect the transition pattern, and this can be used in the selection of the appropriate positioning technology to apply. With this, the system can switch from the GNSS/IMU positioning method suitable for outdoor positioning to UWB/IMU for indoor positioning, for example.



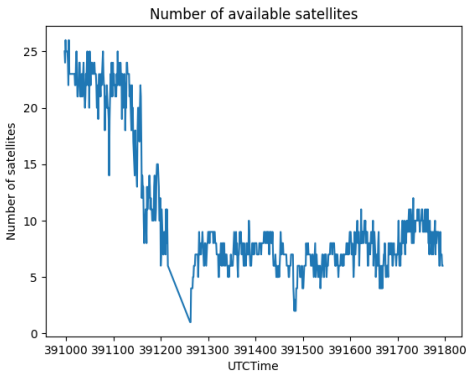
(a) C/No for first trip.



(b) C/No for second trip.



(c) Maximum available satellites for first trip.



(d) Maximum available satellites for second trip.

Figure 22. Comparison of C/No and number of available satellites for two test trips along the same navigation path.

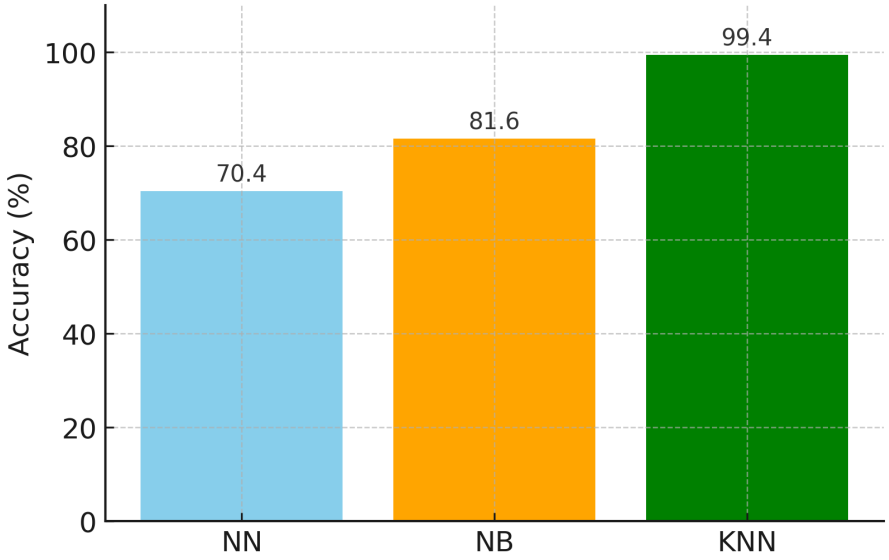


Figure 23. Classification accuracy using different ML classifier algorithms.

Table 10. Confusion matrix for NB classifier.

Environment	Open Outdoors	Light Indoors
Open Outdoors	75.84%	24.16%
Light Indoors	4.88%	95.12%

The classification results for distinguishing between open outdoor and light indoor environments yielded an accuracy of 100% across the LR, SVC, RF, and DT algorithms. This exceptionally high performance is attributed to the significant differences in the observed features between these two environments, such as strong GNSS signal availability and quality in open outdoor areas compared to degraded or absent signals indoors. Given the simplicity of this binary classification task and the clear separability of the classes, achieving perfect accuracy is both plausible and expected in this context.

As a result, these algorithms were excluded from subsequent experiments to focus the evaluation of other algorithms such as NN, NB, and KNN algorithms. The conceptual pipeline for Indoor–Outdoor detection is illustrated in Figure 24.

k-Nearest Neighbors (kNN)

The kNN algorithm was implemented using the `scikit-learn` library. Training and testing datasets were constructed from GNSS and IMU features, with labels indicating indoor or outdoor states. The classifier was trained with varying values of k (from 1 to 14) to evaluate sensitivity to neighborhood size.

For each test sample, the Euclidean distance to all training samples was computed,

and the majority label among the k nearest neighbors was assigned:

$$\hat{y} = \arg\max_{c \in \{0,1\}} \sum_{i=1}^{k} \mathbb{1}(y_i = c)$$

where $\hat{y}$ is the predicted class, and y_i are the labels of the nearest neighbors.

Performance was evaluated using accuracy, confusion matrices, and classification reports. The model achieved competitive accuracy, with results varying depending on the choice of k.

Naïve Bayes (NB)

A Gaussian Naïve Bayes classifier was also implemented using `scikit-learn`. This probabilistic model assumes conditional independence among features given the class label. The likelihood of each feature was modeled as a Gaussian distribution, and classification was performed by maximizing the posterior probability:

$$\hat{y} = \arg\max_{c \in \{0,1\}} P(c) \prod_{j=1}^{d} P(x_j \mid c)$$

where x_j are the feature values and d is the feature dimension.

The NB classifier was trained on the same dataset as kNN and evaluated using confusion matrices and classification reports. Despite its simplicity, NB provided a baseline performance for IO detection.

Neural Network (NN)

A feedforward neural network was implemented using PyTorch Lightning. The architecture consisted of five hidden layers with decreasing dimensions (512, 256, 128, 64, 32), each followed by ReLU activation. The output layer used a sigmoid activation to produce a binary classification (indoor vs. outdoor).

The forward pass is defined as:

$$y = \sigma(W_5 f(W_4 f(W_3 f(W_2 f(W_1 f(W_0 x))))))$$

where $f(\cdot)$ is the ReLU activation, W_i are the weight matrices, and $\sigma(\cdot)$ is the sigmoid function.

Training was performed using the Adam optimizer with a learning rate of 1×10^{-4}. Binary cross-entropy loss was used, and accuracy was tracked during training, validation, and testing. These results indicate that while the neural network was able to capture nonlinear transition patterns, its performance was limited compared to the simpler probabilistic Naïve Bayes model. The NN had the most complex architecture, but its performance was limited by:

Training data size: Neural networks require large datasets to generalize well. If the dataset was relatively small, the NN may have overfit or underfit.

Hyperparameter tuning: Learning rate, batch size, and architecture depth all affect performance. Without extensive tuning, the accuracy of the NN can lag behind simpler models.

Feature engineering: The raw GNSS/IMU features being used without extensive transformations resulted in NN not extracting the most discriminative patterns.

Performance of Indoor–Outdoor Detection Models

The three machine learning algorithms implemented for Indoor–Outdoor (IO) detection were evaluated on their ability to distinguish between open outdoor and light indoor environments. The classification accuracies obtained demonstrate the relative strengths of each approach:

- **Neural Network (NN):** 70.4% accuracy. NN lagged because it requires more data, careful tuning, and possibly feature preprocessing to outperform simpler models.
- **Naïve Bayes (NB):** 81.6% accuracy. NB performed moderately well because feature distributions differed, but correlations limited its effectiveness.
- **k-Nearest Neighbors (kNN):** 99.4% accuracy. kNN excelled because the dataset's indoor/outdoor separation was strong and local neighborhood voting captured it directly.

The kNN classifier achieved the highest accuracy, suggesting that distance-based neighborhood voting was particularly effective for distinguishing between indoor and outdoor GNSS feature patterns. The comparative performance highlights the importance of algorithm selection and tuning in IO detection tasks, as different models exhibit varying capabilities depending on the structure of the data.

This result highlights a common phenomenon in applied machine learning: simpler models can outperform complex ones when the dataset is small, features are well-separated, and assumptions align with the data structure. Neural networks shine

with large, diverse datasets and careful optimization, but in this case, kNN was the most natural fit.

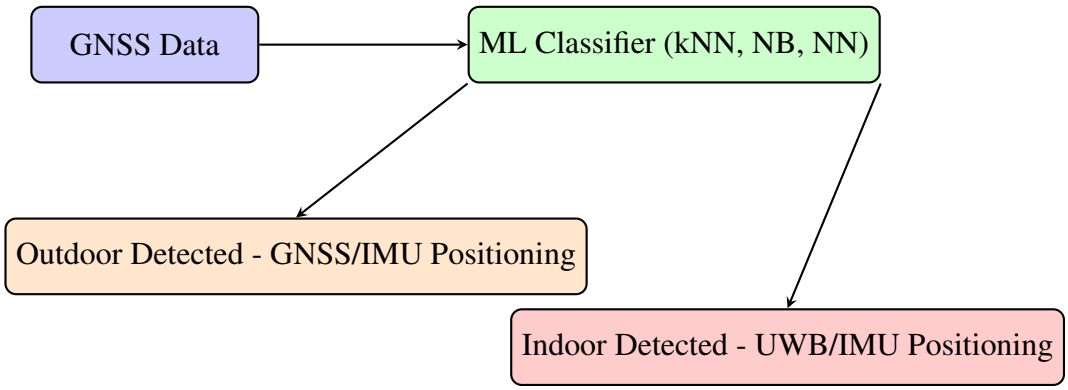


Figure 24. Conceptual pipeline for Indoor–Outdoor detection. GNSS data is classified by ML models (kNN, NB, NN) into indoor or outdoor categories. Based on the classification, the system selects GNSS/IMU positioning for outdoor environments or UWB/IMU positioning for indoor environments.

From Figure 23, we find that all three algorithms included in the study have accuracies better than 70%. The confusion matrix for the NB algorithm is presented in Table 10 which shows that we can detect open outdoor environments correctly. There is a 24.16% possibility of identifying light indoors as open outdoor environments, and a 4.88% possibility of identifying open outdoors as light indoors environments. The NN, NB, and KNN algorithms could distinguish between the open outdoors and light indoors environments with accuracies of 70.4%, 81.6%, and 99.4% respectively.

The IO detection method can be useful in the integration of GNSS with UWB/Wi-Fi/IMU systems to achieve seamless navigation for indoor-outdoor positioning. The integration of GNSS with UWB/Wi-Fi/IMU systems can be done using different integration schemes. The positioning method can be selected using a switching mechanism between, for example, a GNSS/IMU positioning method suitable for outdoor positioning and UWB/IMU for indoor positioning depending on the ML-based IO detection result.

3.8 Factor graph optimization application to GNSS

The Inertial Measurement Unit (IMU) is frequently used in robotics, vehicles, and, of course, also in mobile phones. An IMU is usually made up of an accelerometer, gyroscope, and also a magnetometer. A tactical grade IMU is fully calibrated,

whereas low-cost ones typically come with factory calibrations stored in the IMU registers. The calibration only covers the scaling factors of each axis. The accelerometer measures the specific force that can be derived into the acceleration of a movement, the gyroscope measures the angular speed of the sensor, and the magnetometer measures the Earth's local magnetic field direction and magnitude. The integration of GNSS/IMU using factor graph optimization (FGO) has recently attracted attention as an alternative to the extended Kalman filter (EKF) for GNSS-IMU integration. FGO has been increasingly applied in GNSS research, offering robust solutions for sensor fusion and positioning challenges. Notable studies presented at the Institute of Navigation (ION) GNSS conferences from 2021 to 2024 include: (Wen et al. (2021), Q. Li, Zhang, and Wang (2021)). Researchers have developed an open-source extension that adds custom GNSS factors and Python and MATLAB wrappers to GTSAM. This makes it easier to apply FGO to GNSS positioning problems (Suzuki (2020)).

Datasets:
The dataset used in publication VI. is similar to the dataset used for the GSDC competitions of 2021 - 2023. The dataset provided by Google as part of the GSDC competitions includes reference data from GNSS survey-grade receivers to aid model training and validation (Chow et al. (2023)). The Smartphone Decimeter 2023 dataset is a benchmark collection of raw GNSS and IMU data from smartphones, paired with precise ground truth, designed to test and improve algorithms that aim for decimeter-level positioning accuracy in real-world conditions.

Methods:
The recently introduced formulation of Factor Graph Optimization (FGO) by Indelman et al. (Indelman et al. (2012)) has introduced a fresh perspective on multisensor integration, as highlighted by Chen & Gao (D. Chen and Gao (2019)), and Pfeifer & Protzel (Pfeifer and Protzel (2019b)). This approach is structured as a probabilistic graphical model, featuring nodes representing various system states ($x_i \in \mathbf{X}$) and factors ($f_i \in \mathbf{F}$) corresponding to measurements. The flow chart of the method proposed in this paper is shown in Figure 25

FGO provides a flexible framework for integrating various sensor data measurements ($z_i \in \mathbf{Z}$) through the factorization of the probability distribution, representing the joint probability distribution of all states and measurements as a product of factors, each relating a small subset of variables (Indelman et al. (2012)),

$$f(\mathbf{X}) = \prod_i f_i(\mathbf{X}_i), \tag{3.2}$$

where $\mathbf{X}_i$ be the set of all variables x_i that are connected to the factor f_i by an edge. In terms of non-linear least squares optimization, each factor encapsulates an error function aimed at minimization. The optimal estimate $\mathbf{X}^*$ is determined by

minimizing the overall error across the entire factor graph,

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \sum_i \|e_i(\mathbf{X}_i, \mathbf{Z}_i)\|^2_{\Sigma_i}, \tag{3.3}$$

where Σ is the known covariance of the measurement vector.

In FGO for loosely coupled GNSS/IMU integration, we assume that measurements follow a Gaussian distribution. Considering two types of factors used in the loosely coupled GNSS/IMU integration with FGO, the optimal state set $\mathbf{X}^*$ is obtained by solving the following optimization problem:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \sum_i \|e_i^{\text{GNSS}}\|^2_{\Sigma_i^{\text{GNSS}}} + \sum_i \|e_i^{\text{IMU}}\|^2_{\Sigma_i^{\text{IMU}}}, \tag{3.4}$$

where $\|e\|_\Sigma = e \cdot \Sigma^{-1} \cdot e$ is the Mahalanobis norm of e weighted by covariance Σ, $e_i = z_i - h(x_i)$ is the difference between the received measurement and the measurement expected for the true state, and Σ^{GNSS} and Σ^{IMU} are the covariance matrix for the GNSS and IMU factors, respectively. To solve optimization problem in Equation (3.4), a batch solver like Gauss-Newton (Y. Wang (2012)), Levenberg-Marquardt (Levenberg (1944); Marquardt (1963)), or an incremental one like provided by iSAM2 (Kaess et al. (2012)) can be used.

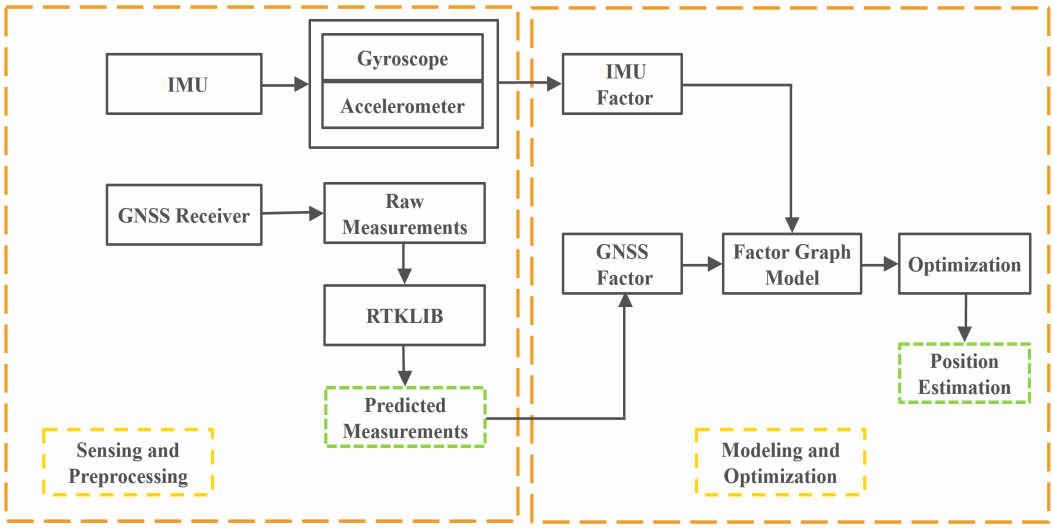


Figure 25. Flowchart of the proposed FGO-based GNSS/IMU integration.

3.8.1 Levenberg–Marquardt (LM) algorithm

The Levenberg-Marquardt algorithm combines gradient descent and Gauss-Newton numerical minimization algorithms. Gradient descent reduces the sum of squared

errors by adjusting coefficients in the direction of steepest descent. Meanwhile, Gauss-Newton reduces these errors by approximating the least squares function as locally quadratic in coefficients and finding its minimum. The Levenberg-Marquardt algorithm behaves akin to gradient descent when coefficients are far from optimal, and shifts towards Gauss-Newton when coefficients are close to their optimal values (Gavin (2019)).

Unlike the traditional EKF-based integration, the FGO captures the posterior probability of states over time, integrating both past measurements and system updates to optimize the entire state set. Historical information plays a crucial role in FGO, where all measurements and states are encoded into a factor graph, and the sensor fusion problem is iteratively tackled through optimization using non-linear optimization method. Consequently, errors arising from linearization steps are minimized. Additionally, FGO adeptly handles delayed measurements by seamlessly incorporating them as supplementary factors into the FGO upon reception. It should be noted that we performed all experiments with the GTSAM framework (Dellaert and Contributors (2022)), in which we used the Levenberg-Marquardt algorithm to solve the problem in Equation (3.4). In addition, we used the IMU preintegration scheme as presented in (Forster, Carlone, Dellaert, and Scaramuzza (2015)).

3.9 Keyframe Policy and Graph Management in the Implemented FGO

Although the fundamentals of Factor Graph Optimization (FGO) and the use of non-linear solvers such as the Levenberg–Marquardt algorithm provide the mathematical backbone for state estimation, the practical implementation of FGO involves several design choices that critically influence the accuracy, robustness, and computational efficiency of the solution. Based on the FGO implementation, the following aspects can be specified.

Policy for Defining Keyframes: In the presented implementation, a new keyframe is defined at every epoch where GNSS measurements are available. Each GNSS observation is converted into a `Pose3` object and inserted into the factor graph as a state variable. Thus, the policy is dense: every GNSS epoch corresponds to a receiver state (pose, velocity, and bias) in the graph.

In this implementation, keyframes are not eliminated. The implementation does not perform marginalization or introduce condensed priors to preserve historical information. Instead, all states remain in the graph until optimization. If elimination were required, GTSAM provides marginalization techniques to compress past information into prior factors, but this functionality is not utilized here.

Number of Keyframes Retained: The factor graph retains all keyframes throughout the trajectory. There is no sliding–window or fixed–lag mechanism implemented. Consequently, the optimization is performed in batch mode, with the graph growing to include all epochs until the end of the dataset.

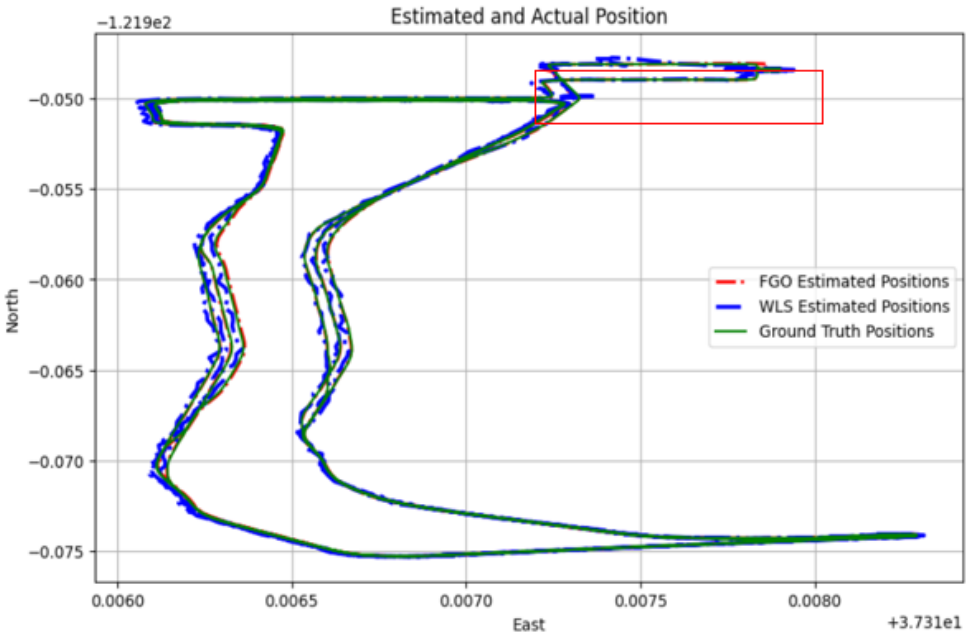
Data Association Strategy: Data association is performed at the epoch level rather than at the satellite measurement level. Each GNSS epoch is preprocessed into Cartesian coordinates (ECEF/NED) and directly associated with the corresponding receiver state through a `PriorFactorPose3`. The graph therefore links each epoch's GNSS–derived position to the receiver state, bypassing satellite–specific pseudorange modeling. This simplifies association but does not explicitly handle multipath, cycle slips, or outlier rejection at the measurement level.

The approach taken implements a dense batch-mode factor graph where every GNSS epoch is a keyframe, all keyframes are retained until optimization, no elimination or marginalization is applied, and data association is performed by linking preprocessed GNSS positions directly to receiver states. Although this approach demonstrates the fundamentals of GNSS FGO, more advanced implementations would incorporate sliding–window smoothing, marginalization for information preservation, and satellite–level data association for robustness.

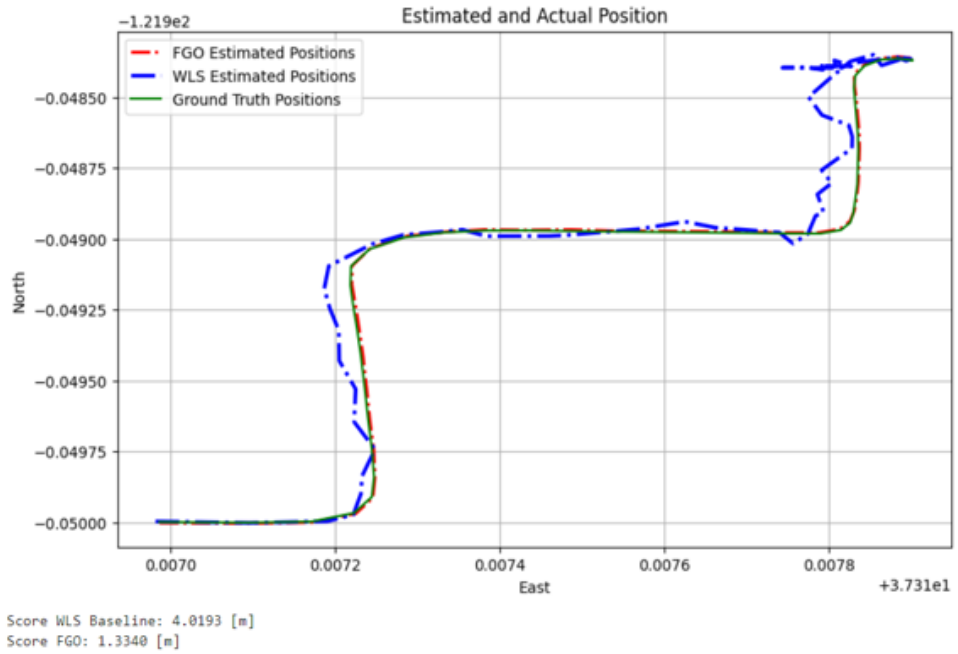
Accurate position estimation is important in several GNSS applications. Recent studies have focused on the comparison of using EKF-based and FGO-based methods for GNSS/IMU integration, with the focus on the potential of the latter. This research shows that the latter approach is more reliable as it helps to reduce the chance of a decrease in position accuracy due to missing GNSS data points and its ability to capture the posterior probability of states over time, integrating both past measurements and system updates to optimize the entire state set. Filling up the missing GNSS data can also enhance combined navigation, ensuring uninterrupted position estimation even during GNSS device degradation or outages.

When comparing the WLS position estimation with the implemented FGO-based GNSS/IMU integration using the Vincenty distance from the ground truth in the training dataset, we observed a significant improvement in the accuracy score. In table 11, we present the accuracy score comparison between the WLS baseline score and the FGO-based GNSS/IMU integration for some selected phones and routes. It was however noticed that at some drive routes, the Samsung phones performed very poorly compared to other phones. The best position estimation accuracy when comparing phones were the Pixel phones and the Pixel6pro to be specific. However, when estimating the percentage increase in accuracy, the Mi8 phone has a percentage increase in accuracy of approximately 65.46%.

An interesting route 2020-12-10-22-52-us-ca-sjc-c in the train dataset had four phones on the same drive route to process. The figures 26 and 27 shows the plot of Mi8 and Pixel4 with the ground-truth, WLS and FGO plots.



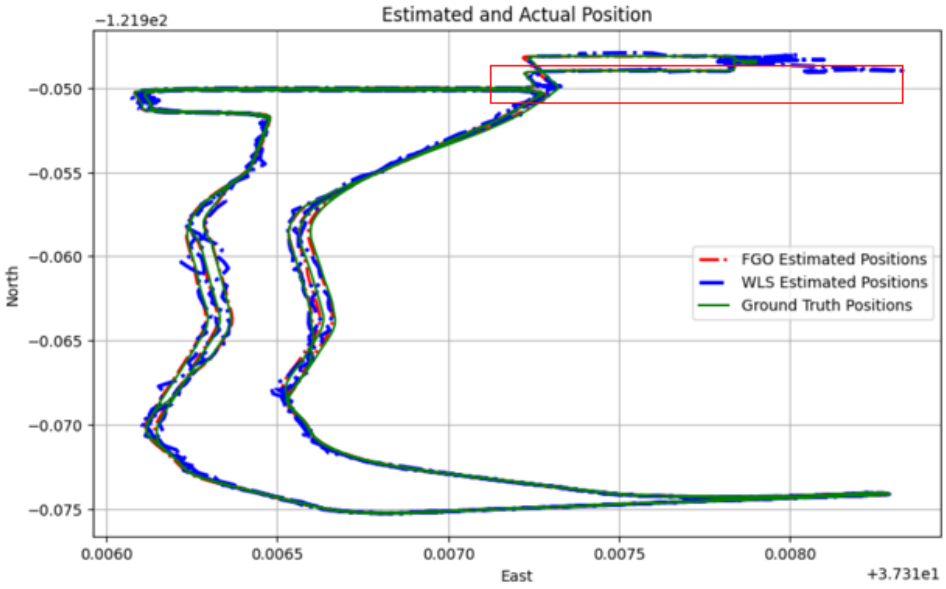
(a) Mi8.



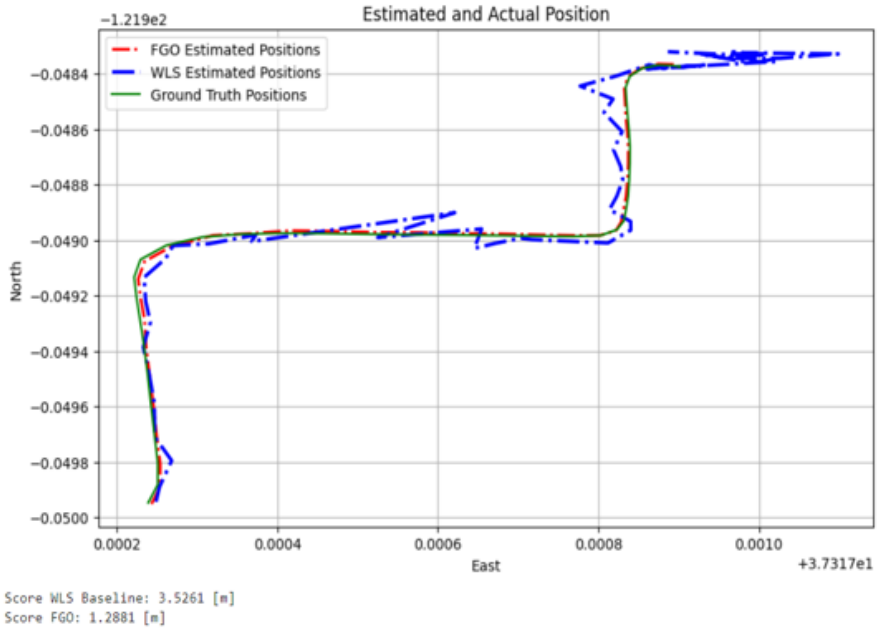
(b) Mi8 zoomed-in.

Figure 26. Comparison of positioning error between the selected phones (Mi8) using WLS baseline and proposed FGO-based method for a specific driving path.

In the GSDC 2023-2024 final published results, although the final place of our team ”DigiEco Univaasa” in the leader-board was 12th out of 279 teams, our method



(a) Pixel4.



(b) Pixel4 zoomed-in.

Figure 27. Comparison of positioning error between the selected phones (Pixel4) using WLS baseline and proposed FGO-based method for a specific driving path.

achieved mean accuracy of 1.661 meters (Kaggle (2023)). This results supports the claim of previous studies that FGO provides better results in the integration of

Table 11. Position estimation accuracy.

Phone	WLS Baseline [m]	FGO [m]
Pixel4	2.94	1.47
Pixel4xl	3.64	1.36
Pixel5	5.03	1.48
Pixel6pro	3.85	0.99
Pixel7pro	5.39	1.46
Samsung	4.43	1.96
Mi8	5.99	2.07

GNSS/INS compared to KF-based integrations.

When the FGO-based method of integration was compared with Kalman filtering (KF-RTS) for the same integration process, our KF-RTS-based results (publication V.) gave an accuracy of 2.291 meters (Siemuri, Elsanhoury, et al. (2022)) while the accuracy of our FGO-based implementation in our latest publication VI. was 1.661 meters (Siemuri et al. (2023)) when evaluated on www.kaggle.com for the GSDC 2023-2024 competition. There was a 27.50% increase in the accuracy.

4 LEO SATELLITES: ORBIT DETERMINATION AND POSITIONING

The use of GNSS for precise orbit determination (POD) of low-Earth-orbit (LEO) satellites has played a pivotal role in advancing various space applications, including navigation, telecommunication, remote sensing, and Earth observation systems. These applications greatly benefit from precise satellite orbit tracking enabled by onboard GNSS receivers. Recent advancements have led to the development of GNSS receivers specifically designed to meet POD requirements, which have been deployed on numerous satellites, tailored to their mission goals that necessitate accurate orbit knowledge. The performance of the POD process is influenced by factors such as the measurement environment, the technique employed, and the satellite's mission application. In addition to accuracy, reducing latency in obtaining precise solutions has also become an area of significant interest (Siemuri, Selvan, Kuusniemi, Valisuo, and Elmusrati (2022)). This benefits numerous end-user applications by enabling faster access to the required orbit solutions (Gebre-Egziabher and Gleason (2009)). Machine learning for orbit determination of LEO satellites has been explored in some studies, as highlighted in the systematic literature review (SLR) in publication IV.. However, it was observed that very few studies have investigated the use of machine learning for the precise orbit determination (POD) of LEO satellites. This may be due to the fact that most GNSS navigation-related research focuses on improving positioning accuracy and mitigating GNSS errors. Moreover, LEO satellites equipped with GNSS receivers have only recently become available, which explains the limited research on their POD via GNSS, and the relatively small body of work involving ML models for LEO satellites and GNSS.

From the insights gotten from the SLR in publication IV., we performed a study on the application of *"Machine Learning for LEO and MEO Satellite Orbit Prediction"*, presented at the 37th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2024), which aims to enhance the accuracy of orbit predictions for Low Earth Orbit (LEO) and Medium Earth Orbit (MEO) satellites (publications VII.). The methodologies and datasets are presented in the next sections.

4.1 Methods and datasets

For the Low Earth Orbit (LEO) scenario, the model is trained using the precise ephemeris data of the SWARM-A satellite, provided in SP3 format. SWARM-A is part of the European Space Agency's (ESA) SWARM constellation and operates in a near-polar circular orbit at an altitude of approximately 460 km (Friis-

Christensen, Lühr, and Hulot (2006)). The ephemeris data includes the observation time, position, velocity, and satellite clock parameters. In this study, the time series of SWARM-A's dynamic position and velocity, referenced to the center of mass and expressed in the International Terrestrial Reference Frame (ITRF), is utilized. The data has a 10-second sampling interval, meaning the positions and velocities are recorded every 10 seconds. Ephemeris data spanning from October 2, 2023, to January 31, 2024, was retrieved from the ESA portal (ESA (2024)).

For the Medium Earth Orbit (MEO) scenario, the model is trained using the final ephemeris data of GNSS satellites, provided by the International GNSS Service (IGS) in SP3 format. The GNSS dataset includes only the 32 GPS satellites, which operate in multiple orbital planes. The ephemeris data comprises observation time, position, velocity, and satellite clock parameters. The analysis leverages time series data for the dynamic positions and velocities of the GPS satellites. With a sampling interval of 5 minutes, the positions and velocities of all 32 GPS satellites are recorded every 5 minutes. Data corresponding to GPS weeks 2086 through 2251 was collected from the NGA portal for this analysis (NGA Public Affairs (2023)). The summary of the data processing steps are listed below:

1. **Data Selection:**
    - Different durations of data are considered for machine learning (ML) models: 1, 3, 10, 30, and 50 days.
    - For Medium Earth Orbit (MEO) satellites, data processing focuses on a single GPS satellite at a time, including its PRN number.
2. **Data Retrieval and Transformation:**
    - Satellite tracking data (time, position, velocity) is retrieved for the SWARM-A satellite or a GPS satellite.
    - Data is transformed from the International Terrestrial Reference Frame (ITRF) to the Geocentric Celestial Reference System (GCRS) to ensure consistency.
3. **Interpolation for Temporal Resolution:**
    - Lagrange interpolation is applied to enhance temporal resolution by generating a polynomial that passes through measured data points.
    - For SWARM-A satellite data sampled every 10 seconds, interpolation increases the dataset to one point per second, resulting in 86,390 points over 24 hours.
4. **Prediction Time Intervals:**
    - For SWARM-A, prediction intervals of 1, 3, 5, 7, 9, and 10 seconds are analyzed, alongside varying the number of days of input data.

- Interpolated points (black dots) fill gaps between actual data points (red dots) to create denser datasets, as illustrated in Figure 28.

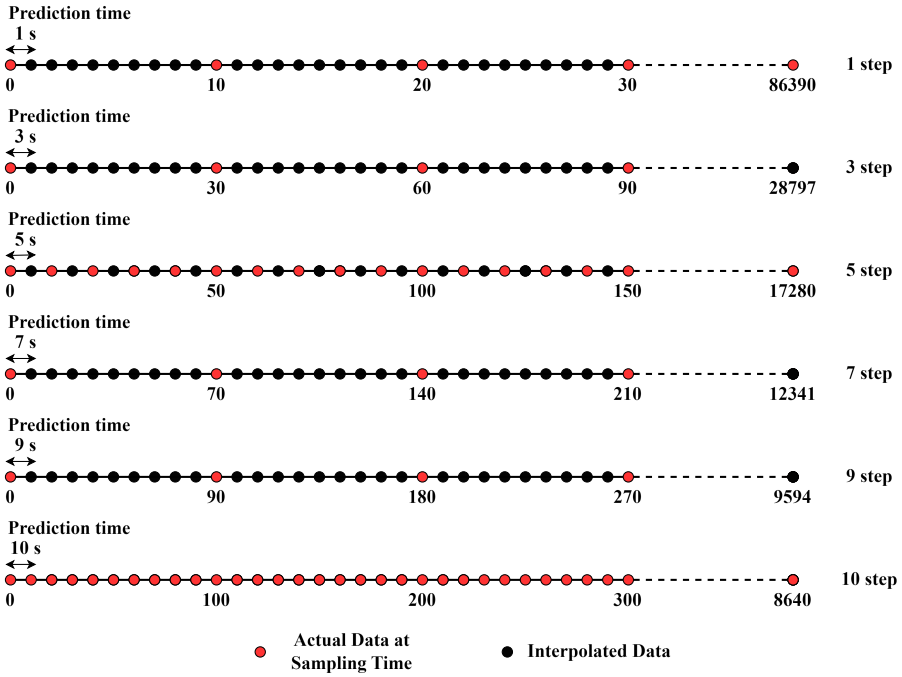


Figure 28. Interpolating points between the observed data points for LEO SWARM-A satellite data using Lagrange interpolation method.

These processing steps ensures the preparation of high-resolution, consistent datasets for the ML-based satellite orbit prediction. Upon completing the interpolation process, the dataset now includes both observed and interpolated data points. Utilizing this enriched dataset, the developed algorithm generates the input features and labels required for training the machine learning (ML) models. The input features consist of the historical positions P_x, P_y, P_z and velocities V_x, V_y, V_z of the SWARM-A and GPS satellites.

After generating the input features and labels, the dataset is split into training (70%) and testing (30%) subsets. From our simulations, we found that this 70/30 split balances model learning and evaluation: 70% provides enough data for the model to learn patterns effectively, while 30% ensures a reliable assessment of its performance on unseen data. The model's predictive performance is subsequently assessed using the testing subset. This thorough data preparation process ensures that the ML model is trained with a high-quality dataset, enhancing its reliability and effectiveness.

4.1.1 ML-Based prediction method for satellite orbits

This section outlines the methodology for predicting satellite orbits using machine learning (ML) models based on precise ephemeris data. Four ML models are implemented: **KNR (K-Nearest Regressor)**, **RFR (Random Forest Regressor)**, **LR (Linear Regression)**, and **PR (Polynomial Regression)**.

Input Features and Prediction

- **Input features:** Historical positions (P_x, P_y, P_z) and velocities (V_x, V_y, V_z) of SWARM-A and GPS satellites.
- **Prediction:** The trained ML models predict position and velocity offsets based on test data.

The general prediction equation is:

$$\hat{y} = C - \hat{N} \tag{4.1}$$

where:

- $\hat{y}$: Predicted offset (position or velocity),
- C: Current position or velocity,
- $\hat{N}$: Estimated position or velocity.

From the offsets, the estimated position or velocity is:

$$\hat{N} = C - \hat{y} \tag{4.2}$$

4.1.2 Performance evaluation

The root-mean-square error (RMSE) is used to evaluate model accuracy:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(N_i - \hat{N}_i)^2} \tag{4.3}$$

where:

- n: Number of observations,

- N_i: True values,
- $\hat{N}_i$: Predicted values.

4.2 ML models implemented

4.2.1 KNR model

The KNR model is a non-parametric algorithm that predicts the dependent variable $\hat{y}$ based on the average of the k nearest neighbors:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{4.4}$$

where:

- $\hat{y}$: Predicted offset (position or velocity),
- k: Number of nearest neighbors,
- y_i: Target values of the nearest neighbors.

4.2.2 RFR model

The RFR model uses an ensemble of decision trees to make predictions, capturing non-linear relationships. The general formula is:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(x) \tag{4.5}$$

where:

- $\hat{y}$: Predicted offset (position or velocity),
- T: Number of decision trees,
- $f_t(x)$: Prediction of the t^{th} decision tree for input x.

4.2.3 LR model

The LR model is a simple regression technique defined by:

$$y = \beta X + \epsilon \tag{4.6}$$

where:

- y: Predicted offset (position or velocity),
- X: Input features (historical positions and velocities),
- β: Coefficients,
- ϵ: Error term.

4.2.4 PR model

The PR model fits a polynomial function to data to capture non-linear relationships:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \epsilon \tag{4.7}$$

where:

- y: Predicted offset (position or velocity),
- $\beta_0, \beta_1, \ldots, \beta_n$: Polynomial coefficients,
- ϵ: Error term.

4.3 Results

As outlined in section 4.1, we consider different durations of training data for the ML models, including 1, 3, 10, 30, and 50 days. For clarity, we focus on presenting and analyzing the performance of the ML models that achieved the best results with a specific data duration. In this case, the models showed optimal and reliable performance when trained with 3 days of data. Therefore, the subsequent performance comparison of ML models is based on this 3-day dataset, with the root-mean-square error (RMSE) serving as the primary evaluation metric.

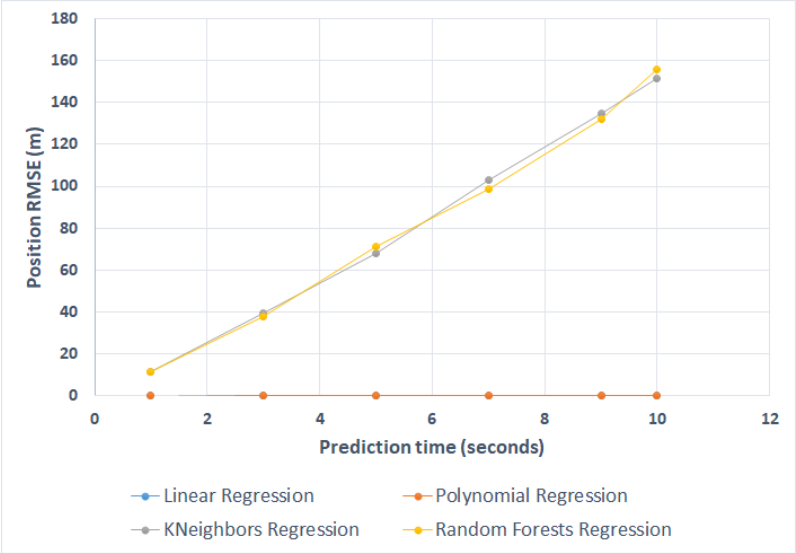


Figure 29. Position RMSE of LEO SWARM-A satellite for implemented ML models across various prediction time intervals.

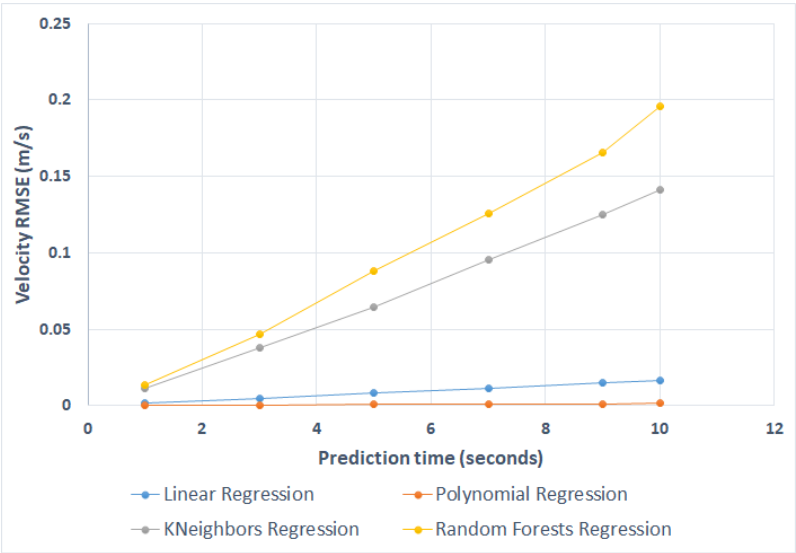


Figure 30. Velocity RMSE of LEO SWARM-A satellite for implemented ML models across various prediction time intervals.

Figures 29 and 30 illustrates that, for the Low Earth Orbit (LEO) SWARM-A satellite, the Polynomial Regression (PR) model consistently outperformed the K-Nearest Neighbor Regression (KNR), Random Forest Regression (RFR), and Linear Regression (LR) models in both position RMSE (m) and velocity RMSE (m/s). For position RMSE in Figure 29, the PR model achieved the lowest error across different prediction intervals, with the LR model following closely. However, due to the high position RMSE values of the KNR and RFR models, the LR model's results appear nearly indistinguishable from those of the PR model in the plot. For velocity

RMSE in Figure 30, the PR model again exhibited the lowest error compared to the other models. Additionally, while the RFR model performed better than the KNR model, the PR model demonstrated superior accuracy overall, highlighting its effectiveness in estimating the position and velocity of the LEO SWARM-A satellite.

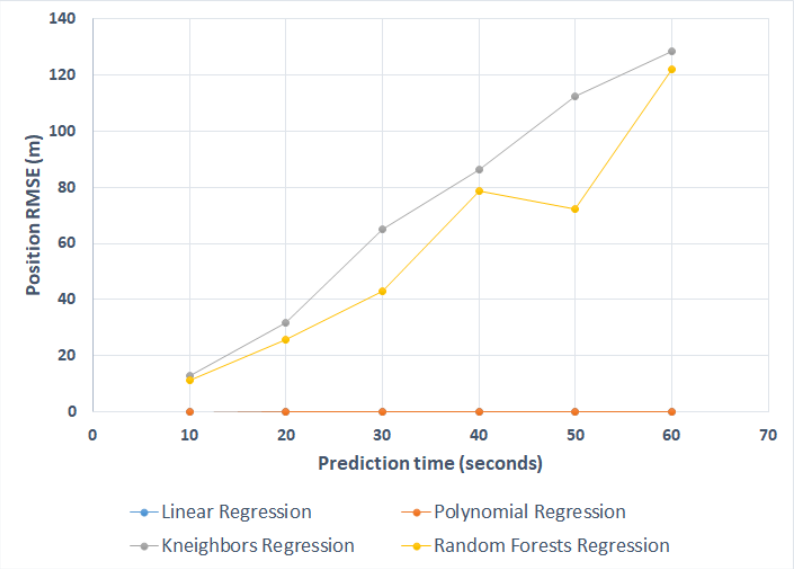


Figure 31. Position RMSE of MEO GPS satellite PRN-14 for implemented ML models across various prediction time intervals.

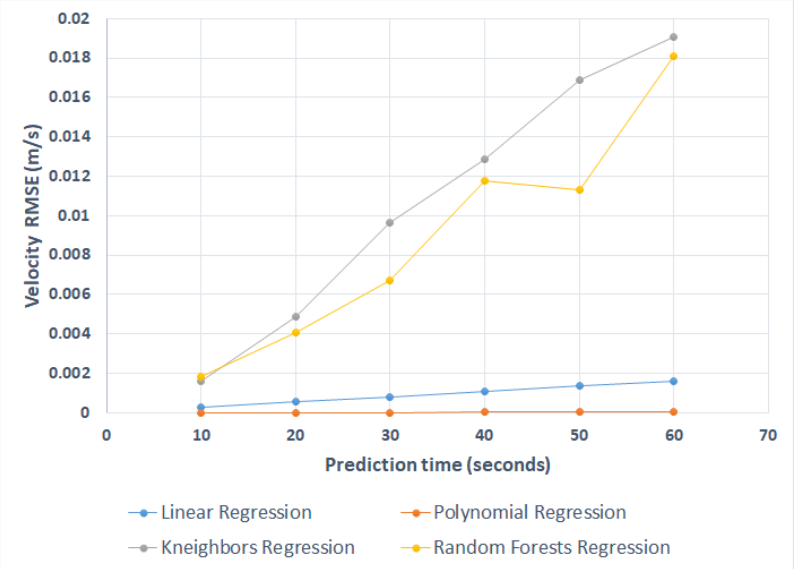


Figure 32. Velocity RMSE of MEO GPS satellite PRN-14 for implemented ML models across various prediction time intervals.

Similarly, the ML models were evaluated for Medium Earth Orbit (MEO) GPS satellites to estimate position and velocity based on predicted offsets. Figures

31 and 32 shows that, for the GPS satellite PRN-14, the PR model delivered the best overall performance compared to KNR, RFR, and LR models in both position RMSE (m) and velocity RMSE (m/s). The PR model consistently achieved the lowest RMSE values, demonstrating its superior capability in accurately estimating the position and velocity of the MEO GPS satellite PRN-14.

This work demonstrates how ML can overcome limitations in traditional orbit prediction by effectively modeling the complex dynamics of satellite motion. ML frameworks estimate satellite orbits through position and velocity offsets, refined via interpolation, with RMSE minimized to improve accuracy. Each model offers distinct strengths, depending on the data structure and complexity, supporting robust and reliable orbit prediction.

5 EMBEDDED ML APPLICATIONS FOR LOCALIZATION

This section presents the benefits and challenges of implementing ML models on edge devices in a distributed manner, with a focus on adapting or designing ML models for deployment on resource-constrained edge devices. We examine approaches for caching, training, inference, and offloading ML models on edge devices. We proceed to analyze the benefits and limitations of deploying ML models on edge devices. TinyML (Tiny Machine Learning), a new paradigm to support ML-based techniques running on ultra-low-power embedded devices on edge, emerges as an exciting solution to execute the necessary AI techniques directly on the device to be tracked (Dutta and Bharali (2021)). In our publication VIII., we address the benefits, applications, and challenges of the application of federated learning (FL) and TinyML for localization (Siemuri and Elmusrati (2025)).

TinyML is a branch of machine learning focused on deploying small, efficient ML models directly on resource-constrained edge devices, such as microcontrollers and sensors. These devices typically have limited computational power, memory, and battery life, making it challenging to run conventional ML models. TinyML enables real-time, low-power data processing and analytics directly on these devices without relying on constant cloud connectivity (Ray (2022)).

FL, on the other hand, is a distributed ML approach where multiple devices collaboratively train a shared model under the orchestration of a central server. Instead of transmitting raw data, devices send model updates, preserving data privacy. The integration of FL with TinyML—often termed Federated TinyML—allows resource-limited devices to participate in collaborative learning, making it particularly suitable for IoT scenarios where devices are numerous and heterogeneous (Ficco et al. (2024); Ren, Anicic, and Runkler (2023)).

By optimizing ML models to fit within the constraints of edge devices, TinyML allows for applications like voice recognition, image classification, anomaly detection, and predictive maintenance in environments where cloud access may be limited or where data privacy is critical. This field has grown rapidly with advancements in model compression, quantization, and specialized hardware accelerators tailored for on-device ML.

5.1 Training AI/ML models on the edge

Training AI/ML models directly on edge devices like the Raspberry Pi is possible but limited by constrained computational resources (Sikiru, Adepoju, Adebayo,

Salawu, and Olawoyin (2024)). For optimal performance, Raspberry Pi 4 (or later) with at least 4GB of RAM is recommended, optionally enhanced with accelerators such as the Google Coral USB Accelerator, Intel Movidius Neural Compute Stick, or NVIDIA Jetson Nano.

The software stack typically includes Raspberry Pi OS or a lightweight Linux distribution. Python is the preferred language due to its simplicity and ecosystem support. Lightweight frameworks such as TensorFlow Lite, PyTorch Mobile, Edge Impulse, and scikit-learn are commonly used for developing and deploying models on resource-constrained hardware (Banbury, Reddi, Torelli, and et al. (2021)).

In the context of IoT localization, we can produce location based predictions faster and without the need for transmitting large amounts of raw data across a network by running ML algorithms on edge devices such as laptops, smartphones, and embedded systems found in smartwatches, washing machines, cars, manufacturing robots, etc. Figure 33 shows the use of ML on edge devices.

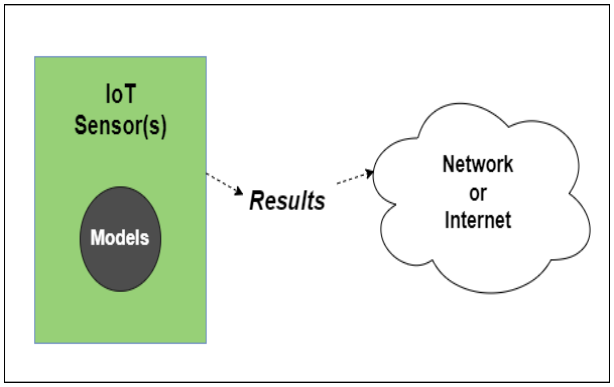


Figure 33. Machine learning model on edge device, adapted from Subramani and Araujo (2022).

Training on the edge is feasible for simple models using efficient libraries (e.g., scikit-learn) or microML frameworks. However, it is often more practical to train complex models on a cloud platform or high-performance local machine, and then deploy the optimized models to edge devices for inference.

To improve inference efficiency, models should be optimized using quantization (e.g., `int8`, `float16`), pruning, or knowledge distillation (Mittal (2019)), and converted to edge-compatible formats such as TensorFlow Lite, ONNX, or Edge TPU binaries.

The development workflow typically involves data collection (on-device or externally), lightweight preprocessing (on-device or offloaded), model training (simple models on-device, complex models externally), and deployment using runtime environments like TensorFlow Lite Interpreter, PyTorch Mobile, or ONNX Runtime.

Remote monitoring and update mechanisms are essential for continuous model improvement.

Key tools supporting this workflow include TensorFlow, PyTorch, and scikit-learn for training; TensorFlow Lite, PyTorch Mobile, and ONNX Runtime for deployment; and edge-specific solutions like Edge Impulse or TensorFlow Lite for Microcontrollers for streamlined integration on low-power devices (Warden and Situnayake (2019)).

5.2 Cloud-Based training and edge deployment for AI/ML models

A common approach in edge AI involves training models in the cloud and deploying them to edge devices. This enables the use of powerful cloud infrastructure (e.g., GPUs, TPUs) for developing large-scale models while maintaining low-latency, efficient inference on edge hardware (Shi, Cao, Zhang, Li, and Xu (2016)).

The cloud workflow begins with data collection and preprocessing, either from edge devices or external sources. Preprocessing ensures data quality via normalization, augmentation, or filtering. Model training is performed using frameworks like TensorFlow, PyTorch, or Keras, typically on platforms such as Google Cloud AI Platform, AWS SageMaker, or Azure Machine Learning.

Post-training, models are optimized through quantization, pruning, and distillation (Han, Mao, and Dally (2016); TensorFlow Team (2020)), then converted to edge-compatible formats like TensorFlow Lite, ONNX, or TensorRT (for NVIDIA devices). These models are transferred to edge devices configured with interpreters such as TensorFlow Lite Interpreter, PyTorch Mobile, or ONNX Runtime. Compatibility with accelerators like Coral USB or Jetson Nano ensures efficient performance (Sze, Chen, Yang, and Emer (2020)). Figure 34 demonstrates the deployment of ML model at the edge.

This cloud-to-edge workflow offers scalability, efficient inference, flexibility for iterative updates, and reduced hardware demands on edge devices (Premsankar, Di Francesco, and Taleb (2018)).

Typical use cases include smart surveillance (e.g., on-camera object detection), industrial IoT (e.g., predictive maintenance from sensor data), and voice assistants (e.g., real-time speech recognition).

Key platforms enabling this pipeline include Google Cloud AI, AWS SageMaker, and Azure ML for training; TensorFlow Model Optimization Toolkit, ONNX tools,

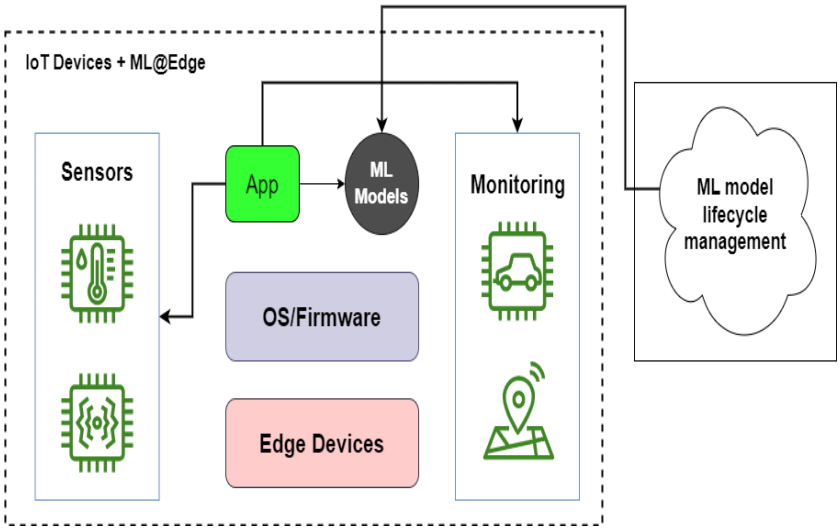


Figure 34. Deploying machine learning at the edge, adapted from Subramani and Araujo (2022).

and TensorRT for refinement; and TensorFlow Lite, PyTorch Mobile, and Edge Impulse for deployment on edge devices.

5.3 Federated Learning Approaches

Federated Learning enables collaborative training across edge devices without centralizing data. In localization contexts, devices can aggregate insights from local RF signals or sensor data to build global models (Bari and Yelamarthi (2025)). This is particularly advantageous for TinyML, as it distributes computational load and enhances model robustness through diverse data sources (Subramani and Araujo (2022)).

FL workflows involve local training on devices, followed by aggregation of model updates on a central server. Techniques like FedAvg (Federated Averaging) are commonly used (McMahan, Moore, Ramage, Hampson, and y Arcas (2017)). For resource-constrained settings, optimizations such as secure aggregation and differential privacy are integrated to bolster security and efficiency (Heikkilä (2025)). Recent advancements include personalized FL, where models are adapted to individual device contexts for better performance (Bari and Yelamarthi (2025)).

5.4 Applications of TinyML in localization

TinyML enables real-time, on-device processing for localization tasks, making it ideal for scenarios where connectivity is limited or power constraints are critical. One such application is *environment or zone classification*, where TinyML models distinguish between different areas (e.g., kitchen vs. living room) using Wi-Fi RSSI, Bluetooth, or other RF signals. Similarly, *activity-based localization* uses IMU data to infer a user's context or location, such as detecting whether a person is walking down a hallway or climbing stairs, aiding pedestrian dead reckoning (PDR).

In industrial and IoT environments, TinyML supports *anchor-free positioning* and *RF fingerprinting* by learning signal patterns without relying on fixed infrastructure (Mendez, Zennaro, Altayeb, and Manzoni (2024)). This includes estimating positions from IMU sequences, tagging UWB-based gestures, or classifying zones based on RF characteristics. Applications also extend to *asset tracking* using WiFi RSSI information, where edge devices monitor object presence and movement in an indoor environment (Mendez et al. (2024)).

Furthermore, TinyML enhances *adaptive localization systems* that switch modes (e.g., from GNSS to UWB) depending on the environment, using lightweight classifiers for seamless transitions. It also aids in wildlife or personnel tracking with wearable sensors by estimating location changes from sparse GNSS and IMU inputs. These capabilities make TinyML a key enabler of intelligent, responsive, and power-efficient localization in edge-centric systems (Repole (2024)).

In this context, the use of TinyML on low-cost GNSS/UWB devices, becomes highly interesting and relevant. Most modern smartphones are now equipped with both GNSS and UWB chipsets, making it possible to utilize them for both outdoor (GNSS) and indoor (UWB) positioning. This opens up new opportunities for implementing seamless localization using widely available consumer devices, reducing the dependence on high-grade positioning hardware and significantly lowering the barrier to entry for advanced location-aware applications.

There are not many publications in the area of TinyAI/ML applications in localization. The available studies were mostly published recently, between 2023 and 2024. Two interesting research publications are presented here. The paper (Avellaneda, Mendez, and Fortino (2023)) (**A TinyML Deep Learning Approach for Indoor Tracking of Assets**) presents the design, implementation, and deployment of embedded devices for indoor asset tracking, enabling object localization within a defined environment by transmitting data to a central system. The approach leverages TinyML to facilitate deep learning on resource-constrained devices, allowing the system to account for various external error factors that impact the accuracy of traditional positioning algorithms. A deep learning-based neural network is trained and deployed at the edge, with Edge Impulse serving as the primary platform for

data standardization, pre-processing, model training, evaluation, and deployment. The final system effectively classifies real data from the installed TAGs, achieving an accuracy of 88%, which can be enhanced to 94% with the integration of a post-processing stage.

While the study (Mendez et al. (2024)) (**On TinyML WiFi Fingerprinting-Based Indoor Localization: Comparing RSSI vs. CSI Utilization**) investigates the use of TinyML for indoor localization through WiFi fingerprinting. It compares Received Signal Strength Indicator (RSSI) and Channel State Information (CSI) in terms of their effectiveness for localization tasks. Two distinct scenarios were evaluated: single-sample data and time-series data, each utilizing different configurations of the trained neural network. The findings indicate that the ML model based on CSI data consistently outperformed its RSSI-based counterpart, with a notable performance advantage in the time-series scenario. The findings indicate that ML models based on CSI data consistently outperform their RSSI counterparts, with a significant performance gap observed in the time-series scenario. The model utilizing CSI data consistently outperformed the RSSI-based model across all configurations. For single-sample classification, the CSI model achieved accuracies of 89.4% and 98.4%, surpassing the RSSI model by 5–6%. However, the performance gap widened significantly in the time-series scenario, where the CSI model outperformed the RSSI model by more than 70% when using 5 samples and by 50–60% when using 10 samples. The exception is the simplest neural network configuration (2 layers, 20–10 neurons), which performs poorly in both cases.

5.5 Advantages of edge ML/TinyML

TinyML offers several advantages in the context of seamless indoor and outdoor localization:

- **Low Power Consumption** – TinyML models run efficiently on edge devices, reducing energy usage and extending battery life for IoT applications (Schizas, Karras, Karras, and Sioutas (2022)).
- **Real-Time Processing** – On-device inference enables quick decision-making without relying on cloud computing, essential for seamless localization (Elhanashi, Dini, Saponara, and Zheng (2024)).
- **Reduced Latency** – By eliminating the need for continuous communication with remote servers, TinyML ensures faster response times in dynamic environments (Elhanashi et al. (2024)).
- **Privacy and Security** – Processing data locally minimizes the risk of data

breaches and reduces reliance on external networks for sensitive location information (Schizas et al. (2022)).

- **Robustness in GNSS-Denied Environments** – TinyML can integrate multiple sensor inputs (IMU, UWB, BLE, Wi-Fi) to maintain accurate positioning even when GNSS signals are weak or unavailable (Luo, Yin, and Gui (2024)).
- **Adaptive Learning** – TinyML models can be fine-tuned for different environments, allowing for improved scenario detection and localization accuracy (Subramani and Araujo (2022)).

These benefits make TinyML a valuable solution for context-aware and power-efficient localization across different environments. By optimizing ML models to fit within the constraints of edge devices, TinyML allows for applications like voice recognition, image classification, anomaly detection, and predictive maintenance in environments where cloud access may be limited or where data privacy is critical. This field has grown rapidly with advancements in model compression, quantization, and specialized hardware accelerators tailored for on-device ML.

5.6 Challenges of edge ML/TinyML

Deploying ML models on edge devices, introduces several significant challenges. These challenges span computational limitations, energy constraints, model accuracy, interoperability, and security and privacy concerns.

One of the primary constraints is the limited computational and memory resources available on edge hardware such as microcontrollers and single-board computers. These devices often lack the processing power required to train or infer complex models efficiently, necessitating the use of lightweight models, quantization, pruning, and model distillation techniques (Banbury, Zhou, Fedorov, and et al. (2021)). Despite these optimizations, maintaining model accuracy while reducing size and complexity remains a significant trade-off (Lin, Chen, and et al. (2020)).

Energy efficiency is another major concern, especially in battery-operated or remote applications. Inference must be optimized not only for latency but also for power consumption, which can significantly affect device lifespan and usability (David, Warden, and et al. (2020)).

Interoperability and toolchain fragmentation also present obstacles. The ecosystem for TinyML includes a diverse range of hardware and software platforms, which often lack standardized APIs or deployment workflows. This diversity complicates development and model portability across platforms (Warden and Situnayake (2019)).

Security and privacy are critical issues in Edge ML, particularly when dealing with sensitive data such as images, audio, or location information. Models deployed on edge devices may be vulnerable to adversarial attacks or model extraction. Moreover, on-device processing is often advocated as a privacy-preserving solution; however, ensuring that data never leaves the device and remains secure requires rigorous engineering and compliance with privacy standards (Yousefpour and et al. (2019)).

These challenges necessitate careful design choices and robust toolchains to make Edge ML both viable and trustworthy in real-world deployments.

In conclusion, training and deploying AI/ML models on edge devices—like Raspberry Pi or microcontrollers—enables real-time inference for applications such as localization, activity recognition, and object tracking. While training directly on the edge is feasible for simple models using efficient frameworks (e.g., scikit-learn, TensorFlow Lite), most complex models are trained in the cloud and optimized for deployment. Optimization techniques such as quantization, pruning, and distillation help shrink models to fit within memory and processing limits of edge hardware. This strategy balances the strengths of cloud-based computation with the speed and privacy advantages of local inference. Deployment workflows often rely on tools like ONNX, PyTorch Mobile, or Edge Impulse, paired with accelerators like Google's Coral USB or NVIDIA's Jetson Nano, to enhance inference performance.

Edge AI and TinyML are especially valuable for localization in environments with limited connectivity or power constraints. Applications include indoor tracking using WiFi RSSI or CSI, zone classification, asset monitoring, and adaptive positioning that integrates GNSS and UWB. On-device processing reduces latency and enhances data privacy, particularly for location-sensitive tasks. However, deploying ML at the edge is not without challenges: limited computational resources, energy constraints, fragmented toolchains, and privacy vulnerabilities demand robust engineering practices. Despite these hurdles, TinyML continues to expand with growing hardware support and community-driven toolchains, making it an increasingly practical approach for seamless, intelligent localization in real-world IoT systems.

6 ETHICAL CONCERNS OF AI/ML

Ethical AI and ML encompasses a broad range of principles aimed at ensuring that artificial intelligence technologies are developed and deployed in ways that are fair, accountable, transparent, and aligned with human values. As AI systems increasingly influence decision-making in critical domains, such as healthcare, finance, and criminal justice, it becomes essential to address issues such as algorithmic bias, data privacy, accountability, and explainability (I. Y. Chen, Joshi, and Ghassemi (2020); Lepri et al. (2021); Mehrabi, Morstatter, Saxena, Lerman, and Galstyan (2021)).

A core challenge in ethical AI/ML is ensuring fairness and addressing bias. ML models frequently reflect societal and historical biases embedded within their training data. To mitigate this, appropriate fairness definitions and strategies must be applied. Achieving fairness in AI/ML demands careful consideration of both technical and social factors (Mehrabi et al. (2021)).

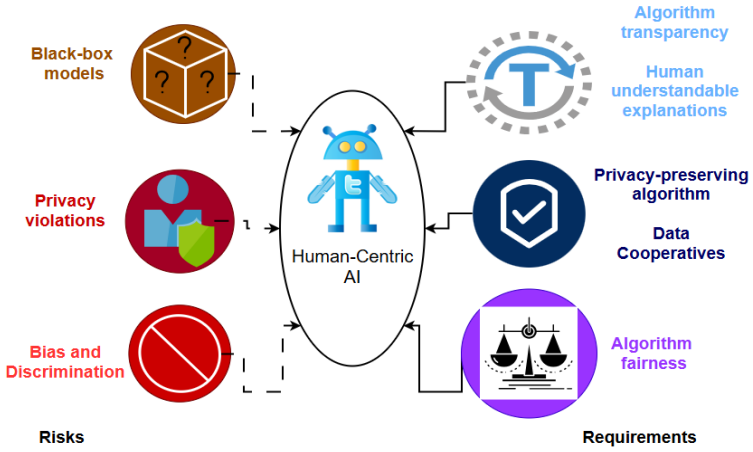


Figure 35. Human-centric AI, adapted from Lepri et al. (2021).

Figure 35 presents a comprehensive ethical framework that incorporates transparency, accountability, and human centric principles in the development of AI systems (Lepri et al. (2021)). The figure illustrates how existing risks (dashed lines on the left) should be addressed and replaced by essential requirements (solid lines on the right).

To ensure that AI systems contribute positively to society, it is important to embed ethical guidelines throughout the AI lifecycle, from problem formulation and data collection to deployment and monitoring. This includes promoting transparency, implementing fairness-aware algorithms, safeguarding user privacy, and establishing clear accountability for outcomes generated by AI systems.

6.1 European Union's Artificial Intelligence Act

The European Union's Artificial Intelligence Act (AI Act), enacted on 2 August 2024, represents a significant step toward embedding ethical principles throughout the AI lifecycle. The AI Act introduces a risk-based framework that categorizes AI systems into four levels: unacceptable risk, high risk, Specific transparency risk, and minimal risk. Unacceptable-risk applications, such as social scoring and manipulative biometric surveillance, are prohibited. High-risk systems, including those used in critical sectors such as healthcare and education, are subject to stringent requirements that encompass transparency, data governance, human oversight, and accountability. The Act provides a harmonized approach for all EU member states. This represents one of the most structured and comprehensive regulatory efforts to monitor AI usage to date.

This regulatory approach ensures that AI systems are developed and deployed in a manner that respects fundamental rights and societal values. By mandating comprehensive documentation and post-market monitoring, the AI Act fosters transparency and continuous evaluation of AI systems' impact. Furthermore, the Act's provisions extend to general-purpose AI models, requiring compliance with transparency obligations and, in some cases, additional assessments based on their capabilities. Through these measures, the EU aims to promote trustworthy AI that aligns with ethical standards and contributes positively to society (European Parliament (2024)).

A recent study highlighted the ethical risks and outlined future directions for building trust in the application of large language models (LLMs) within the framework of the EU AI Act. The study explored the evolving regulatory landscape shaped by the EU AI Act, highlighting a critical period where early adopters implementing these ethical measures can gain reputational and competitive advantages. The study also discussed the need for agile educational programs and organizational adaptation to meet future compliance requirements (D. Lee, Todorova, and Dehghani (2024)).

The first deadline for compliance with the EU AI Act will take effect as planned on 2 August 2025, applying to general-purpose AI models that have not yet been released to the market (European Commission (2025)).

6.2 Ethical concerns of AI/ML in localization applications

The integration of AI/ML and Large Language Models (LLMs) into localization applications such as GNSS navigation, geofencing, and location-based services raises significant ethical concerns, particularly with regard to user privacy and data protection. The following sections discuss some of the major ethical concerns related to AI/ML.

6.2.1 Privacy risks in localization data

Localization technologies inherently collect sensitive geospatial data, which, when processed by AI/ML systems, can reveal intricate details about an individual's habits, routines, and personal life. Even when data are anonymized, advanced inference techniques employed by LLMs can re-identify individuals by correlating location patterns with other available data sources. Staab et al. (2023) demonstrated that LLMs could infer personal attributes, such as location and income, with high precision from seemingly harmless text inputs, highlighting the potential for privacy breaches in localization contexts (Staab, Vero, Balunović, and Vechev (2024)).

6.2.2 Inference capabilities of LLMs

LLMs, trained on vast datasets, possess the capability to deduce sensitive information from minimal input. This means that even without explicit location data, these models can predict a user's whereabouts based on language patterns and contextual cues. Such capabilities pose ethical dilemmas, especially when users are unaware of the extent to which their data can be analyzed and inferred. The potential misuse of these inferences by malicious actors or unauthorized profiling underscores the need for stringent ethical guidelines (Staab et al. (2024)).

6.2.3 Data governance and regulatory compliance

The General Data Protection Regulation (GDPR) of the European Union emphasizes the protection of personal data, including location information. However, the dynamic nature of AI/ML systems challenges traditional data governance frameworks. Ensuring compliance requires continuous monitoring and updating of data

handling practices. Moreover, the EU AI Act categorizes AI systems based on risk, with high-risk systems, including those processing location data, subject to rigorous requirements to ensure transparency, accountability, and protection of user rights (European Parliament (2024)).

6.2.4 Mitigation strategies

To address these ethical concerns, developers and organizations should adopt privacy-preserving techniques, such as differential privacy and federated learning, which minimize data exposure. Implementing transparency measures, obtaining informed consent, and allowing users to control their data are crucial steps. In addition, regular audits and impact assessments can help identify and mitigate potential risks associated with AI-driven localization services.

In conclusion, while AI/ML and LLMs offer enhanced capabilities in localization applications, they also introduce complex ethical challenges. Balancing technological advancement with the imperative to protect individual privacy requires a multifaceted approach, integrating robust data governance, user-centric design, and adherence to evolving regulatory standards.

7 CONCLUSION

Precision positioning is essential for navigation-based services and location-aware applications that drive many Industry 4.0 operations, such as human navigation, smart logistics, industrial automation, and context-aware IoT systems. Depending on environmental complexity, user requirements, and technological capabilities, positioning systems vary in design and accuracy. Traditionally, achieving high precision has required expensive GNSS receivers, dense infrastructure, or complex sensor fusion.

This work explores the integration of machine learning (ML) techniques with low-cost GNSS devices to enhance positioning accuracy across outdoor, indoor, and transitional environments for seamless navigation. By leveraging ML's capability to learn complex patterns from GNSS, UWB, and IMU sensor data and using TinyML frameworks, lightweight yet powerful models capable of being deployed on edge devices were developed. These models address challenges such as signal noise, multi-path effects, and environmental interference commonly affecting low-cost systems. The thesis focuses on four key objectives: (1) improving outdoor positioning accuracy, (2) increasing indoor localization precision, (3) enabling seamless indoor-outdoor navigation through scenario detection, and (4) deploying optimized ML models directly on edge hardware. This approach offers cost-effective, portable, and energy-efficient solutions, reducing barriers to adoption in smart logistics and enabling scalable, privacy-preserving localization systems for industrial and consumer applications.

7.1 Main findings

The main findings of this doctoral thesis are listed as follows:

1. GNSS-enabled smartphones offer a cost-effective solution for precise outdoor positioning and, when equipped with UWB chipsets, can also support accurate indoor-outdoor localization even in challenging industrial environments.
2. Integrating GNSS with IMU significantly enhances outdoor positioning accuracy, while combining UWB with IMU leads to notable improvements in indoor localization performance.
3. Applying machine learning models for scenario detection and context awareness can significantly improve position estimation by enabling dynamic switching between localization methods based on environmental cues. This approach supports seamless navigation across indoor and outdoor environments.

4. Our enhancements in outdoor vehicular positioning for low-cost commercial-grade GNSS receivers in smartphones led to an average accuracy (MAE) of up to 5.62 meters, using GNSS only solutions, and the simple weighted average of Linear Regression (LR), Bayesian Ridge (BR) and Neural Network (NN) results. These results were validated in an international competition held by Google in 2021.

5. When using multi-sensor fusion, state-space estimation, and machine learning techniques, our enhancements in outdoor vehicular positioning for low-cost commercial grade GNSS receivers in smartphones led to an average accuracy (MAE) of up to 2.29 meters. These results were validated as achieving a Bronze medal rankings in the GNSS positioning estimations at an international competition held by Google in 2022.

6. The use of factor graph optimization (FGO) is a growing trend for position estimation. Our implementation in outdoor vehicular positioning for low-cost commercial grade GNSS receivers in smartphones led to an average accuracy (MAE) of up to 1.66 meters. This results supports the claim of previous studies that FGO provides better results in the integration of GNSS/IMU compared to KF-based integrations. These results were validated as achieving a Silver medal rankings in the GNSS positioning estimations at an international competition held by Google in 2023.

7. The use of GNSS for precise orbit determination (POD) of LEO satellites is essential for key space-based services like navigation, telecommunication, and Earth observation. Traditional methods often fall short in capturing the nonlinear dynamics of satellite motion. This study explored ML models as a data-driven alternative for improving POD accuracy in LEO satellites.

8. This study demonstrates the effectiveness of machine learning in enhancing orbit prediction accuracy. Among the evaluated models—Polynomial Regression (PR), k-Nearest Neighbors (KNR), Random Forest Regression (RFR), and Linear Regression (LR)—the PR model outperformed the others, achieving the lowest root mean square error (RMSE) in both position and velocity estimations for the MEO GPS satellite PRN-14.

9. Edge AI and TinyML can offer practical solutions for localization in low-connectivity or power-constrained environments by enabling on-device processing for tasks like indoor tracking, zone classification, and GNSS-UWB integration. While challenges such as limited resources, energy constraints, and privacy concerns exist, advancements in hardware and toolchains are making TinyML an increasingly viable option for efficient, privacy-aware localization in real-world IoT systems (Schizas et al. (2022)).

7.2 Scientific contributions

1. **Literature review of Machine Learning Utilization in GNSS Positioning**

   A comprehensive literature review was conducted in publications I., and IV. to explore the utilization of machine learning (ML) in GNSS-related use cases. The review discussed in Section 3.4.3 of Chapter 3 highlighted current trends among researchers, showing a growing interest in applying ML techniques to enhance positioning accuracy, mitigate signal degradation, and support robust localization across diverse environments. This review was designed to serve both as a step-by-step guide for GNSS/ML beginners and enthusiasts, and as a credible academic reference for experienced GNSS/ML researchers and practitioners. Both publications have recorded high research interest scores and citations on ResearchGate .

2. **Machine Learning for precise low-cost outdoor GNSS Positioning**

   The research presented in publications II. and III. focused on developing a cost-effective GNSS-based positioning system using commercial-grade receivers, such as those found in smartphones. By integrating ML techniques with GNSS/IMU sensor fusion, the system achieves enhanced positioning accuracy suitable for outdoor environments, including urban canyons. Publication II. applied ML techniques alongside a KF-based GNSS-only solution, while publication III. extended this by combining GNSS and IMU data using RTS smoothing. These approaches are detailed in Sections 3.6 and 3.7 of Chapter3. The method developed in publicationII., referred to as LC-GNSS/IMU/ML, incorporates a Machine Learning-based Adaptive Positioning (MAP) algorithm with RTK post-processing (MAP-PPK) to further enhance estimation accuracy. This work was presented, and validated through real-world experiments such as the Google Smartphone Decimeter Challenge 2022 where it earned a Bronze ranking for its contribution to precise GNSS solutions.

3. **Seamless Indoor-Outdoor Navigation**

   This work presented in publication V. focused on combining low-cost GNSS receivers (e.g., smartphone-grade) with UWB and ML techniques to achieve accurate and seamless positioning, particularly across indoor–outdoor transition zones. The approach supports real-time adaptability and cost-effective deployment for smart logistics and location-aware services. This is discussed in Section 3.7.2 of Chapter 3.

4. **Factor Graph Optimization (FGO) for GNSS/IMU**

   An alternative to Kalman Filter-based solutions is explored using FGO to achieve higher precision in low-cost systems, showing superior performance

in complex urban scenarios. This study, detailed in publication VI. and discussed in Section 3.8 of Chapter 3. The FGO-based method outperformed traditional KF integration, achieving superior positioning accuracy. Notably, this research earned a Silver ranking in the Google Decimeter Challenge 2023, highlighting its practical effectiveness among state-of-the-art GNSS solutions.

5. **ML-based Precise Orbit Determination (POD) of LEO Satellites**

   This work detailed in publication VII. contributes to the field of satellite navigation and space-based services by introducing and evaluating ML models as data-driven alternatives to traditional methods for POD of LEO satellites. This study discussed in Section 4 of Chapter 4 specifically addresses the limitations of conventional approaches in modeling the nonlinear dynamics of satellite motion. Through empirical evaluation, it demonstrates that the Polynomial Regression (PR) model significantly improves the accuracy of orbit prediction, outperforming other ML models such as k-Nearest Neighbors (KNN), Random Forest Regression (RFR), and Linear Regression (LR) in both position and velocity estimation metrics. This advancement underscores the potential of ML-based POD for enhancing the performance of space-based applications, including navigation, Earth observation, and telecommunications.

6. **TinyML for Edge Localization**

   This work discussed in publication VIII. highlights the feasibility and growing practicality of Edge AI and TinyML for localization tasks in constrained environments. It contributes to the field by demonstrating how on-device machine learning can support indoor tracking, zone classification, and GNSS/IMU/UWB integration, even under low-connectivity and power-limited conditions. Additionally, it addresses key challenges including resource limitations and privacy concerns while emphasizing the ongoing advancements in hardware and toolchains that are making TinyML a scalable and privacy-preserving solution for real-world IoT localization applications. This contribution advances the understanding of how lightweight ML can be effectively utilized in constrained edge environments for reliable and intelligent localization.

7. **Ethical Considerations in AI/ML Localization**

   The study in section 6 emphasizes the ethical challenges of AI in localization arising from the integration of AI/ML and Large Language Models (LLMs) into localization applications such as GNSS navigation, geofencing, and location-based services. Key issues include user privacy, data security, and the risk of misuse of sensitive location information. These challenges underscore the need for robust privacy-preserving techniques and transparent data governance. In light of the EU AI Act released in 2024, this topic

gains even greater relevance, reinforcing the importance of adopting a human-centric approach in the development and deployment of AI/ML technologies.

7.3 Novel Contributions

The novel contributions of this dissertation are organized into two primary methodological categories:

1. **Machine Learning (ML)-Based Methods**

   The application of ML techniques in this research resulted in significant improvements in positioning accuracy across different environments. Two contributions stand out as particularly novel:

   (a) **Enhanced Outdoor Positioning:** The proposed ML-based methods outperformed conventional GNSS-only positioning under certain conditions, especially in challenging environments such as urban canyons.

   (b) **Scenario Detection for Seamless Navigation:** A novel ML-based scenario detection model was developed, enabling seamless transitions between outdoor and indoor positioning systems. The system enables the adaptive switch between GNSS/IMU for outdoor environments and UWB/IMU for indoor settings, thus enhancing overall navigation continuity.

   These contributions address notable gaps in the existing literature by providing practical, adaptive, and cost-effective localization solutions.

2. **Factor Graph Optimization (FGO)-Based Methods**

   This dissertation also proposed and evaluated an alternative GNSS/IMU integration approach based on Factor Graph Optimization (FGO). The results demonstrated that the FGO-based method achieved a **27.5% improvement in positioning accuracy** compared to conventional Kalman Filter integration with RTS smoothing (KF-RTS).

   This finding highlights the potential of FGO as a robust, scalable solution for precise positioning using low-cost sensor configurations, particularly in complex environments such as urban canyons.

7.4 Limitations

While this research demonstrated promising results in the development of low-cost, precise positioning systems, several limitations were identified that present avenues for future work:

1. **Limited Availability of UWB-Enabled Smartphones:** The commercial availability of smartphones equipped with UWB chipsets remains limited. Consequently, for the experimental setup, it was necessary to rely on external UWB devices to evaluate the proposed seamless navigation solutions.

2. **Challenges in Real-Time Multi-Sensor Fusion:** Integrating GNSS, IMU, UWB, and other sensors for real-time applications is computationally demanding, particularly for systems operating at nanosecond-level precision. As the complexity of the multisensor scheme increases, so do processing delays, which can hinder real-time performance.

3. **GNSS Signal Degradation in Urban and Natural Environments:** Environments such as tree-lined roads and highways introduce significant signal shadowing and multipath effects, degrading GNSS signal quality. This poses a major challenge for applications requiring high-precision and reliable positioning, such as autonomous driving.

4. **Limited Availability of Indoor Mapping Data:** The lack of comprehensive and accessible indoor mapping resources presents a barrier to the straightforward implementation of indoor navigation solutions, especially for scalable deployments.

5. **FGO-Based Methods Limited to GNSS Pseudorange Measurements:** The FGO-based approach in this research was applied solely to GNSS pseudorange measurements. However, existing literature indicates that pseudorange-only solutions offer limited positioning accuracy compared to carrier-phase-based methods.

6. **TinyML Hardware Implementation Constraints:** Although the feasibility and advantages of TinyML for on-device positioning were discussed, practical implementation on low-cost hardware was not conducted due to time and resource constraints.

7.5 Future research

Building on the findings of this research, several areas merit further investigation:

1. **Carrier-Phase-Based FGO Integration:** Future work will explore the integration of GNSS carrier-phase measurements within the FGO framework to further enhance positioning accuracy, particularly in challenging environments.

2. **Experimental Validation of TinyML Deployments:** Practical implementation and testing of TinyML-based positioning models on low-cost embedded devices will be pursued to validate their performance and energy efficiency for real-world applications.

3. **Enhanced Multi-Sensor Fusion for Real-Time Applications:** Investigating optimized fusion schemes and efficient computational pipelines will help reduce processing delays and improve the reliability of real-time multi-sensor positioning systems.

4. **Integration with UWB-Enabled Smartphones:** As UWB-enabled smartphones become more widely available, future research will focus on fully integrating these devices with existing UWB infrastructure to develop seamless, portable indoor-outdoor positioning solutions.

5. **Development of Indoor Mapping Solutions:** Addressing the scarcity of indoor maps by leveraging crowdsourcing, vision-based SLAM, or infrastructure-supported mapping techniques will be essential for practical and scalable indoor navigation.

6. **Ethical and Privacy-Preserving Localization:** Further research is needed to design and evaluate privacy-preserving mechanisms for AI/ML-based localization, in alignment with emerging regulations such as the EU AI Act, to ensure human-centric, trustworthy positioning systems.

REFERENCES

Alabi, R. O. (2021). *Machine learning for personalized prognostication of tongue cancer* (Doctoral dissertation, University of Vaasa). (Doctoral dissertation) Retrieved from https://osuva.uwasa.fi/handle/10024/12272

Alarifi, A., Al-Salman, A., Alsaleh, M., Alnafessah, A., Al-Hadhrami, S., Al-Ammar, M., & Al-Khalifa, H. (2016, may). Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, *16*(5), 707. https://doi.org/10.3390/s16050707

Ali, J., & Ullah Baig Mirza, M. R. (2010). Performance comparison among some nonlinear filters for a low cost sins/gps integrated solution. *Nonlinear Dynamics*, *61*(3), 491–502.

Avellaneda, D., Mendez, D., & Fortino, G. (2023). A tinyml deep learning approach for indoor tracking of assets. *Sensors*, *23*(3). https://doi.org/10.3390/s23031542

Banbury, C., Reddi, V. J., Torelli, P., & et al. (2021). Benchmarking tinyml systems: Challenges and direction. *arXiv preprint arXiv:2102.07638*.

Banbury, C., Zhou, C.-F., Fedorov, I., & et al. (2021). Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers. *Proceedings of Machine Learning and Systems*, *3*, 517–532.

Bao, J., Li, D., Qiao, X., & Rauschenbach, T. (2020). Integrated navigation for autonomous underwater vehicles in aquaculture: A review. *Information processing in agriculture*, *7*(1), 139–151.

Bari, B. S., & Yelamarthi, K. (2025, April). Advancing federated learning: A comprehensive solution for model aggregation, heterogeneity, privacy, and security. *SN Computer Science*, *6*(411). Retrieved from https://link.springer.com/article/10.1007/s42979-025-03934-1

Bar-Shalom, Y. (1989). Recursive tracking algorithms: from the kalman filter to intelligent trackers for cluttered environment. In *Proceedings of the iccon ieee international conference on control and applications.* IEEE. https://doi.org/10.1109/iccon.1989.770605

Biagi, L., Grec, F. C., & Negretti, M. (2016). Low-cost gnss receivers for local monitoring: Experimental simulation, and analysis of displacements. *Sensors*, *16*(12). https://doi.org/10.3390/s16122140

Biswas, S. (2017). *Computationally efficient non-linear kalman filters for on-board space vehicle navigation* (Unpublished doctoral dissertation). The Univesity of New South Wales.

Bourdoux, A., Barreto, A. N., van Liempd, B., de Lima, C., Dardari, D., Belot, D., ... Miao, Y. (2020). 6g white paper on localization and sensing. *arXiv*. (arXiv preprint arXiv:2006.01779) Retrieved from https://arxiv.org/abs/2006.01779

Brum, D., Veronez, M. R., Menezes de Souza, E., Koch, I. E., Gonzaga, L., Klein, I., ... et al. (2019). A Proposed Earthquake Warning System Based on Ionospheric Anomalies Derived From GNSS Measurements and Artificial Neural Networks. In *Igarss 2019 - 2019 ieee international geoscience and remote sensing symposium* (p. 9295–9298). Retrieved from 10.1109/IGARSS.2019.8900197

Cao, S., Guo, L., & Chen, W. (2018). Anti-disturbance fault tolerant initial alignment for inertial navigation system subjected to multiple disturbances. *Aerospace Science and Technology*, *72*, 95–103.

Carlone, L., Tron, R., Daniilidis, K., & Dellaert, F. (2015, May). Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4597–4604). (ISSN: 1050-4729)

Carrera Villacrés, J. L., Zhao, Z., Braun, T., & Li, Z. (2019). A Particle Filter-Based Reinforcement Learning Approach for Reliable Wireless Indoor Positioning. *IEEE Journal on Selected Areas in Communications*, *37*(11), 2457–2473.

Chen, C.-S. (2017, January). A non-line-of-sight error mitigation method for location estimation. *International Journal of Distributed Sensor Networks*, *13*(1), 155014771668273. https://doi.org/10.1177/1550147716682739

Chen, D., & Gao, G. X. (2019). Probabilistic graphical fusion of lidar, gps, and 3d building maps for urban uav navigation. *Navigation*, *66*(1), 151–168.

Chen, I. Y., Joshi, S., & Ghassemi, M. (2020). Ethical machine learning in health care. *Annual Review of Biomedical Data Science*, *3*(1), 123–144.

Cheng, Y., & Zhou, T. (2019, August). UWB indoor positioning algorithm based on TDOA technology. In *2019 10th international conference on information technology in medicine and education (ITME).* IEEE. https://doi.org/10.1109/itme.2019.00177

Chow, A., Orendorff, D., Fu, M., Khider, M., Dane, S., & Gulati, V. (2023). *Google smartphone decimeter challenge 2023-2024.* https://kaggle.com/competitions/

smartphone-decimeter-2023. (Kaggle)

Cunningham, A., Paluri, M., & Dellaert, F. (2010, October). DDF-SAM: Fully distributed SLAM using Constrained Factor Graphs. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3025–3030). (ISSN: 2153-0866)

Dabove, P., & Pietra, V. D. (2022, November). Towards the use of low-cost GNSS receivers for permanent networks in precision agriculture. In *2022 IEEE workshop on metrology for agriculture and forestry (MetroAgriFor).* IEEE. https://doi.org/10.1109/metroagrifor55389.2022.9965163

Darbellay, G. A. (1999, November). An estimator of the mutual information based on a criterion for conditional independence. *Computational Statistics & Data Analysis*, *32*(1), 1–17. https://doi.org/10.1016/s0167-9473(99)00020-1

Dasgupta, S., Ghosh, T., & Rahman, M. (2021a). *A reinforcement learning approach for gnss spoofing attack detection of autonomous vehicles.*

Dasgupta, S., Ghosh, T., & Rahman, M. (2021b). *A reinforcement learning approach for gnss spoofing attack detection of autonomous vehicles.* https://doi.org/10.48550/arXiv.2108.08628

David, R., Warden, P., & et al. (2020). Tensorflow lite micro: Embedded machine learning on tinyml systems. *arXiv preprint arXiv:2010.08678*.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-1*(2), 224-227.

Dellaert, F., & Contributors, G. (2022, May). *borglab/gtsam.* Georgia Tech Borg Lab. https://doi.org/10.5281/zenodo.5794541

Dems, K. (2010). *What are almanac and ephemeris data?* https://www.brighthub.com/electronics/gps/articles/73766/. (Accessed: 2025-09-07)

Dutta, D. L., & Bharali, S. (2021). Tinyml meets iot: A comprehensive survey. *Internet of Things*, *16*, 100461. https://doi.org/https://doi.org/10.1016/j.iot.2021.100461

Dwivedi, S., Shreevastav, R., Munier, F., Nygren, J., Siomina, I., Lyazidi, Y., ... Gunnarsson, F. (2021, November). Positioning in 5g networks. *IEEE Communications Magazine*, *59*(11), 38–44. https://doi.org/10.1109/mcom.011.2100091

Elhanashi, A., Dini, P., Saponara, S., & Zheng, Q. (2024). Advancements

in tinyml: Applications, limitations, and impact on iot devices. *Electronics*, *13*(17), 3562. (Open-access review discussing TinyML's role in enabling real-time, low-latency edge intelligence for IoT, including outdoor deployments) https://doi.org/10.3390/electronics13173562

Elliott D., K., & Christopher J., H. (2017). *Understanding gps/gnss: Principles and applications*. Artech house.

Elsanhoury, M., Makela, P., Koljonen, J., Valisuo, P., Shamsuzzoha, A., Mantere, T., ... Kuusniemi, H. (2022). Precision positioning for smart logistics using ultra-wideband technology-based indoor navigation: a review. *IEEE Access*, 1–1. https://doi.org/10.1109/access.2022.3169267

El-Sheimy, N., & Youssef, A. (2020). Inertial sensors technologies for navigation applications: State of the art and future trends. *Satellite Navigation*, *1*(1), 1–21.

ESA. (2020). *Low earth orbit* (Tech. Rep.). European Space Agency. Retrieved from https://www.esa.int/ESA_Multimedia/Images/2020/03/Low_Earth_orbit

ESA. (2024). *European Space Agency, SWARM Data Access.* Retrieved from [2024-05-08]https://swarm-diss.eo.esa.int

Europa (Tech. Rep.). (2021). European Space Agency. Retrieved from https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo_HAS_Info_Note.pdf

European Commission. (2025). *General-purpose ai code of practice now available.* (Accessed: 2025-07-26) Retrieved from https://ec.europa.eu/commission/presscorner/detail/en/ip_25_1787

European GNSS Agency. (2019). *PPP-RTK market and technology report.* Retrieved from [2021-05-15]https://www.euspa.europa.eu/simplecount_pdf/tracker?file=calls_for_proposals/rd.03_-_ppp-rtk_market_and_technology_report.pdf

European GNSS Agency. (2020). *European GNSS Agency: GNSS User Technology Report 2020* (Tech. Rep.). Author. Retrieved from [2024-11-14]https://www.euspa.europa.eu/sites/default/files/uploads/technology_report_2020.pdf

European Parliament. (2024). *Eu ai act: First regulation on artificial intelligence.* https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence. (Accessed: 2025-05-28)

European Space Agency. (2011). *Galileo signal plan* (Tech. Rep.). European Space Agency. Retrieved from [2024-11-13]https://gssc.esa.int/navipedia/index.php/Galileo_Signal_Plan

European Space Agency. (2018). *BeiDou general introduction* (Tech. Rep.). European Space Agency. Retrieved from [2024-11-13]https://gssc.esa.int/navipedia/index.php/BeiDou_General_Introduction

European Space Agency. (2019). *Glonass signal plan* (Tech. Rep.). Author. Retrieved from [2024-11-13]https://gssc.esa.int/navipedia/index.php/GLONASS_Signal_Plan

European Space Agency. (2021). *Galileo space segment* (Tech. Rep.). European Space Agency. Retrieved from [2024-11-13]https://gssc.esa.int/navipedia/index.php/Galileo_Space_Segment

European Union Agency for the Space Programme. (2020). *Galileo High Accuracy Service (HAS)* (Tech. Rep.). European GNSS Agency. Retrieved from [2024-11-14]https://www.gsc-europa.eu/galileo/services/galileo-high-accuracy-service-has

EUSPA. (2022). *Galileo services* (Tech. Rep.). European Union Agency for Space Programme. Retrieved from [2024-11-13]https://www.euspa.europa.eu/galileo/services

Falco, G., Pini, M., & Marucco, G. (2017, January). Loose and tight GNSS/INS integrations: Comparison of performance assessed in real urban scenarios. *Sensors*, *17*(2), 255. https://doi.org/10.3390/s17020255

Faragher, R. (2012, September). Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]. *IEEE Signal Processing Magazine*, *29*(5), 128–132. https://doi.org/10.1109/msp.2012.2203621

Farrell, J. A., & Barth, M. (1999). *The global positioning system and inertial navigation*. McGraw-Hill Professional.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, *27*(8), 861-874. (ROC Analysis in Pattern Recognition) https://doi.org/https://doi.org/10.1016/j.patrec.2005.10.010

Ficco, M., Guerriero, A., Milite, E., Palmieri, F., Pietrantuono, R., & Russo, S. (2024). Federated learning for iot devices: Enhancing tinyml with on-board training. *Information Fusion*, *104*, 102189. https://doi.org/10.1016/j.inffus.2023.102189

Filho, C. P., Marques, E., Chang, V., dos Santos, L., Bernardini, F., Pires, P. F., ... Delicato, F. C. (2022). A systematic literature review on distributed machine learning in edge computing. *Sensors*, *22*(7). https://doi.org/10.3390/s22072665

Forster, C., Carlone, L., Dellaert, F., & Scaramuzza, D. (2015). *Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation*. Georgia Tech Borg Lab.

Friis-Christensen, E., Lühr, H., & Hulot, G. (2006). Swarm: A constellation to study the earth's magnetic field. *Earth, Planets and Space*, *58*, 351-358. Retrieved from https://doi.org/10.1186/BF03351933

Gavin, H. P. (2019). The levenberg-marquardt algorithm for nonlinear least squares curve-fitting problems. *Department of Civil and Environmental Engineering Duke University August*, *3*.

Gebre-Egziabher, D., & Gleason, S. (2009). *GNSS applications and methods*. Artech House, ESA Bulletin Nr. 90.

Gite, H. R., Solankar, M. M., Surase, R. R., & Kale, K. V. (2019). Comparative study of dimensionality reduction techniques for hyperspectral data. In *Information and communication technology for intelligent systems* (pp. 1–10). Springer.

Google. (2021). *Google smartphone decimeter challenge.* (Accessed: 2025-01-06) Retrieved from https://www.kaggle.com/competitions/google-smartphone-decimeter-challenge

Google. (2022). *Google smartphone decimeter challenge 2022.* (Accessed: 2025-01-06) Retrieved from https://www.kaggle.com/competitions/smartphone-decimeter-2022

Google. (2023). *Google smartphone decimeter challenge 2023-2024.* (Accessed: 2025-01-06) Retrieved from https://www.kaggle.com/competitions/smartphone-decimeter-2023

Grewal, M. S., Weill, L. R., & Andrews, A. P. (2007). *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons.

Grimm, D. (2008). Gnss direction finding. *Geodezija ir kartografija*, *34*(3), 100–102.

Groves, P. D., Pulford, G. W., Littlefield, C. A., Nash, D. L., & Mather, C. J. (2007). Inertial navigation versus pedestrian dead reckoning: Optimizing the integration. In *Proceedings of the 20th international technical meeting of the satellite division of the institute of navigation (ion gnss 2007)* (pp. 2043–2055).

Guo, G., Chen, R., Ye, F., Liu, Z., Xu, S., Huang, L., … Qian, L. (2022, September). A robust integration platform of wi-fi rtt, rss signal, and mems-imu for locating

commercial smartphone indoors. *IEEE Internet of Things Journal*, *9*(17), 16322–16331. https://doi.org/10.1109/jiot.2022.3150958

Guo, X., & Ansari, N. (2020). Indoor positioning systems: A survey. *IEEE Internet of Things Journal*, *7*(5), 4318–4337. https://doi.org/10.1109/jiot.2020.2966999

Guo, X., Ansari, N., Hu, F., Shao, Y., Elikplim, N. R., & Li, L. (2020a). A survey on fusion-based indoor positioning. *IEEE Communications Surveys & Tutorials*, *22*(1), 566–594. https://doi.org/10.1109/comst.2019.2951036

Guo, X., Ansari, N., Hu, F., Shao, Y., Elikplim, N. R., & Li, L. (2020b). A survey on fusion-based indoor positioning. *IEEE Communications Surveys and Tutorials*, *22*(1), 566-594.

Gutierrez, J., Gilabert, R., Dill, E., Hernandez, G., Kaeli, D., & Closas, P. (2024). Multipath mitigation via clustering for position estimation refinement in urban environments. In *Pacific pnt.* NASA Technical Reports.

Guvenc, I., Chong, C.-C., & Watanabe, F. (2007). NLOS identification and mitigation for UWB localization systems. In *2007 IEEE wireless communications and networking conference.* IEEE. https://doi.org/10.1109/wcnc.2007.296

Hamza, V., Stopar, B., & Sterle, O. (2021, March). Testing the performance of multi-frequency low-cost GNSS receivers and antennas. *Sensors*, *21*(6), 2029. https://doi.org/10.3390/s21062029

Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International conference on learning representations (iclr).*

Hartikainen, J., Solin, A., & Särkkä, S. (2011). *Optimal filtering with Kalman filters and smoothers - a Manual for Matlab toolbox EKF/UKF*. Aalto University.

Heikkilä, M. A. (2025, March). *On using secure aggregation in differentially private federated learning with multiple local steps.* Retrieved from https://arxiv.org/html/2407.19286v2

Heublein, L., Feigl, T., Nowak, T., Rügamer, A., Mutschler, C., & Ott, F. (2025). Evaluating ml robustness in gnss interference classification, characterization & localization. In *Proc. ieee icl-gnss.*

Hexagon. (2021). *Chapter 5 - resolving errors: Multi-constellation and multi-frequency* (Tech. Rep.). Author. Retrieved from [2024-11-14]https://novatel.com/an-introduction-to-gnss/resolving-errors/multi-frequency-multi-constellation

Howard, A., Chow, A., Julian, B., Orendorff, D., Fu, M., Khider, M., … Dane, S. (2022). *Google smartphone decimeter challenge 2022.* https://kaggle.com/competitions/smartphone-decimeter-2022. (Kaggle)

Hsu, L.-T. (2017). GNSS multipath detection using a machine learning approach.. Retrieved from 10.1109/ITSC.2017.8317700

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*(1), 193–218. https://doi.org/10.1007/BF01908075

Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: principles and practice. *Monash University, Australia*. Retrieved from https://otexts.com/fpp3/arima.html

I, L. M., Ratnam, D. V., Raman, S., & Sivavaraprasad, G. (2020). Machine learning algorithm to forecast ionospheric time delays using Global Navigation satellite system observations. *Acta Astronautica*, *173*. Retrieved from 10.1016/j.actaastro.2020.04.048

Indelman, V., Williams, S., Kaess, M., & Dellaert, F. (2012). Factor graph based incremental smoothing in inertial navigation systems. In *2012 15th international conference on information fusion* (pp. 2154–2161).

Internet of business. (2022). *Google's DeepMind AI is learning to navigate cities without a map.* (https://internetofbusiness.com/deepmind-ai-learning-to-navigate-without-map/, Last accessed on 2022-07-04)

Jackson, J., Davis, B., & Gebre-Egziabher, D. (2018, April). A performance assessment of low-cost RTK GNSS receivers. In *2018 IEEE/ION position, location and navigation symposium (PLANS).* IEEE. https://doi.org/10.1109/plans.2018.8373438

Jia, H., Yang, Z., & Li, B. (2024). Roti-based statistical regression models for gnss precise point positioning errors associated with ionospheric plasma irregularities. *GPS Solutions*, *28*, 105.

Jiang, H., Li, J., Zhao, P., Zeng, F., Xiao, Z., & Iyengar, A. (2021, jan). Location privacy-preserving mechanisms in location-based services. *ACM Computing Surveys*, *54*(1), 1–36. https://doi.org/10.1145/3423165

Jiang, M., Ye, Z., Xiao, Y., & Gou, X. (2024). *Federated transfer learning aided interference classification in gnss signals.*

Jiang, W., Cao, Z., Cai, B., Li, B., & Wang, J. (2021, October). Indoor and outdoor

seamless positioning method using UWB enhanced multi-sensor tightly-coupled integration. *IEEE Transactions on Vehicular Technology*, *70*(10), 10633–10645. https://doi.org/10.1109/tvt.2021.3110325

Jolliffe, I. T. (1986). *Principal component analysis*. Springer.

Jwo, D. J., Biswal, A., & Mir, I. A. (2023a). Artificial neural networks for navigation systems: A review of recent research. *Applied Sciences*, *13*(7), 4475.

Jwo, D.-J., Biswal, A., & Mir, I. A. (2023b). Artificial neural networks for navigation systems: A review of recent research. *Applied Sciences*, *13*(7), 4475. https://doi.org/10.3390/app13074475

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, *31*(2), 216–235.

Kaggle. (2022). *Smartphone Decimeter 2022 Challenge Leaderboard.* (Accessed: January 8, 2025) Retrieved from https://www.kaggle.com/competitions/smartphone-decimeter-2022/leaderboard

Kaggle. (2023). *Smartphone Decimeter 2023-2024 Challenge Leaderboard.* (Accessed: January 8, 2025) Retrieved from https://www.kaggle.com/competitions/smartphone-decimeter-2023/leaderboard

Kalman, R. E. (1960, March). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45. https://doi.org/10.1115/1.3662552

Kanhere, A. V., Gupta, S., Shetty, A., & Gao, G. (2022). *Improving gnss positioning using neural network-based corrections.* https://doi.org/10.48550/arXiv.2110.09581

Keating, R., Säily, M., Hulkkonen, J., & Karjalainen, J. (2019). Overview of positioning in 5G new radio. In *2019 16th international symposium on wireless communication systems (ISWCS)* (p. 320-324).

Khodjaev, J., Park, Y., & Malik, A. S. (2009, aug). Survey of nlos identification and error mitigation problems in UWB-based positioning algorithms for dense environments. *annals of telecommunications - annales des télécommunications*, *65*(5-6), 301–311. https://doi.org/10.1007/s12243-009-0124-z

Kiani, M. (2020). *On the suitability of generalized regression neural networks for gnss position time series prediction.*

Kitchenham, B. (2004, 08). Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, *33*. Retrieved from https://www.researchgate.net/publication/228756057_Procedures_for_Performing_Systematic_Reviews

Klos, A., Bogusz, J., Bos, M., & Gruszczynska, M. (2019, 08). *Modelling the gnss time series: Different approaches to extract seasonal signals.*

Kondaveeti, S., Ratnam, D. V., & Sivavaraprasad, G. (2022). *Support vector regression model to predict tec for gnss signals.* Retrieved from https://link.springer.com/article/10.1007/s11600-022-00954-w

Lawrence, D., Cobb, H. S., Gutt, G., O'Connor, M., G.R.Reid, T., Walter, T., & Whelan, D. (2017). *Navigation from LEO: Current capability and future promise* (Tech. Rep.). GPSworld. Retrieved from [2024-11-14]https://web.stanford.edu/group/scpnt/gpslab/pubs/papers/Lawrence_GPSWorld_July2017.pdf

Lee, D., Todorova, C., & Dehghani, A. (2024, 12). Ethical risks and future direction in building trust for large language models application under the eu ai act. In (p. 41-46).

Lee, J.-Y. (2021). *Google smartphone decimeter eda + keras (tpu).* Retrieved from https://www.kaggle.com/jeongyoonlee/google-smartphone-decimeter-eda-keras-tpu

Lei, M., Jin, M., Huang, T., Guo, Z., Wang, Q., Wu, Z., ... Zhang, J. (2020, oct). Ultra-wideband Fingerprinting Positioning Based on Convolutional Neural Network. In *2020 international conference on computer information and telecommunication systems (CITS).* IEEE. https://doi.org/10.1109/cits49457.2020.9232628

Lepri, B., Oliver, N., & Pentland, A. (2021). Ethical machines: The human-centric use of artificial intelligence. *iScience*, *24*(3), 102249. https://doi.org/https://doi.org/10.1016/j.isci.2021.102249

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, *2*(2), 164–168.

Li, L., Elhajj, M., Feng, Y., & Ochieng, W. Y. (2023a). Machine learning based gnss signal classification and weighting scheme design in the built environment. *Satellite Navigation*, *4*(12).

Li, L., Elhajj, M., Feng, Y., & Ochieng, W. Y. (2023b). Machine learning based gnss signal classification and weighting scheme design in the built environment: a comparative experiment. *Satellite Navigation*, *4*, 12. https://doi.org/10.1186/s43020-023-00101-w

Li, L., Guo, X., Ansari, N., & Li, H. (2019). A hybrid fingerprint quality evaluation model for wifi localization. *IEEE Internet of Things Journal*, *6*(6), 9829–9840.

Li, M., Huang, G., Wang, L., & Xie, W. (2022, 10). *Comprehensive classification assessment of gnss observation data quality by fusing k- means and knn algorithms.*

LI, P., XU, Y., SHEN, T., & BI, S. (2019, August). INS/UWB integrated AGV localization employing kalman filter for indoor LOS/NLOS mixed environment. In *2019 international conference on advanced mechatronic systems (ICAMechS).* IEEE. https://doi.org/10.1109/icamechs.2019.8861620

Li, Q., Zhang, L., & Wang, X. (2021). Loosely coupled gnss/ins integration based on factor graph and aided by arima model. *IEEE Sensors Journal*, *21*(21), 24379-24387.

Li, X., Deng, Z. D., Rauchenstein, L. T., & Carlson, T. J. (2016, April). Contributed review: Source-localization algorithms and applications using time of arrival and time difference of arrival measurements. *Review of Scientific Instruments*, *87*(4), 041502. https://doi.org/10.1063/1.4947001

Lin, J., Chen, W.-M., & et al. (2020). Mcunet: Tiny deep learning on iot devices. In *Advances in neural information processing systems* (Vol. 33, pp. 11711–11722).

Linty, N., Farasin, A., Favenza, A., & Dovis, F. (2019). Detection of GNSS Ionospheric Scintillations Based on Machine Learning Decision Tree. *IEEE Transactions on Aerospace and Electronic Systems*, *55*(1), 303–317. Retrieved from 10.1109/TAES.2018.2850385

Liu, B., Zhou, W., Zhu, T., Gao, L., & Xiang, Y. (2018). Location privacy and its applications: A systematic study. *IEEE Access*, *6*, 17606–17624. https://doi.org/10.1109/access.2018.2822260

Liu, J., Pu, J., Sun, L., & He, Z. (2019a). An approach to robust ins/uwb integrated positioning for autonomous indoor mobile robots. *Sensors*, *19*(4), 950.

Liu, J., Pu, J., Sun, L., & He, Z. (2019b, feb). An Approach to Robust INS/UWB Integrated Positioning for Autonomous Indoor Mobile Robots. *Sensors*, *19*(4), 950. https://doi.org/10.3390/s19040950

Liu, S., Li, S., Zheng, J., & Fu, Q. (2022). Benefits of using carrier phase tracking errors for ultratight gnss/ins integration: Precise velocity estimation and fine imu calibration. *IEEE Transactions on Instrumentation and Measurement*, *71*, 1-12.

Liu, W., Li, J., Zeng, Q., Guo, F., Wu, R., & Zhang, X. (2019). An improved robust Kalman filtering strategy for GNSS kinematic positioning considering small cycle slips. *Advances in Space Research*, *63*(9), 2724–2734. https://doi.org/10.1016/j.asr.2017.11.041

Liu, Y., Morton, Y. J., & Jiao, Y. (2018). Application of machine learning to the characterization of GPS L1 ionospheric amplitude scintillation. In *2018 ieee/ion position, location and navigation symposium (plans)* (p. 1159–1166). Retrieved from 10.1109/PLANS.2018.8373500

Liu, Z., Lo, S., & Walter, T. (2021). *Gnss interference detection using machine learning algorithms on ads-b data.* ION GNSS+. Retrieved from https://web.stanford.edu/group/scpnt/gpslab/pubs/papers/Liu_IONGNSS2021_ADS_B.pdf

Louis, A. (2019). Neural Network Based Evil Waveforms Detection.. Retrieved from 10.23919/RFI48793.2019.9111769

Luo, J., Yin, Z., & Gui, L. (2024). A gnss uwb tight coupling and imu eskf algorithm for indoor and outdoor mixed scenario. *Cluster Computing*, *27*, 4855–4865. https://doi.org/10.1007/s10586-023-04208-2

Mahato, S., Santra, A., Dan, S., Rakshit, P., Banerjee, P., & Bose, A. (2019, March). Preliminary results on the performance of cost-effective GNSS receivers for RTK. In *2019 URSI asia-pacific radio science conference (AP-RASC).* IEEE. https://doi.org/10.23919/ursiap-rasc.2019.8738736

Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, *27*, 504-518. https://doi.org/https://doi.org/10.1016/j.asoc.2014.11.023

Manjunath, H., Heublein, L., Feigl, T., & Ott, F. (2025). Multimodal-to-text prompt engineering in llms for gnss interference characterization. In *Proc. ieee wcnc.*

Manoj Joshi. (2022). *Google uses Deep Learning with Street View to update its maps.* (https://www.geospatialworld.net/blogs/google-uses-deep-learning-with-street-view-to-update-its-maps/, Last accessed on 2022-07-04)

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, *11*(2), 431–441.

Marthala, R. V. R. (2020). *Wifi based indoor positioning - a machine learning approach* (Master's thesis, University of Vaasa). Retrieved from [2021-08-26]https://osuva.uwasa.fi/handle/10024/11126

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the international conference on artificial intelligence and statistics (aistats).*

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, *54*(6), 1–35.

Mendez, D., Zennaro, M., Altayeb, M., & Manzoni, P. (2024). On tinyml wifi fingerprinting-based indoor localization: Comparing rssi vs. csi utilization. In *2024 ieee 21st consumer communications and networking conference (ccnc)* (p. 1-6).

Miller, L. E. (2003). *Why uwb? a review of ultrawideband technology, report to netex project office, darpa* (Tech. Rep.). National Institute of Standards and Technology (NIST) U.S. Department of Commerce. Retrieved from http://citeseerx .ist.psu.edu/viewdoc/download?doi=10.1.1.76.7051&rep=rep1&type=pdf

Minetto, A., Gurrieri, A., & Dovis, F. (2020). A Cognitive Particle Filter for Collaborative DGNSS Positioning. *IEEE Access*, *8*, 194765–194779.

Mittal, S. (2019). A survey on model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1910.02272*.

Mohamed, S. A., Haghbayan, M.-H., Westerlund, T., Heikkonen, J., Tenhunen, H., & Plosila, J. (2019). A survey on odometry for autonomous navigation systems. *IEEE Access*, *7*, 97466–97486.

Mohanty, A., & Gao, G. (2024a). *A survey of machine learning techniques for improving global navigation satellite systems.* Retrieved from https://arxiv.org/pdf/ 2406.16873

Mohanty, A., & Gao, G. (2024b). A survey of machine learning techniques for improving gnss. *EURASIP Journal on Advances in Signal Processing*, *2024*(73).

Mostafa, M., Zahran, S., Moussa, A., El-Sheimy, N., & Sesay, A. (2018). Radar and visual odometry integrated system aided navigation for uavs in gnss denied environment. *Sensors*, *18*(9), 2776.

Munin, E., Blais, A., & Couellan, N. (2020). Convolutional Neural Network for Multipath Detection in GNSS Receivers.. Retrieved from 10.1109/AIDA-AT48540 .2020.9049188

Murshed, M., Murphy, C., D.Hou, Khan, N., Ananthanarayanan, G., & Hussain, F.

(2021). Machine learning at the network edge: A survey. *arxiv.org*. Retrieved from https://arxiv.org/pdf/1908.00080

Mäkelä, P. (2008). *Local positioning systems and indoor navigation* (Master's thesis, Tampere University of Technology). Retrieved from http://urn.fi/URN:NBN:fi:amk-2014121619934

Nekoogar, F. (2005). *Ultra-wideband communications: fundamentals and applications*. Prentice Hall PTR.

Newson, P., & Krumm, J. (2009). Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th acm sigspatial gis* (pp. 336–343). Retrieved from http://dx.doi.org/10.1145/1653771.1653818

NGA Public Affairs. (2023). *National Geospatial-Intelligence Agency, Office of Geomatics.* Retrieved from [2023-06-15]https://earth-info.nga.mil/index.php?action=home

NOAA. (2022). *The Global Positioning System* (Tech. Rep.). National Coordination Office for Space-Based Positioning, Navigation, and Timing. Retrieved from [2024-11-13]https://www.gps.gov/systems/gps/

Novatel. (2025). *Gnss ephemerides and almanacs.* https://shop.novatel.com/s/contactsupport/article/GNSS-Ephemerides-and-Almanacs. (Accessed: 2025-09-07)

Orendorff, D., van Diggelen, F., Elliott, J., Fu, M., Khider, M., & Dane, S. (2021). *Google smartphone decimeter challenge.* https://kaggle.com/competitions/google-smartphone-decimeter-challenge. (Kaggle)

Orus Perez, R. (2019). Using TensorFlow-based Neural Network to estimate GNSS single frequency ionospheric delay (IONONet). *Advances in Space Research*, *63*(5), 1607–1618. Retrieved from http://www.sciencedirect.com/science/article/pii/S0273117718308652

Ozarpaci, S., Kilic, B., Bayrak, O. C., Taskiran, M., Dogan, U., & Floyd, M. (2023). Machine learning approach for gnss geodetic velocity estimation. *GPS Solutions*. Retrieved from http://dx.doi.org/10.1007/s10291-023-01607-1

Park, J., Nam, S., Choi, H., Ko, Y., & Ko, Y.-B. (2020, oct). Improving Deep Learning-Based UWB LOS/NLOS Identification with Transfer Learning: An Empirical Approach. *Electronics*, *9*(10), 1714. https://doi.org/10.3390/electronics9101714

Pearl, J. (1987, October). Distributed revision of composite beliefs. *Artificial Intelligence*, *33*(2), 173–215. https://doi.org/10.1016/0004-3702(87)90034-8

Pfeifer, T., & Protzel, P. (2019a). Expectation-maximization for adaptive mixture models in graph optimization. In *2019 international conference on robotics and automation (icra)* (p. 3151-3157).

Pfeifer, T., & Protzel, P. (2019b). Expectation-maximization for adaptive mixture models in graph optimization. In *2019 international conference on robotics and automation (icra)* (pp. 3151–3157).

Poulose, A., & Han, D. S. (2020, sep). UWB Indoor Localization Using Deep Learning LSTM Networks. *Applied Sciences*, *10*(18), 6290. https://doi.org/10.3390/app10186290

Powers, D., & Ailab. (2011, 01). Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation. *J. Mach. Learn. Technol*, *2*, 2229-3981.

Premsankar, G., Di Francesco, M., & Taleb, T. (2018). Edge computing for the internet of things: A case study. *IEEE Internet of Things Journal*, *5*(2), 1275–1284.

Prol, F. S., Morales Ferre, R., Saleem, Z., Välisuo, P., Pinell, C., Lohan, E. S., ... Kuusniemi, H. (2022). Position, Navigation, and Timing (PNT) through Low Earth Orbit (LEO) Satellites: A Survey on Current Status, Challenges, and Opportunities. *IEEE Access*, 1–1. (Conference Name: IEEE Access)

Qi, X., Xu, B., Wang, Z., & Hsu, L.-T. (2024). Random forest-based multipath parameter estimation. *GPS Solutions*, *28*, 126. (Open preprint available) Retrieved from https://doi.org/10.1007/s10291-024-01667-x

Quan, Y., Lau, L., Roberts, G. W., Meng, X., & Zhang, C. (2018). Convolutional neural network based multipath detection method for static and kinematic gps high precision positioning. *Remote Sensing*, *10*(12). Retrieved from https://www.mdpi.com/2072-4292/10/12/2052

Ramswaroop. (2021). *Baseline appraoch - linear regression.* Retrieved from https://www.kaggle.com/ramswaroopbhakar14/baseline-appraoch-linear-regression

Rana, S. P., Dey, M., Siddiqui, H. U., Tiberi, G., Ghavami, M., & Dudley, S. (2017, sep). UWB localization employing supervised learning method. In *2017 IEEE 17th international conference on ubiquitous wireless broadband (ICUWB).* IEEE. https://doi.org/10.1109/icuwb.2017.8250971

Ray, P. P. (2022). A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, *34*(4), 1595-1623. https://doi.org/https://doi.org/10.1016/j.jksuci.2021.11.019

Ren, H., Anicic, D., & Runkler, T. A. (2023). Tinyreptile: Tinyml with federated meta-learning. In *International joint conference on neural networks (ijcnn).* https://doi.org/10.48550/arXiv.2304.05201

Repole, G. (2024). *Tinyml-enabled human indoor localization with uwb-radar imaging* (Master's thesis, Politecnico di Milano). (Master's thesis, discusses TinyML for adaptive indoor localization using UWB and lightweight models) Retrieved from https://www.politesi.polimi.it/handle/10589/219456

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53-65. https://doi.org/https://doi.org/10.1016/0377-0427(87)90125-7

Ryu, J. H., Gankhuyag, G., & Chong, K. T. (2016). Navigation system heading and position accuracy improvement through gps and ins data fusion. *Journal of Sensors*, *2016*.

Sambu, P., & Won, M. (2022, April). An experimental study on direction finding of bluetooth 5.1: Indoor vs outdoor. In *2022 IEEE wireless communications and networking conference (WCNC).* IEEE. https://doi.org/10.1109/wcnc51071.2022.9771930

Santhanam, M. (2011). *Uwb technology and its applications – a survey* (Unpublished master's thesis). School of Engineering, Jönköping University.

Sartayeva, Y., & Chan, H. C. B. (2023, aug). A survey on indoor positioning security and privacy. *Computers & Security*, *131*, 103293. https://doi.org/10.1016/j.cose.2023.103293

Savolainen, O. (2022). *Detecting anomalies in gnss signals with complex-valued lstm networks* (Unpublished master's thesis). University of Helsinki.

Schalk, G., & Leuthardt, E. C. (2011). Brain-computer interfaces using electrocorticographic signals. *IEEE Reviews in Biomedical Engineering*, *4*, 140–154. https://doi.org/10.1109/rbme.2011.2172408

Schizas, N., Karras, A., Karras, C., & Sioutas, S. (2022). Tinyml for ultra-low power ai and large scale iot deployments: A systematic review. *Future Internet*, *14*(12), 363. https://doi.org/10.3390/fi14120363

Selvan, K., Siemuri, A., Prol, F. S., Välisuo, P., Bhuiyan, M. Z. H., & Kuusniemi, H. (2023). Precise orbit determination of leo satellites: a systematic review. *GPS Solutions*, *27*(4), 178. https://doi.org/10.1007/s10291-023-01520-7

Semanjski, S., Muls, A., Semanjski, I., & De Wilde, W. (2019). Use and validation of supervised machine learning approach for detection of gnss signal spoofing. In *2019 international conference on localization and gnss (icl-gnss)* (p. 1-6). Retrieved from 10.1109/ICL-GNSS.2019.8752775

Shabnam, M., Chowdhury, I. H., Tushar, Z. H., Sultana, S., & Hossam-E-Haider, M. (2017, February). Performance evaluation of GNSS receiver in multi-constellation system. In *2017 international conference on electrical, computer and communication engineering (ECCE).* IEEE. https://doi.org/10.1109/ecace.2017.7912977

Shahbazian, R., Macrina, G., Scalzo, E., & Guerriero, F. (2023). Machine learning assists iot localization: A review of current challenges and future trends. *Sensors*, *23*(7). https://doi.org/10.3390/s23073551

Shamohammadi, O., Pahlavani, P., & Sharifi, M. A. (2023). Comparison of gradient boosting, logistic regression, and linear svc in classifying travel modes based on gnss data. In *Isprs annals* (Vol. XLVIII-4/W2-2022, pp. 103–108). https://doi.org/10.5194/isprs-archives-XLVIII-4-W2-2022-103-2023

Shen, G., Zetik, R., & Thoma, R. S. (2008, March). Performance comparison of TOA and TDOA based location estimation algorithms in LOS environment. In *2008 5th workshop on positioning, navigation and communication.* IEEE. https://doi.org/10.1109/wpnc.2008.4510359

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646.

Siemuri, A., Ahmadi, E., Elsanhoury, M., Selvan, K., Välisuo, P., Kuusniemi, H., & Elmusrati, M. S. (2024, September). Optimal robust positioning using factor graph. In *Proceedings of the 37th international technical meeting of the satellite division of the institute of navigation (ion gnss+ 2024)* (pp. 2684–2690). Baltimore, Maryland. https://doi.org/10.33012/2024.19909

Siemuri, A., & Elmusrati, M. (2025). *Federated learning and tinyml for localization: Benefits, applications, and challenges.* https://www.researchgate.net/publication/396849725_Federated_Learning_and_TinyML_for_Localization_Benefits_Applications_and_Challenges. (Accessed October 2025)

Siemuri, A., Elsanhoury, M., Selvan, K., Välisuo, P., Kuusniemi, H., & Elmusrati, M. S. (2023, September). Seamless navigation for indoor-outdoor positioning using

gnss-aided uwb/wifi/imu system. In *Proceedings of the 36th international technical meeting of the satellite division of the institute of navigation (ion gnss+ 2023)* (pp. 2616–2623). Denver, Colorado. https://doi.org/10.33012/2023.19323

Siemuri, A., Elsanhoury, M., Välisuo, P., Kuusniemi, H., & Elmusrati, M. S. (2022, September). Application of machine learning to gnss/imu integration for high precision positioning on smartphones. In *Proceedings of the 35th international technical meeting of the satellite division of the institute of navigation (ion gnss+ 2022)* (pp. 2256–2264). Denver, Colorado. https://doi.org/10.33012/2022.18375

Siemuri, A., Kuusniemi, H., Elmusrati, M. S., Välisuo, P., & Shamsuzzoha, A. (2021). Machine learning utilization in gnss—use cases, challenges and future applications. In *2021 international conference on localization and gnss (icl-gnss)* (p. 1-6).

Siemuri, A., Selvan, K., Kuusniemi, H., Välisuo, P., & Elmusrati, M. S. (2021, September). Improving precision gnss positioning and navigation accuracy on smartphones using machine learning. In *Proceedings of the 34th international technical meeting of the satellite division of the institute of navigation (ion gnss+ 2021)* (pp. 3081–3093). St. Louis, Missouri. https://doi.org/10.33012/2021.18004

Siemuri, A., Selvan, K., Kuusniemi, H., Valisuo, P., & Elmusrati, M. S. (2022). A systematic review of machine learning techniques for gnss use cases. *IEEE Transactions on Aerospace and Electronic Systems*, *58*(6), 5043-5077.

Sikiru, M. S., Adepoju, S. A., Adebayo, K. A., Salawu, I. A., & Olawoyin, L. A. (2024). Benchmarking deep learning models on nvidia jetson nano for real-time systems: An empirical investigation. *Available at SSRN or ResearchGate*. (Available: https://www.researchgate.net/publication/381703585_Benchmarking_Deep_Learning_Models_on_NVIDIA_Jetson_Nano_for_Real-Time_Systems_An_Empirical_Investigation)

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, *45*(4), 427-437. https://doi.org/https://doi.org/10.1016/j.ipm.2009.03.002

Staab, R., Vero, M., Balunović, M., & Vechev, M. (2024). *Beyond memorization: Violating privacy via inference with large language models.* Retrieved from https://arxiv.org/abs/2310.07298

Standard, H. E. (2016). *Short range devices (srd) using ultra wide band technology (uwb); harmonised standard covering the essential requirements of article 3.2 of the directive 2014/53/eu; part 1: Requirements for generic uwb applications* (Tech. Rep.). European Telecommunications Standards Institute (ETSI). Re-

trieved from https://www.etsi.org/deliver/etsi_en/302000_302099/30206501/02.01.01_60/en_30206501v020101p.pdf

Subramani, D., & Araujo, S. (2022, June). *Demystifying machine learning at the edge through real use cases.* https://aws.amazon.com/blogs/machine-learning/demystifying-machine-learning-at-the-edge-through-real-use-cases/. (Accessed: 2025-09-24)

Suzuki, T. (2020). *Factor graph optimization library for gnss positining.* GitHub. Retrieved from https://github.com/taroz/gtsam_gnss

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2020). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, *108*(8), 1355–1386.

Sünderhauf, N., & Protzel, P. (2012). Towards robust graphical models for gnss-based localization in urban environments. In *International multi-conference on systems, signals devices* (p. 1-6).

Takahashi, A., Yano, K., & Kano, M. (2025). A gnss-velocity clustering method applicable at local to global scales. *Journal of Geophysical Research: Solid Earth*, *130*(2), e2024JB029689.

Takasu, T., & Yasuda, A. (2009). Development of the low-cost rtk-gps receiver with an open source program package rtklib. In *The international symposium on gps/gnss, jeju, korea.*

Tang, J., Li, Z., Hou, K., Li, P., Zhao, H., Wang, Q., ... Xie, S. (2024a). Improving gnss positioning correction using deep reinforcement learning with adaptive reward augmentation. *NAVIGATION*, *71*(4).

Tang, J., Li, Z., Hou, K., Li, P., Zhao, H., Wang, Q., ... Xie, S. (2024b). Improving gnss positioning correction using deep reinforcement learning with an adaptive reward augmentation method. *NAVIGATION*, *71*(4). (Author-accepted manuscript open)

Tegou, T., Kalamaras, I., Votis, K., & Tzovaras, D. (2018, sep). A low-cost room-level indoor localization system with easy setup for medical applications. In *2018 11th IFIP wireless and mobile networking conference (WMNC).* IEEE. https://doi.org/10.23919/wmnc.2018.8480912

TensorFlow Team. (2020). *Tensorflow model optimization toolkit.* https://www.tensorflow.org/model_optimization. (Accessed: 2025-05-28)

Teunissen, P. J., & Montenbruck, O. (2017). *Springer handbook of global navigation satellite systems* (P. J. Teunissen & O. Montenbruck, Eds.). Springer International Publishing. https://doi.org/10.1007/978-3-319-42928-1

Teunissen, P. J. G., & Montenbruck, O. (Eds.). (2017). *Springer handbook of global navigation satellite systems*. Springer.

Tian, S., Jiabin, C., Chunlei, S., & Huan, Y. (2017). The application of r-t-s smoothing algorithm in the post-processing of the integrated navigation. In *2017 29th chinese control and decision conference (ccdc)* (p. 197-201).

Tim, E. (2021). *Batch processing rtklib solutions with rnx2rtkp and python.* Retrieved from https://rtklibexplorer.wordpress.com/2022/01/05/batch-processing-rtklib-solutions-with-rnx2rtkp-and-python/

Titterton, D., Weston, J. L., & Weston, J. (2004). *Strapdown inertial navigation technology* (Vol. 17). IET.

Tracy Cozzens. (2020). *Apple applies for machine learning GNSS device.* (https://www.gpsworld.com/apple-applies-for-machine-learning-gnss-device/, Last accessed on 2022-07-04)

Tuan, L., Hongping, Z., Xiaoji, N., & Zhouzheng, G. (2017). Tightly-coupled integration of multi-gnss single-frequency rtk and mems-imu for enhanced positioning performance. *Sensors*, *17*(11), 2462.

Tucker, S. (2024). A systematic review of geospatial location embedding approaches in large language models: A path to spatial ai systems. *ArXiv*, *abs/2401.10279*. Retrieved from https://api.semanticscholar.org/CorpusID:267061215

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*, 2579–2605.

Vandermeeren, S., Van de Velde, S., Bruneel, H., & Steendam, H. (2018). A Feature Ranking and Selection Algorithm for Machine Learning-Based Step Counters. *IEEE Sensors Journal*, *18*(8), 3255–3265.

Wang, H., Pan, S., Gao, W., Xia, Y., & Ma, C. (2022). Multipath/nlos detection based on k-means clustering for gnss/ins tightly coupled system in urban areas. *Micromachines*, *13*(7), 1128. https://doi.org/10.3390/mi13071128

Wang, H., Wang, X., Xue, Y., & Jiang, Y. (2020, jun). UWB-based Indoor Localization Using a Hybrid WKNN-LSTM Algorithm. In *2020 IEEE 4th information tech-*

nology networking, electronic and automation control conference (ITNEC). IEEE. https://doi.org/10.1109/itnec48623.2020.9085050

Wang, J., Gao, Y., Li, Z., Meng, X., & Hancock, C. M. (2016). A tightly-coupled gps/ins/uwb cooperative positioning sensors system supported by v2i communication. *Sensors*, *16*(7). https://doi.org/10.3390/s16070944

Wang, L., Jiang, Y., Ji, H., & Wei, W. (2024). Amplitude scintillation detection with geodetic gnss receivers leveraging machine learning decision tree. *Satellite Navigation*, *5*, 18. https://doi.org/10.1186/s43020-024-00136-7

Wang, S., Hu, T., Xiao, H., Li, Y., Zhang, C., Ning, H., ... Ye, X. (2024). Generative ai models in geospatial science: A systematic review. *International Journal of Digital Earth*, *17*(1).

Wang, Y. (2012). Gauss–newton method. *Wiley Interdisciplinary Reviews: Computational Statistics*, *4*(4), 415–420.

Wang, Y., & Li, X. (2017, August). The imu/uwb fusion positioning algorithm based on a particle filter. *ISPRS International Journal of Geo-Information*, *6*(8), 235. https://doi.org/10.3390/ijgi6080235

Wang, Z., Li, X., Zhu, Y., Li, Q., & Fang, K. (2022). Integrity monitoring of global navigation satellite system/inertial navigation system integrated navigation system based on dynamic fading filter optimisation. *IET Radar, Sonar & Navigation*, *16*(3), 515–530.

Warden, P., & Situnayake, D. (2019). *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media.

Wen, W., Pfeifer, T., Bai, X., & Hsu, L.-T. (2021). Factor graph optimization for gnss/ins integration: A comparison with the extended kalman filter. *NAVIGATION: Journal of the Institute of Navigation*, *68*(2), 315–331. https://doi.org/10.1002/navi.421

Wilzeck, A., Guirao, M. P., & Dimitrov, E. (2018). White Paper on UWB Technology and Regulation. *wiseSense GmbH*.

Woodman, O. J. (2007). *An introduction to inertial navigation* (Tech. Rep.). University of Cambridge, Computer Laboratory. Retrieved from [2024-11-14]chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf

Wu, Z., Zhang, Y., Yang, Y., Liang, C., & Liu, R. (2020). Spoofing and anti-

spoofing technologies of global navigation satellite system: A survey. *IEEE Access*, *8*, 165444-165496.

Wymeersch, H., Muppirisetty, L. S., Raulefs, R., Slock, D., Svensson, T., & Wymeersch, H. (2014). Location-aware communications for 5g networks: How location information can improve scalability, latency, and robustness of 5g. *IEEE Signal Processing Magazine*, *31*(6), 102–112. Retrieved from chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://mcube.lab.nycu.edu.tw/~cfung/docs/5G/di_taranto2014location_aware5G.pdf

Xhafa, A., del Peral-Rosado, J. A., Seco-Granados, G., & Lopez-Salcedo, J. A. (2021, April). Performance of NLOS base station exclusion in cmWave 5g positioning. In *2021 IEEE 93rd vehicular technology conference (VTC2021-spring)*. IEEE. https://doi.org/10.1109/vtc2021-spring51267.2021.9448738

Xia, N., & Weitnauer, M. A. (2019). TDOA-Based Mobile Localization Using Particle Filter With Multiple Motion and Channel Models. *IEEE Access*, *7*, 21057–21066.

Yang, M. Y., Rosenhahn, B., & Murino, V. (2019). *Chapter 8 - multimodal localization for embedded systems: A survey*. Academic Press. Retrieved from https://www.sciencedirect.com/science/article/pii/B9780128173589000147

Yang, S., & Wang, B. (2017, July). Residual based weighted least square algorithm for bluetooth/UWB indoor localization system. In *2017 36th chinese control conference (CCC)*. IEEE. https://doi.org/10.23919/chicc.2017.8028303

Yang, X., Zhao, F., & Chen, T. (2018, apr). NLOS identification for UWB localization based on import vector machine. *AEU - International Journal of Electronics and Communications*, *87*, 128–133. https://doi.org/10.1016/j.aeue.2018.02.003

Y. Bar-Shalom, T. K., X. Rong Li. (2001). *Estimation with Applications To Tracking and Navigation*. John Wiley & Sons, Inc.

Yiming, Q. (2017). A new machine learning based method for multi-gnss data quality assurance and multipath detection. *PhD thesis, Faculty of Science and Engineering, Department of Civil Engineering, University of Nottingham*. Retrieved from [08 December 2021]http://eprints.nottingham.ac.uk/39748/

Yousefpour, A., & et al. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, *98*, 289–330.

Zafari, F., Gkelias, A., & Leung, K. K. (2019). A survey of indoor localization sys-

tems and technologies. *IEEE Communications Surveys & Tutorials*, *21*(3), 2568–2599. https://doi.org/10.1109/comst.2019.2911558

Zafari, F., Papapanagiotou, I., & Christidis, K. (2016). Microlocation for internet-of-things-equipped smart buildings. *IEEE Internet of Things Journal*, *3*(1), 96-112.

Zhang, K., & Papadimitratos, P. (2019). Secure multi-constellation gnss receivers with clustering-based solution separation algorithm. In *2019 ieee aerospace conference* (pp. 1–9). Retrieved from https://kth.diva-portal.org/smash/get/diva2:1301724/FULLTEXT01.pdf

Zhao, S., Huang, B., & Liu, F. (2016). Localization of indoor mobile robot using minimum variance unbiased fir filter. *IEEE Transactions on Automation Science and Engineering*, *15*(2), 410–419.

Zheng, Y., Xie, W., & Ma, W.-Y. (2009). *Mining interesting locations and travel sequences from gps trajectories.* Retrieved from https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/fp120-zheng.pdf

Zhou, Q., Zhang, H., Li, Y., & Li, Z. (2015, September). An adaptive low-cost GNSS/MEMS-IMU tightly-coupled integration system with aiding measurement in a GNSS signal-challenged environment. *Sensors*, *15*(9), 23953–23982. https://doi.org/10.3390/s150923953

Zhu, B., Tao, X., Zhao, J., Ke, M., Wang, H., & Deng, W. (2020, October). An integrated gnss/uwb/dr/vmm positioning strategy for intelligent vehicles. *IEEE Transactions on Vehicular Technology*, *69*(10), 10842–10853. https://doi.org/10.1109/tvt.2020.3014516

Zhu, D., Zhong, Z., Zhang, M., Wu, S., Zhang, K., Li, Z., … Liu, J. (2023). An improved principal component analysis method for the interpolation of missing data in gnss-derived pwv time series. *Remote Sensing*, *15*(21), 5153. https://doi.org/10.3390/rs15215153

Publication I

Machine Learning Utilization in GNSS—Use Cases, Challenges and Future Applications

Akpojoto Siemuri
Digital Economy Research Platform
University of Vaasa
Vaasa, Finland
akpo.siemuri@univaasa.fi

Heidi Kuusniemi
Digital Economy Research Platform
University of Vaasa
Vaasa, Finland
heidi.kuusniemi@univaasa.fi

Mohammed S. Elmusrati
School of Technology and Innovations
University of Vaasa
Vaasa, Finland
moel@univaasa.fi

Petri Välisuo
School of Technology and Innovations
University of Vaasa
Vaasa, Finland
petri.valisuo@univaasa.fi

Ahm Shamsuzzoha
School of Technology and Innovations
University of Vaasa
Vaasa, Finland
ahm.shamsuzzoha@uwasa.fi

***Abstract*— The algorithms and models of traditional global navigation satellite systems (GNSSs) perform very well in terms of the availability and accuracy of positioning, navigation and timing (PNT) under good signal conditions. Research is still ongoing to improve their robustness and performance in less than optimal signal environments. A growing interest in the study of machine learning (ML) and the potential for its application in many fields has also led to several types of research on its utilization in GNSSs. In the field of GNSSs, ML is changing the ways that navigation problems are prevented and resolved, and it is taking on a significant role in advancing PNT technologies for the future. We illustrate this point by reviewing how ML can enhance GNSS performance and usability and also discuss areas of GNSSs in which ML algorithms have been applied. We also highlight the commonly implemented ML algorithms and compare their performance when used in similar GNSS use cases. In addition, the challenges and risks of the utilization of ML techniques in GNSSs are discussed. Insight is given into prospective areas in GNSSs in which ML can be applied for increased performance, accuracy and robustness, thereby providing fertile ground for novel research.**

Keywords*— *Machine Learning (ML), Deep Learning (DL), Global Navigation Satellite Systems (GNSSs), PNT technologies, GNSS performance.

I. Introduction

In global navigation satellite systems (GNSSs), there are several sources of errors for satellite-based positioning that affect accuracy and degrade receiver performance, namely, the ionospheric effect, multipath propagation, spoofing and jamming interferences and GNSS-denied environments. Other sources of error in GNSSs include receiver noise and resolution, satellite clock errors and hardware biases [1]. Therefore, it is interesting to see how machine learning (ML) has and can be applied in such cases.

Machine-learning techniques are well suited for real-life problems and are tolerant to data that are imprecise, partially incorrect or uncertain [2]. ML is a very powerful technique in handling time-series data, and it can learn time-dependent patterns across multiple models. Therefore, ML can be found useful in the discovery of unknown and hidden information in GNSS data. The nearly limitless quantity of available data, affordable data storage and the growth of less expensive and more powerful processing has propelled the growth of ML. Compared to statistical methods, ML methods enable us to identify tricky dependencies in datasets for which exploratory analysis has not enabled the proper determination of the shape of the underlying model. The aim in ML is not to provide an explicit formula for the distribution of the data; rather, an algorithm is trained to detect the relationships among the features of the data on its own, directly from the data. Learning methods makes it possible to avoid the assumptions involved in many statistical methodologies. A demonstration of the concept of training using a machine-learning approach is shown in Fig. 1. It shows an ML model trained to classify a GNSS signal as line of sight (LOS)/multipath/non-line of sight (NLOS).

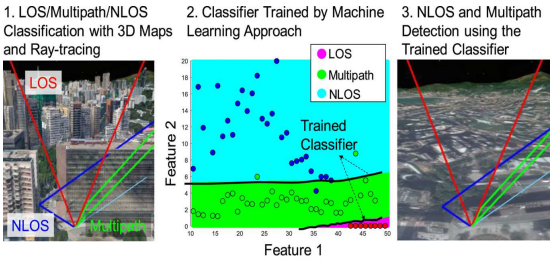


Fig. 1. Demonstration of the concept of training using a machine-learning approach [3].

This paper presents some major applications in GNSSs in which ML has been utilized to provide a novel solution or a new service. The different ML algorithms, methods and solutions used in the primary literature are compared to show how they are used. The rest of the paper is organized as follows: Section II presents the review process and the research questions that are discussed. In Sections III through V, the research questions used are discussed, while Section VI discusses the challenges and risks of using ML algorithms in GNSSs. Section VII presents the conclusions and future directions.

II. Feasibility Assessment

To facilitate the use of ML algorithms in GNSSs, it is necessary to review their performance and usage in the extant literature. This review began with a systematic search for applications of ML in GNSS use cases of digital libraries, leading to the retrieval of 182 papers. A total of 47 of these papers were deemed relevant and required full-text screening for detailed information on the application of ML in a GNSS. The search phrase used to identify the primary studies was *"GNSS AND (machine learning OR deep learning OR artificial intelligence OR random forest OR support vector machine OR decision tree OR neural network OR regression)"*. The four electronic databases used for the search were IEEE Xplore, ScienceDirect, Google Scholar and the Web of Science. We restricted the search to articles published from 2000 to 2020. Two other papers ([43 and [50]) were added based on recommendations from reviewers. The

978-1-7281-9644-2/21/$31.00 ©2021 IEEE

following research questions (RQs) were identified to guide the review:

RQ1. To which GNSS use cases have ML algorithms been applied and in what are other prospective areas of GNSSs could ML algorithms be utilized?

RQ2. Which ML algorithms are utilized in GNSSs and are there ML algorithms that significantly outperform the others?

RQ3. What are the challenges and risks of using ML algorithms in GNSSs?

These research questions are discussed in Sections III through V, respectively.

III. Machine Learning Utilization in GNSS Use Cases

The use of ML in GNSSs is promising, as it has proven useful in various studies. Here, we discuss the ML techniques implemented in the primary studies. The ML techniques described include decision trees (DTs), gradient-boosting decision trees (GBDTs), logistic regression (LR), long short-term memory (LSTM), naïve Bayes (NB), Gaussian process regression (GPR), deep learning (DL) models (neural networks, or NN), extreme learning machines (ELMs), support vector machines (SVM) etc. The algorithms were used for classification, clustering, forecasting and anomaly detection depending on the GNSS application. ML cannot completely replace GNSS physical models, but the use of ML/DL together with traditional GNSS has been promising and has become more popular among researchers, as discussed in the following subsections.

A. GNSS Signal Acquisition

The signal acquisition process estimates if a particular satellite signal is present or not in the received signal, and it also gives a rough estimate of its associated code delay and Doppler frequency if present [4]. This acquisition process is implemented by all GNSS receivers [5], [1]. This process is achieved through the evaluation of a cross-ambiguity function (CAF), usually in a discrete-time domain. The CAF is a two-dimensional function that is related to the correlation between the received signal and local code for every possible delay/Doppler pair. The CAF, which can be considered an image, has certain characteristics that can be used to distinguish the presence or absence of a signal from a specific satellite. This knowledge can be used to train a data-driven model, such as a multi-layer perceptron (MLP), which is a neural network architecture with moderate complexity that has been widely used in ML studies, or a convolution neural network (CNN), which is able to capture complex non-linear phenomena, at the expense of much greater complexity when compared to MLPs [4].

B. Signal Detection and Classification

Most of the solutions already in the market that implement multipath detection and mitigation are designed using stochastic modelling, spatial geometry modelling, advanced techniques in data processing, or new special hardware designs [6]; however, this is not the case for all receivers, for example, low-cost receivers and GNSS-enabled smartphones. In the detection and classification of the characteristics of an acquired signal, the model needs to be able to accurately and reliably classify the LOS, multipath and NLOS signals. However, when there is an event outside the statistical models (too complex to be modelled with classical statistics and that does not fit the mathematical assumptions used to develop the models), these solutions become ineffective. ML methods enable us to throw off assumptions required in the statistical methodology. In [7], [6], [8], [9] and [10], ML methods, such as logistic regression (LR), support vector machines (SVMs), naïve Bayes (NB), decision trees (DTs), CNNs and recurrent neural networks (RNNs), have been implemented. An artificial neural network (ANN) model capable of processing the structure of the autocorrelation function (ACF) was used for the detection of evil waveforms (EWF) in [11]. In [12] and [13], a robust gradient-boosting decision tree (GBDT) and DT-based classifier were employed, respectively, for GNSS signal reception classification that made use of the carrier-to-noise-density ratio (C/N0), pseudo-range residuals and satellite elevation angle as the input features to improve the performance of the signal classification at the receiver.

C. Earth Observation and Monitoring

GNSSs have been used in understanding the atmospheric effects of earthquakes by analysing the ionospheric behaviour before and after them through the determination of the total electron content (TEC) using GNSS data [14]. TEC is a representation of the electron density in the signal trajectory between the satellite and the receiver on the earth's surface. In [14], vertical total electron content (VTEC) data from the National Oceanic and Atmosphere Administration (NOAA) along with ionospheric disturbance information were used for training an ANN for the prediction and detection of earthquakes and the determination of their magnitude. Hurricane detection and prediction are based on the categorization of the maximum wind speed that can be estimated from the reflected GNSS signals, but to be able to track the hurricane efficiently, a large amount of measurement data is required. This is where a deep learning algorithm such as CNN is used, as seen in [15]. Similarly, in [16], a CNN model based on classification was used for sea ice detection, while a regression-based CNN was applied in sea ice concentration (SIC) estimation from TechDemoSat-1 GNSS reflectometry delay-Doppler maps (DDMs). DL has also been useful in other cases, such as environmental remote sensing, which includes land cover mapping, aerosol monitoring and the prediction of agricultural yield, snow cover, ocean colour parameters etc. [17].

D. GNSS Navigation and Precise Positioning

Nowadays, GNSS is successfully implemented to achieve precise positioning in both indoor and outdoor environments [18], [19]. Such precise positioning is essential for safe operations [20], [21]. For indoor positioning, GNSS is applied in areas like warehouse management [22], automated manufacturing [23] etc. In the case of outdoor positioning, GNSS is also suitable for tracking and tracing in various sectors, such as intelligent transportation systems (ITSs), autonomous vehicles [24], supply chain and logistics [25], port operations [26] etc. Achieving accurate and precise positioning is achieved by minimizing errors in both indoor and outdoor positioning.

ML has been applied in location-based services (LBS) with the aim of improving GNSS navigation and positioning in several scenarios. In [27], the least-squares support vector machine (LSSVM) technique was used to enhance the accuracy of Kalman filtering (KF). This enhanced model is called LSSVM-enhanced KF (LSSVM-KF). The KF relies on the quality of the observations as well as the dynamic model; therefore, LSSVM-KF adaptively estimates the dynamic modelling bias from the historical information and then uses the bias estimate to compensate the dynamic model. The dynamic model bias is treated as a time-variant ambiguous function by the algorithm, which is trained with the LSSVM [27]. Also, ML has been applied in ITSs to estimate the GNSS position error by aiding motion sensor units in providing a more accurate position estimate during periods of outage or blockage of the GNSS signal. In [28], a DT and SVM were implemented to choose the best feature for classifying the input data at every iteration and for the selection of features that were most relevant for the location error estimation.

E. Indoor Navigation

Scenario recognition is essential for seamless indoor localization and robust positioning in complex environments. In [29], the influence of multi-constellation GNSS measurements on scenario recognition performance was studied using a hidden Markov model (HMM) algorithm. This work was based on the

recognition of four scenarios (deep indoor, shallow indoor, semi-outdoor and open outdoor). The results from the scenario recognition showed a significant improvement with the increase in the number of constellations received by the smartphones. However, when the user was switching between indoor and outdoor environments, the observed variables (such as the number of visible satellites) did not respond to this change of environment immediately, as the change to the current scenario was rather slow. This caused the result of scenario recognition to lag behind the true value, and it is a challenge to effectively shorten this transition delay during scenario recognition. A scenario recognition algorithm based on an RNN, which showed better results in terms of accuracy, adaptability to new environments and real-time performance, was implemented to meet this challenge.

F. Ionospheric Scintillations and Tropospheric Wet Delay

In a GNSS receiver, signal acquisition and tracking can be severely impacted by strong scintillation, resulting in GNSS performance degradation in accuracy and continuity. It is difficult to predict and model scintillation because there are different reasons for the occurrence of this phenomenon, for example, solar activity, magnetic storms, local electric fields, conductivity, wave interaction etc. [30]. A DT model in [31] and an SVM in [30] are examples of ML algorithms employed for the detection of scintillation. The time delay of a GPS (L1 and L2) signal in the ionosphere is an example of propagation path delay, and it depends on the TEC of the atmospheric layer. This delay contributes to a potential source of error in time measurements and can produce an error range in tens of meters. In [32], Gaussian process regression (GPR) was used in comparison to an auto regressive moving average (ARMA) model and ANN model to implement the forecasting of low-latitude ionospheric conditions, and in [33], a single frequency correction algorithm based on a GNSS global ionosphere map (GIM) vertical TEC (vTEC) and fully connected neural networks (FC-NN) was used to estimate GNSS single-frequency ionospheric delay. [34] is another study on GNSS atmosphere delays in which ML has been employed. In [35], an ANN model was implemented to predict tropospheric wet delay based on meteorological and GNSS data.

G. GNSS Security—Spoofing and Jammer Attacks

Mitigation against spoofing attacks has been achieved at the pseudorange level, with receivers employing receiver autonomous integrity monitoring (RAIM) by the discovery of an inconsistent set of five or more pseudoranges to allow the receiver to identify an unsophisticated spoofer that broadcasts one or more false signals with no attempt to achieve believable consistency. However, in response to efforts to defend against spoofing, advanced forms of GNSS spoofing have been conceived, for example, a spoofing method called “in the wild”, which is an actual malicious spoofing attack [36]. Therefore, ML has been implemented to detect and defend against spoofing; for example, in [37], one- and two-hidden-layer neural networks with various numbers of hidden neurons were implemented to detect GPS spoofing signals by making use of different features, such as pseudo-range, Doppler shift and signal-to-noise ratio (SNR), to perform the classification of GPS signals. While [38] used an MLP neural network classifier trained by particle swarm optimization (PSO). Other ML algorithms used to detect jamming and spoofing include RNN based on long short-term memory (LSTM), classification SVM (C-SVM) with principal component analysis (PCA), and SC-SVM, as seen in [39], [40] and [41], respectively. In the case of GNSS jamming, in [42], SVM and CNN models were used for the detection and classification of the jammer signal. In [43], ML is used to protect users against Secure Code Estimation and Replay (SCER) attacks on Galileo OS-NMA making use of the analysis done on the features extracted from the victim’s Search Space. This ML technique is, however, complementary to the Navigation Message Authentication (NMA) techniques meaning that it will not offer any guarantee the navigation message was not modified if the ML is used separately without the NMA technique. The features extraction used is based on fitting the correlation peaks in the search space as 2D Gaussians and detecting radio frequency interferences (RFIs) during a time analysis window. The detection used in this paper is based on simply finding outliers with the assumption that an Automatic Gain Controller (AGC) is present. Using algorithms based on Decision Trees (DT, ADA Boost, and Random Forest) gave the best results when signal C/N0 is above 30dBHz. Other algorithms implemented were Nearest Neighbour, and RBF SVM algorithm with the accuracy decreasing when the location of the peaks is introduced, particularly for high C/N0. While Linear SVM showed that as the C/N0 improves the results also improve, and give values similar to that of the Decision Trees when C/N0 is greater than 30 dBHz.

H. GNSS/Inertial Navigation Systems (GNSS/INS) Integration

Several technologies that are essential to reducing positioning errors are now being researched [28]. Nowadays, a GNSS is interfaced with inertial navigation systems (INS) along with some filtering techniques with the objective of improving the overall positioning system [44]. Usually, a combination of an INS with various filters is used to calculate the accurate position. In such a situation, a GNSS outage of shorter duration does not have much effect on estimating accurate positioning. However, if the outage becomes longer, then the accuracy of the positioning decreases significantly [45]. Therefore, it is a common concern for a longer GNSS outage that makes for inaccurate precision positioning. In the case of a longer outage, the application of machine-learning algorithms can help to maintain positioning accuracy [28].

The Kalman filter (KF) is widely used in navigation as a data-fusion algorithm. In the integration of GNSS/INS, when a GNSS cannot normally supply measurement updates, the filter time will be increased. In such cases, the divergence of strap-down inertial navigation system (SINS) or INS error occurs quickly without GNSS information correction. This could be detrimental depending on the use case, such as in the concealment of unmanned underwater vehicles (UUVs) and unmanned aerial vehicles (UAVs). In [46], a backpropagation neural network (BPNN)-aided integrated navigation method based on vehicle motion learning was proposed. In [47], a recurrent neural network (RNN) in comparison with an extreme learning machine (ELM) and extended Kalman filter (EKF) was employed, while [48] used an NN model to design a neural network-assisted GNSS/SINS calibration system. A CNN-based adaptive Kalman filter was designed and discussed in [49]. In [50], ML is integrated with 3DMA (3D modelled assisted GNSS) to achieve high accuracy in urban environments. This is done using ML/AI algorithm together with Google’s vast database of 3D building models, the asymmetric NLOS propagations are modelled in order to correct NLOS pseudorange errors. This was shown to reduce Wrong-Side-of-Street occurrences from GNSS in phones up to 50 to 90%.

In general, when a GNSS is active, the ML model is used to learn the divergence characteristics of the INS error under several basic conditions, depending on the area of application (vehicles, UAV etc.). If there is a disturbed GNSS signal, the ML model is used to correct the position error of the INS in order to improve navigation accuracy.

I. Satellite Selection

Location accuracy is the result of two main factors, namely, the satellite location-dependent geometric dilution of precision (GDOP) and the pseudorange measurement inaccuracies [51]. The benefit of multi-constellation GNSS is that there are more visible satellites to improve user positioning performance. However, due to some issues, such as low-cost receivers having limited tracking receiver channels (four or five channels) and power consumption (also applicable to

sophisticated receivers having up to 12 channels), it is usually not possible, beneficial or desirable to use all satellites in view for positioning [52]. Instead, an optimal subset is generally selected from all the possible visible satellite combinations with the aim of minimizing either the GDOP or weighted GDOP (WGDOP), as in the case of [53]. The GDOP is computed from the trace of the inverse of the measurement matrix of the contributing satellites [54]. Other selection criteria used in studies include elevation angle, C/N0 and range errors. In [53], an end-to-end DL network for satellite selection based on the PointNet [55] and VoxelNet [56] networks were proposed. The satellite selection procedure is converted to a satellite segmentation problem that has two class labels, one for selected satellites and the other for satellites not selected. In [54], the functional relationships between the entries of a measurement matrix and the eigenvalues of its inverse are learned using a neural network, and the model is able to give the GDOP values without the need to compute the inverse of a matrix.

IV. What are Other Prospective GNSS Areas in Which ML Algorithms Could Be Utilized?

There are other GNSS areas in which ML can be utilized, including space applications, such as robotic exploration in space and situation awareness for space debris analysis. ML models could be used in fault detection, altitude determination of space vehicles, flight control etc. Also, ML algorithms could be proposed to estimate products similar to international GNSS service (IGS) products using the IGS observation data model to correct GNSS position error. Furthermore, ML algorithms could be proposed to emulate some of the output of a satellite-based augmentation system (SBAS) processing facility and predict the performance of SBASs in the context of a single frequency, single constellation (SFSC) and/or dual-frequency multi-constellation (DFMC) operation. These are areas that have seen much interest, especially in the latest European Space Agency (ESA) invitations to tender. They could also be used to detect signal diffraction, which has seen less research. This effect occurs when GNSS signals are partially obstructed but not totally blocked. Signal diffraction is usually statistically modelled into the navigation algorithms, in which they are mitigated by smoothing techniques that adapt poorly. This makes these solutions ineffective, especially when there is an event too complex to be modelled with classical statistics and does not fit the mathematical assumptions used to develop the statistical model.

There are well-known models to estimate the precise position of a GNSS receiver (e.g., tropospheric and ionospheric models) under simple scenarios with good line of sight, but there are high non-linearities if the receiver experiences multipath effects (e.g., urban scenarios, which have seen a high amount of research) or high signal dynamics (e.g., a rocket launch—in the domain of space applications). It would be interesting to use deep learning in the later problem. ML could also be applied to space applications (space debris analysis, robotic exploration in space, cube satellites etc.). We have seen from this review that artificial neural networks assisting Kalman filter positioning in order to determine the measurement noise covariance and process noise covariance of the Kalman filter have been promising and can be applied in other scenarios.

V. Are There ML Algorithms that Outperform the Others?

From the review done, ML algorithms were implemented in several GNSS use cases. It was observed that a GBDT-based algorithm achieved a classification accuracy higher than distance weighted k-nearest neighbour (KNN), traditional DT and an adaptive network-based fuzzy inference system (ANFIS) [12] for GPS signal reception classification (LOS/NLOS/multipath). In [13], the proposed DT-based classifier obtained a higher accuracy than the KNN and SVM classification techniques. The DL models have also been seen to perform much better; for example, [8], [9] and [11] showed good performance using CNN models. The next ML model with significant performance was an NB prediction model, as seen in [7], in which the model performed better than logistic regression, SVM and traditional DT, according to their studies. Another ML model with significant performance was the SVM in various forms (SVM, LSSVM-KF, C-SVM), as seen in [27], [28] and [38], respectively.

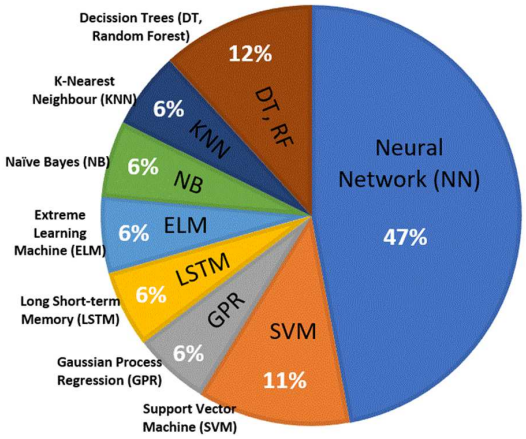


Fig. 2. A summary of the ML algorithms mostly utilized in GNSS.

As illustrated in Fig. 2, it was also noticed that the algorithms utilized in the GNSS use cases examined with better performance mostly included SVMs, 11%; DTs (GBDT), 12%; and NN (or DL) models, 47% [30], [31], [32], [34], [33], and [35]. The distribution of the DL models used based on the literature reviewed is illustrated in Fig. 3.

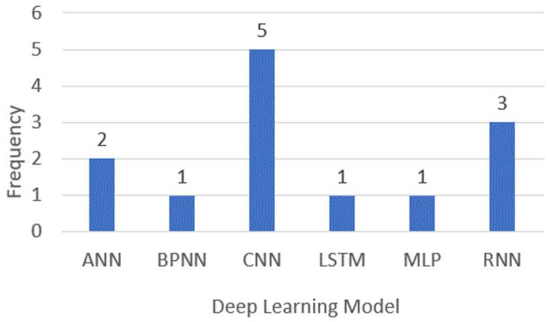


Fig. 3. Distribution of the use of deep learning models in the GNSS studies examined.

From Fig. 3, we can see that CNN was implemented five times, followed by RNN and ANN with frequencies of three and two, respectively.

VI. Challenges and Risks of Using ML Algorithms in GNSSs

A primary source of risk is the data used to train the ML model. It is therefore important to ensure the reliability, integrity, representativeness and security of the training data being used. Otherwise, it is possible to produce false predictions and a bad ML model. Furthermore, “false correlation” is a risk that can

occur when features are completely independent of each other and exhibit very similar behaviour by chance. This can provide a false conclusion that they are somehow connected. An example is when a high C/N0 value is incorrectly attributed to multipath propagation, even though high C/N0 values can also occur during normal LOS situations. ML algorithms are dependent on the data used for training and learning. Therefore, it is especially important to ensure the privacy and confidentiality of the data used for building or training the ML models. Otherwise, data extraction attacks could be launched by hackers that jeopardize the entire ML system. Since most ML systems take advantage of an already trained ML model, a generic model can be tweaked to meet a specific need by conducting specialized training. Therefore, what is referred to as a transfer learning attack can affect the task-specific ML model derived from the generic model. Other risks include a non-representative sample used for training data, developers' bias, system parameters, feedback loops, overfitting/underfitting, misinterpretation of the resultant models and contaminated reference data or data bias.

Lastly, because ML/DL algorithms may be hard to understand and explain (a "black box" effect, that is, a lack of transparency in some algorithms), for example, how the weighted coefficients within a neural network are arranged to arrive at the correct answers, this may lead to difficulty in validating and explaining results to stakeholders in order to allow such models to be applied in real-life scenarios.

VII. Conclusions and Future Work

Machine learning has been an enabling technology for many industrial applications in areas such as image recognition, classification, early anomaly detection and prediction. This review has shown that the utilization of ML in GNSSs has been an area of increasing research interest with positive prospects for future research and application.

The most common use cases for ML in GNSS, according to the recent research, include signal acquisition, NLOS/multipath/evil waveform (EWF) detection, signal reception classification (LOS/NLOS/multipath signal classifier), earth observation/monitoring, GNSS positioning error estimation, indoor navigation, characterization of GNSS ionospheric amplitude scintillation, detection of GNSS ionospheric scintillations/forecasting ionospheric time delay, GNSS phase scintillations, estimation of GNSS single frequency ionospheric delay, prediction of tropospheric wet delay, detection of GNSS spoofing attacks and jammer classification, GNSS/INS integration, and satellite selection etc.

We showed that among all ML methods, DL models (NNs) were the techniques that had been widely used by researchers most lately and provided the best results, followed by DTs and SVMs.

The challenges and risks in using ML in GNSSs were discussed and include a broad, but admittedly, selective sample of training data, developers' biases, system parameters, feedback loops, overfitting/underfitting, misinterpretation of the resultant models and contaminated reference data or data bias. The security of the training data and models was also identified as a risk, as they could be potentially manipulated by hackers.

Finally, we proposed other prospective areas of GNSSs to which ML algorithms could be applied, such as (but not limited to) space applications (space debris analysis, robotic exploration in space, cube satellites etc.), predictions of IGS products through the use of IGS data, SBAS performance prediction, GNSS hardware error detection/compensation, live tracking and tracing for supply chain and logistics and smart port operation.

References

[1] E. Kaplan and C. Hegarty, *Understanding GPS/GNSS: Principles and Applications,* 3rd ed. Artech House, 2017.

[2] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, vol. 27, pp. 504–518, 2015, doi: https://doi.org/10.1016/j.asoc.2014.11.023.

[3] R. Sun, L.-T. Hsu, D. Xue, G. Zhang, and W. Y. Ochieng, "GPS signal reception classification using adaptive neuro-fuzzy inference system," *Journal of Navigation*, vol. 72, no. 3, pp. 685–701, 2019, doi: 10.1017/S0373463318000899.

[4] P. Borhani-Darian and P. Closas, "Deep neural network approach to GNSS signal acquisition," 2020, doi: 10.1109/PLANS46316.2020.9110205.

[5] D. Borio, "Global positioning system: Signals, measurements and performance, second edition," Ph.D. dissertation, 2006.

[6] Y. Quan, "A new machine learning based method for multi-GNSS data quality assurance and multipath detection," Ph.D. dissertation, University of Nottingham, 2017.

[7] M. Socharoentum, H. A. Karimi, and Y. Deng, "A machine learning approach to detect non-line-of-sight GNSS signals in Nav2Nav," *23rd ITS World Congress*, 2016.

[8] Y. Quan, L. Lau, G. W. Roberts, X. Meng, and C. Zhang, "Convolutional neural network based multipath detection method for static and kinematic GPS high precision positioning," *Remote Sensing*, vol. 10, 2018, doi: 10.3390/rs10122052.

[9] E. Munin, A. Blais, and N. Couellan, "Convolutional neural network for multipath detection in GNSS receivers," 2020, doi: 10.1109/AIDA-AT48540.2020.9049188.

[10] L.-T. Hsu, "GNSS multipath detection using a machine learning approach," 2017, doi: 10.1109/ITSC.2017.8317700.

[11] A. Louis, "Neural network based evil waveforms detection," 2019, doi: 10.23919/RFI48793.2019.9111769.

[12] R. Sun, G. Wang, W. Zhang, L.-T. Hsu, and W. Y. Ochieng, "A gradient boosting decision tree based GPS signal reception classification algorithm," *Applied Soft Computing*, vol. 86, 2020, doi: https://doi.org/10.1016/j.asoc.2019.105942.

[13] B. Guermah, H. E. Ghazi, T. Sadiki, and H. Guermah, "A robust GNSS LOS/multipath signal classifier based on the fusion of information and machine learning for intelligent transportation systems," 2018, doi: 10.1109/ITMC.2018.8691272.

[14] D. Brum *et al.*, "A proposed earthquake warning system based on ionospheric anomalies derived from GNSS measurements and artificial neural networks," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2019, pp. 9295–9298, doi: 10.1109/IGARSS.2019.8900197.

[15] M. Alshaye, F. Alawwad, and I. Elshafiey, "Hurricane tracking using multi-GNSS-R and deep learning," in *2020 3rd International Conference on Computer Applications Information Security (ICCAIS)*, 2020, pp. 1–4, doi: 10.1109/ICCAIS48893.2020.9096717.

[16] Q. Yan and W. Huang, "Sea ice sensing from GNSS-R data using convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 10, pp. 1510–1514, 2018, doi: 10.1109/LGRS.2018.2852143.

[17] Q. Yuan *et al.*, "Deep learning in environmental remote sensing: Achievements and challenges," *Remote Sensing of Environment*, vol. 241, p. 111716, 2020, doi: https://doi.org/10.1016/j.rse.2020.111716.

[18] C. Ma, J. Yang, J. Chen, and Y. Tang, "Indoor and outdoor positioning system based on navigation signal simulator and pseudolites," *Advances in Space Research*, vol. 62, no. 9, pp. 2509–2517, 2018, doi: https://doi.org/10.1016/j.asr.2018.07.006.

[19] P. Dabove and V. Di Pietra, "Towards high accuracy GNSS real-time positioning with smartphones," *Advances in Space Research*, vol. 63, no. 1, pp. 94–102, 2019, doi: https://doi.org/10.1016/j.asr.2018.08.025.

[20] A. Uzun, F. A. Ghani, H. Yenigün, and İ. Tekin, "A novel GNSS repeater architecture for indoor positioning systems in ISM band," in *2020 IEEE International Symposium on Antennas and Propagation and North American Radio Science Meeting*, 2020, pp. 1631–1632, doi: 10.1109/IEEECONF35879.2020.9329653.

[21] Y. Sun, J. Wang, and J. Chen, "Indoor precise point positioning with pseudolites using estimated time biases iPPP and iPPP-RTK," *GPS Solutions*, vol. 25, no. 2, Jan. 2021, doi: 10.1007/s10291-020-01064-0.

[22] W. Jiang, Y. Li, C. Rizos, B. Cai, and W. Shangguan, “Seamless indoor-outdoor navigation based on GNSS INS and terrestrial ranging techniques,” *Journal of Navigation*, vol. 70, no. 6, pp. 1183–1204, Jul. 2017, doi: 10.1017/s037346331700042x.

[23] S. A. Platonov, A. V. Platonov, M. E. Postnikov, S. V. Khadonova, and S. S. Dymkova, “Using global navigation satellite systems to solve complex application problems,” in *2019 Systems of Signals Generating and Processing in the Field of on Board Communications*, 2019, doi: 10.1109/sosg.2019.8706807.

[24] K.-W. Chiang, G.-J. Tsai, Y.-H. Li, Y. Li, and N. El-Sheimy, “Navigation engine design for automated driving using INS/GNSS/3D LiDAR-SLAM and integrity assessment,” *Remote Sensing*, vol. 12, no. 10, p. 1564, May 2020, doi: 10.3390/rs12101564.

[25] J. Li *et al.*, “Real-time self-driving car navigation and obstacle avoidance using mobile 3D laser scanner and GNSS,” *Multimedia Tools and Applications*, vol. 76, no. 21, pp. 23017–23039, Nov. 2016, doi: 10.1007/s11042-016-4211-7.

[26] Z. Fu, X. Feng, X. Duan, and Z. Fu, “An improved integrated navigation method based on RINS GNSS and kinematics for port heavy-duty AGV,” *Proceedings of the Institution of Mechanical Engineers Part D: Journal of Automobile Engineering*, vol. 234, no. 8, pp. 2135–2153, Feb. 2020, doi: 10.1177/0954407019900031.

[27] Z. Zhou, Y. Li, C. Fu1, and C. Rizos, “Least-squares support vector machine-based Kalman filtering for GNSS navigation with dynamic model real-time correction,” *IET Radar, Sonar & Navigation*, 2016.

[28] A. Kuratomi, “GNSS position error estimated by machine learning techniques with environmental information input,” 2019.

[29] Y. Xia *et al.*, “Recurrent neural network based scenario recognition with multi-constellation GNSS measurements on a smartphone,” *Measurement*, vol. 153, p. 107420, 2020, doi: https://doi.org/10.1016/j.measurement.2019.107420.

[30] Y. Liu, Y. J. Morton, and Y. Jiao, “Application of machine learning to the characterization of GPS L1 ionospheric amplitude scintillation,” in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2018, pp. 1159–1166, doi: 10.1109/PLANS.2018.8373500.

[31] N. Linty, A. Farasin, A. Favenza, and F. Dovis, “Detection of GNSS ionospheric scintillations based on machine learning decision tree,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 1, pp. 303–317, 2019, doi: 10.1109/TAES.2018.2850385.

[32] L. M. I, D. V. Ratnam, S. Raman, and G. Sivavaraprasad, “Machine learning algorithm to forecast ionospheric time delays using global navigation satellite system observations,” *Acta Astronautica*, vol. 173, 2020, doi: 10.1016/j.actaastro.2020.04.048.

[33] R. Orus Perez, “Using TensorFlow-based neural network to estimate GNSS single frequency ionospheric delay (IONONet)”, *Advances in Space Research*, vol. 63, no. 5, pp. 1607–1618, 2019, doi: https://doi.org/10.1016/j.asr.2018.11.011.

[34] K. Lamb *et al.*, “Prediction of GNSS phase scintillations: A machine learning approach,” *arXiv:1910.01570*, 2019, doi: http://arxiv.org/abs/1910.01570.

[35] M. O. Selbesoglu, “Prediction of tropospheric wet delay by an artificial neural network model based on meteorological and GNSS data,” *Engineering Science and Technology, an International Journal*, vol. 23, no. 5, pp. 967–972, 2020, doi: https://doi.org/10.1016/j.jestch.2019.11.006.

[36] M. L. Psiaki and T. E. Humphreys, “GNSS spoofing and detection,” *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1258–1270, 2016, doi: 10.1109/JPROC.2016.2526658.

[37] M. R. Manesh, J. Kenney, W. C. Hu, V. K. Devabhaktuni, and N. Kaabouch, “Detection of GPS spoofing attacks on unmanned aerial systems”, 2019, doi: 10.1109/CCNC.2019.8651804.

[38] S. Tohidi and M. R. Mosavi, “Effective detection of GNSS spoofing attack using a multi-layer perceptron neural network classifier trained by PSO,” in *2020 25th International Computer Conference, Computer Society of Iran (CSICC)*, 2020, pp. 1–5, doi: 10.1109/CSICC49403.2020.9050078.

[39] R. Calvo-Palomino, A. Bhattacharya, G. Bovet, and D. Giustiniano, “Short: LSTM-based GNSS spoofing detection using low-cost spectrum sensors,” in *2020 IEEE 21st International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2020, pp. 273–276, doi: 10.1109/WoWMoM49955.2020.00055.

[40] S. Semanjski, A. Muls, I. Semanjski, and W. De Wilde, “Use and validation of supervised machine learning approach for detection of GNSS signal spoofing,” in *2019 International Conference on Localization and GNSS (ICL-GNSS)*, 2019, pp. 1–6, doi: 10.1109/ICL-GNSS.2019.8752775.

[41] S. Semanjski, I. Semanjski, W. D. Wilde, and S. Gautama, “Use of supervised machine learning for GNSS signal spoofing detection with validation on real-world meaconing and spoofing data —Part II”, *Sensors*, vol. 20, 2020, doi: 10.3390/s20071806.

[42] R. M. Ferre, A. de la Fuente, and E. S. Lohan, “Jammer classification in GNSS bands via machine learning algorithms,” *Sensors*, vol. 19, 2019, doi: 10.3390/s19224841.

[43] F. Gallardo and A. P. Yuste, "SCER Spoofing Attacks on the Galileo Open Service and Machine Learning Techniques for End-User Protection," in IEEE Access, vol. 8, pp. 85515-85532, 2020, doi: 10.1109/ACCESS.2020.2992119.

[44] I. Belhajem, Y. B. Maissa, and A. Tamtaoui, “Improving vehicle localization in a smart city with low cost sensor networks and support vector machines,” *Mobile Networks and Applications*, vol. 23, no. 4, pp. 854–863, May 2017, doi: 10.1007/s11036-017-0879-9.

[45] S. Joardar, T. A. Siddique, S. Alam, and M. Hossam-E-Haider, “Analyses of different types of errors for better precision in GNSS,” in *2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, 2016, pp. 1–6, doi: 10.1109/CEEICT.2016.7873127.

[46] Z. Zhang, F. Wu, Y. Liu, Y. Zuo, and Y. Ji, “A neural network aided integrated navigation algorithm based on vehicle motion mode information,” in *2019 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, 2019, pp. 1–5, doi: 10.1109/ICSPCC46631.2019.8960761.

[47] H. Dai, H. Bian, R. Wang, and H. Ma, “An INS/GNSS integrated navigation in GNSS denied environment using recurrent neural network,” *Defence Technology*, vol. 16, 2020, doi: 10.1016/j.dt.2019.08.011.

[48] Z. Liu, N. Wu, and F. Wang, “Application of neural network-assisted GNSS/SINS calibration system in UUV,” in *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2019, pp. 1253–1257, doi: 10.1109/IMCEC46724.2019.8984176.

[49] Z. Zou, T. Huang, L. Ye, and K. Song, “CNN based adaptive Kalman filter in high-dynamic condition for low-cost navigation system on highspeed UAV,” in *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, 2020, pp. 103–108, doi: 10.1109/ACIRS49895.2020.9162601.

[50] F. van Diggelen, Google's Use of 3D Building Models to Solve Urban GNSS, ITM/PTTI 2021 PLENARY SESSION. Available online: https://www.ion.org/itm/virtual-schedule.html.

[51] T. Soininen, P. Syrjarinne, S. Ali-Loytty, and C. Schmid, “Data-driven approach to satellite selection in multi-constellation GNSS receivers,” 2018, doi: 10.1109/ICL-GNSS.2018.8440912.

[52] C. Park, I. Kim, J. G. Lee, and G.-I. Jee, “A satellite selection criterion incorporating the effect of elevation angle in GPS positioning,” *Science Direct*, vol. 4, 1996, doi: https://doi.org/10.1016/S0967-0661(96)00192-X.

[53] P. Huang, C. Rizos, and C. Roberts, “Satellite selection with an end-to-end deep learning network,” *GPS Solution*, vol. 22, 2018, doi: 10.1007/s10291-018-0776-0.

[54] D. Simon and H. El-Sherief, “Navigation satellite selection using neural networks,” *Neurocomputing*, vol. 7, no. 3, pp. 247–258, Apr. 1995, doi: 10.1016/0925-2312(94)00024-m.

[55] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” *arXiv:1612.00593*, 2017.

[56] Y. Zhou and O. Tuzel, “VoxelNet: End-to-end learning for point cloud based 3D object detection,” *arXiv:1711.06396*, 2017.

Publication II

Improving Precision GNSS Positioning and Navigation Accuracy on Smartphones using Machine Learning

Akpojoto Siemuri, Kannan Selvan, Heidi Kuusniemi, Petri Välisuo, Mohammed S. Elmusrati
School of Technology and Innovations, University of Vaasa, Finland

BIOGRAPHY

Akpojoto Siemuri received his B.Sc.(tech) degree in electrical and computer engineering from Federal University of Technology Minna, Nigeria in 2010, the M.Sc.(tech) degree in wireless industrial automation, and a minor study in industrial management from the University of Vaasa, Finland in 2019. He is currently pursuing a Ph.D. degree in automation technology at the University of Vaasa. From 2018 to 2019, he was a Research Assistant in the Smart Energy Systems Research Platform (SESP) Project at the University of Vaasa, Finland. He is currently a Project Researcher in Digital Economy Research Platform, University of Vaasa. His research interest includes machine learning, GNSS technologies, smart devices, embedded systems, communication systems, and game theory.

Kannan Selvan received his B.Sc.(tech) degree in Electronics and Communication Engineering from Anna University, India in 2012, the M.Sc.(tech) degree in Communications and Systems Engineering from the University of Vaasa, Finland in 2020. From 2018 to 2020, he was a Research Assistant in the Digital Economy Research Platform at the University of Vaasa, Finland. He is currently a Project Researcher in Digital Economy Research Platform, University of Vaasa. His research interest includes GNSS technologies, satellite-data analysis, machine learning, satellite communication, smart devices and embedded Systems.

Dr. Heidi Kuusniemi is a professor in computer science and director of Digital Economy at the University of Vaasa in Finland. She is also a part-time research professor in satellite navigation at the Finnish Geospatial Research Institute. She has a M.Sc. (Tech.) degree (with distinction) from 2002 and a D.Sc. (Tech.) degree from 2005 in information technology, respectively, from Tampere University of Technology, Finland. She serves as a member of the council of natural sciences and technology at the Academy of Finland and was a member of the scientific advisory committee for GNSS (GSAC) at ESA. Her technical expertise and interests include GNSS reliability and resilience, estimation and data fusion, mobile precision positioning, indoor localization and PNT in new space.

Petri Välisuo is currently working as an Associate Professor (tenure track), sustainable automation, in the School of Technology and Innovation Mangement of University of Vaasa, Finland. He received M.Sc.(tech) degree in computer science from the Tampere University of Technology, Finland, and D.Sc.(tech) degree in automation technology from University of Vaasa, in years 1996 and 2011 respectively. He has authored and co-authored 27 peer reviewed and more than 10 other scientific publications. His research interests cover machine learning, IoT, positioning methods and other technologies relevant to industrial automation. He has been working 10 years in telecommunication industry before research career in the University of Vaasa.

Prof. Mohammed S. Elmusrati received his B.Sc. (with honors) and M.Sc. (with high honors) degrees in electrical and electronic engineering, University of Benghazi, Libya, in 1991 and 1995, respectively, and the Licentiate of Science in technology (with distinction) and the Doctor of Science in Technology (D.Sc.) degrees in automation and control engineering from Aalto University Finland, in 2002 and 2004, respectively. Currently, he is Full Professor and Head of the Digitalization Unit at the School of Technology and Innovations – University of Vaasa, Finland. His research interest includes wireless communications, artificial intelligence, machine learning, biotechnology, big data analysis, stochastic systems, and game theory. Elmusrati has published more than 130 papers, books, and book chapters. Prof. Elmusrati is an active member in different scientific societies such as Senior Member at IEEE, Member at Society of Industrial and Applied Mathematics (SIAM), and Member at Finnish Automation Society.

ABSTRACT

In this work, we developed a precision positioning algorithm for multi-constellation dual-frequency global navigation satellite systems (GNSS) receivers that predicts the latitude and longitude from smartphone GNSS data. Estimation for all epochs that have at least four valid GNSS observations is generated. Receivers (especially low-cost receivers) often have limited channels and computational resources, therefore, the complexity of the algorithm used in them needs to be kept low. The datasets and results in this paper are based on the data provided by Google under the session "High Precision GNSS Positioning on Smartphones Challenge" in the Institute of Navigation (ION GNSS+ 2021) conference.

We began by exploring and analysing the raw GNSS data which includes the training dataset and its ground truth and the test dataset without the ground truth. This analysis gave insight into the nature and correlation of the dataset and helped shape the algorithm that was proposed for the accuracy improvement problem. The design of the algorithm was done using data science techniques to compute the average of the predictions of several devices data in the same collection (training dataset baseline coordinates and their ground truth) and then the data was used to train a few selected machine learning algorithms namely, Linear Regression (LR), Bayesian Ridge (BR) and Neural Network (NN) to predict the offset of the test data baseline coordinates from the expected ground-truth (which was not provided).

A simple weighted average (SWA) which combines all the previous three ML technique was also implemented. The results showed improvement in the position accuracy with the simple weighted average (SWA) method having the best accuracy followed by Bayesian Ridge (BR), Linear Regression (LR), and then Neural Network (NN) respectively.

Keywords - Machine Learning; Linear Regression; Bayesian Ridge; Neural Networks; Simple Weighted Average; Global Navigation Satellite Systems; Smartphones; Mean Prediction

I. INTRODUCTION

1. Global Navigation Satellite Systems (GNSS)

Global Navigation Satellite Systems (GNSS) is a term used to collectively describe satellite-based positioning and timing systems. In the past decades, we have seen significant improvements in the performance of the Global Navigation Satellite System (GNSS) technology. The time it took to get a first accurate position has been improved from minutes to under thirty seconds. This improvement has been seen in the early 2000s. The latter half of the decade was the time when receiver sensitivity improvement was significant, that is, from -130 dBm to -167 dBm. The number of functional positioning satellite constellations has seen an increase in 2015. In 2000, there was only one global constellation (the USA's GPS) and as of 2015, there are four (GPS, GLONASS, BeiDou, and Galileo). This has led to the development of multi-constellation GNSS receivers. Multi-band GNSS became more affordable as of 2018 due to the modernization of the satellite signals. These advances are the foundations for the next big steps to be taken in GNSS such as achieving millimeter- or centimeter-level accuracy.

The benefit of multi-constellation GNSS is the availability of more visible satellites that can be used to improve user positioning performance [1]. This is very useful in challenging environments where GNSS signals could be partially or totally blocked and can suffer from multipath reflections, for example, in urban areas or places with dense foliage. However, most receivers have limited tracking receiver channels therefore, it may be not possible, or desirable, to use all satellites in view for positioning. Other reasons for this include power consumption, multipath or non-line-of-Sight (NLOS) signal measurements, and other issues [2]. This means a subset of satellites from the available satellites in view needs to be selected.

Randomly selecting satellites can result in a poor satellite geometry, which can increase the geometry dilution of precision (GDOP), and consequently induce accuracy degradation. It has been common to use a GDOP or weighted GDOP (WGDOP) as a satellite selection criterion for positioning [1], [3], [4], and [5] etc. This considers only the geometric effects, however, there are other factors that cause positioning error, such as ionospheric delay, tropospheric delay, and multipath propagation causing ranging measurement errors.

The Global Navigation Satellite System (GNSS) provides raw signals, which the GNSS chipset in smartphones uses to compute a position. The Google challenge for which this paper is written provided the data collected from some Android smartphones. These data were released to be used in computing location down to decimeter or even centimeter resolution, if possible. Google also provided access to precise ground truth (for training dataset only), raw GNSS measurements, and assistance data from nearby GNSS stations, in order to train and test machine learning algorithms for providing improved positioning performance.

2. Machine Learning Algorithms

Soft computing techniques are well suited for real-life problems that use methods to extract useful information from complex and intractable problems in less time. They are tolerant to data that is imprecise, partially incorrect,or uncertain. One of the important components of soft computing techniques includes machine learning (ML) techniques. The machine learning (ML) techniques have been used in the literature in order to predict models for estimating the GNSS performance. For example, [6]

have employed Gaussian Process Regression (GPR) model while comparing it to the Autoregressive moving average model, and the Artificial neural network model, in order to predict ionospheric time delays using Global Navigation satellite system observations. There are several advantages of making use of these ML techniques for estimation, prediction, and enhancement of GNSS performance.

ML is a very powerful tool in processing time-series data, as it can be applied in learning time-dependent patterns across multiple models. It has been found useful in discovery hidden and unknown patterns and information in GNSS data. The GNSS provides nearly limitless and affordable quantity of data, also, data storage and the growth of less expensive and more powerful processing capabilities has propelled the growth of ML. When compared to statistical methods, ML techniques enable us to identify tricky dependencies in data for which exploratory analysis has not enabled the proper determination of the shape of the underlying model [7]. The aim of using ML is not to generate an explicit formula for the distribution of the data; however, it is used to train an algorithm to detect the relationships between the features of a data set, directly from the data. This learning methods makes it possible to avoid making assumptions as seen in many statistical methodologies.

II. MATERIALS, METHODS, AND MODELS

In our work, we used the average of the predictions of several phones in the same data collection using baseline location data provided by Google. These baseline location data were computed using a simple least square solution. Next, we apply three machine learning (ML) algorithms Linear Regression (LR), Bayesian Ridge (BR) and Neural Network (NN) respectively to get the final prediction. These ML algorithms were implemented separately for comparison of their accuracy.

The steps or approach taken are as follows:

1. Data Analysis and Preparation
2. Reject Outlier and Apply Kalman Filter
3. Perform Phones Mean Prediction
4. Apply Linear Regression (LR), Bayesian Ridge (BR) and Neural Network (NN) respectively to get the final prediction for Train Dataset
5. Evaluate Train Score
6. Apply Linear Regression (LR), Bayesian Ridge (BR) and Neural Network (NN) respectively to test dataset to get the final prediction in latitude and longitude.
7. Perform a simple weighted average (SWA) on the results of the BR, LR, and NN models.

1. Data Analysis and Preparation

Google releases a total of 121 traces in the challenge, whose collection process is described in a paper published in the proceedings of ION GNSS+ 2020 [8]. The data were released twice, first the training data and next the test data. Most of the data were collected in US Bay area highways with open sky. A small part of some traces show sparse buildings, overpasses, or trees.

a). Training datasets

The training datasets consisted of 73 traces. The data included GnssLogger files, RINEX observation files, ground truth files, and derived files (GNSS intermediate values derived from raw GNSS measurements, provided for convenience).

The derived values contains values like:

1. The ionospheric delay in meters, estimated with the Klobuchar model.
2. The tropospheric delay in meters, estimated with the EGNOS model by Nigel Penna, Alan Dodson and W. Chen (2001).
3. Raw pseudorange uncertainty in meters.
4. GNSS constellation type. An integer number, representing the available GNSS constellations.
5. The satellite ID.
6. The GNSS signal type which is a combination of the constellation name (forexample, GPS) and the frequency band (for example, L1).
7. The signal transmission time received by the chipset, which is the numbers of nanoseconds since the GPS epoch.

8. The satellite position in meters ([x/y/z]SatPosM) in an ECEF coordinate frame at best estimate of "true signal transmission time" defined as ttx = receivedSvTimeInGpsNanos - satClkBiasNanos (defined below). They are computed with the satellite broadcast ephemeris, and have 1-meter error with respect to the true satellite position.
9. The satellite velocity (meters per second - [x/y/z]SatVelMps) in an ECEF coordinate frame at the signal transmission time (receivedSvTimeInGpsNanos). They are computed with the satellite broadcast ephemeris, with this algorithm.
10. The satellite time correction combined with hardware delay in meters at the signal transmission time (receivedSvTime-InGpsNanos) called satellite clock bias (satClkBiasM). Its time equivalent is termed as satClkBiasNanos.
11. The satellite clock drift in meters per second (satClkDriftMps) at the signal transmission time (receivedSvTimeInGp-sNanos). It equals the difference of the satellite clock biases at t+0.5s and t-0.5s.
12. Raw pseudorange in meters (rawPrM). It is the product between the speed of light and the time difference from the signal transmission time (receivedSvTimeInGpsNanos) to the signal arrival time (Raw::TimeNanos - Raw::FullBiasNanos - Raw::BiasNanos).

b). Test datasets

In the released test datasets, there are 48 more traces, which include the same types of data and follow the same convention as the training dataset, except that the ground truth files are not provided. The results of test datasets is used as the prediction of the expected ground truth. The results are evaluated by Google using the unreleased ground truth via kaggle (www.kaggle.com). In both the training and test datasets, the derived values can be used to compute a corrected pseudorange which is a closer approximation to the geometric range from the phone to the satellite.

This can be done using the formula in equation 1:

$$\rho_c = \rho + c\Delta t_{sat} - B_{isr} - c\Delta t_{in} - c\Delta t_{tr} \tag{1}$$

The baseline locations are then computed using corrected pseudorange and the satellite positions, using a standard Weighted Least Squares (WLS) solver, with the phone's position (x, y, z), clock bias (t), and *B*isr (The Inter-Signal Range Bias (ISRB) in meters from a non-GPS-L1 signal to GPS-L1 signals) for each unique signal type as states for each epoch. The GnssLogger files in both the training and test datasets include: receivedSvTime (equivalent: pseudorange), pseudorange-rate (Doppler), Accumulated Delta Range (carrier phase), for all visible GNSS satellites: GPS, GLO, GAL, BDS, QZS at 1 Hz. Data also includes uncalibrated accelerometer, gyroscope, and magnetic field readings. The baseline locations (estimated latitude and longitude coordinate for both the train/test datasets) were also released. This were generated using a simple approach, weighted least square (WLS). These baseline locations were used as the primary data for the algorithm implemented in this paper along side the training dataset (also released as latitude and longitude coordinate).

The next section described the next steps taken in the data pre-processing before the ML algorithms were implemented.

2. Remove Outliers and Apply Kalman Filter

An outlier in an observation set is a point that is unlike other points in an observation set. It is distinct, rare and does not fit to the other observations set in some way. It can be generally defined as samples that are exceptionally far from the mainstream of the data.

The causes of outliers could include:

1. Measurement or input error
2. Data corruption
3. A true outlier observation

In general, outliers cannot be precisely defined and identified because of the specifics of each dataset. Rather, raw observations must be interpreted and a decision made whether a value is an outlier or not. Therefore, great care should be taken not to hastily remove or change values, especially for small sample size data.

In data pre-processing and cleanup, identifying outliers and bad data is probably one of the most difficult task and it takes time to get it right. Even with a very deep understanding of statistics and how outliers might affect your data, it is always an area to address with caution. This is because it does not mean that the values identified will be outliers and should be removed. A way to approach this may be to plot the identified outlier values against the context of non-outlier values to visualize the possible

systematic relationship or pattern to the outliers. If there are, this may mean that they are not outliers and can therefore, be explained. Furthermore, this can help the outliers themselves to be more systematically identified.

The removal of outliers was performed using a function adapted from [9]. This function computes the difference in distance between the previous coordinate and the next coordinate point. It used this to generate a new column that holds the distance between the previous and next coordinate points and a threshold value is set above which values are regarded as outliers. The threshold values used in our case was set at 50, based on the analyses of the results from the observation of the distance between the coordinates.

a). Kalman Filter Implementation

Kalman filter is applied to the results to smooth the output in order to produce more accurate values. The kalman filter function used is adapted from [10]. The applied Kalman Filter is aimed at improving the baseline slightly.

3. Mean Prediction of the Phone Data

The average of the predictions of several phones in the same data collection from the data provided by Google for the "*High Precision GNSS Positioning on Smartphones Challenge (sponsored by Google)*" was computed.

This was done by generating an interpolated latitude, longitude values for different phone times in the same collection. Linear interpolation was used to achieve this. The generated records to be interpolated were derived using a combination of the time of collection of the data and the phone used. An open source function from https://www.kaggle.com/t88take/gsdc-phones-mean-prediction was used to achieve this. Afterwards, the latitude and longitude predictions were made based on the average of the baseline location predictions of phones in the same collection provided by Google. The open source function used to achieve this is also from https://www.kaggle.com/t88take/gsdc-phones-mean-prediction.

The next step taken was to apply the results from the phone mean method to some selected ML algorithm as described in the next section.

III. BASELINE APPROACH - MACHINE LEARNING IMPLEMENTATION

The goal of this study is to use machine learning techniques to estimate the position produced by the use of a certain smartphone taking into account that currently, mobile phones only offer 3-5 meters of positioning accuracy. The approach used in this study is the baseline approach. We used the baseline location values for the train and test datasets provided by Google. These baseline location values were computed using a simple least square method. The ground truth data were also provided by Google only for the training datasets. These ground truth data were the true latitude and longitude coordinates of the data collected using high end GNSS receivers.

1. Aligning the Ground Truths with Training Data Inputs

The ground truth data were extracted and aligned with training data inputs. This was done in order to have the latitude and longitude coordinates of the training data and the ground truth data in one dataset to be used in training the ML algorithm. Fig. 1 shows a sample of the aligned data.

	collectionName	phoneName	millisSinceGpsEpoch	latDeg	lngDeg	latDeg_truth	lngDeg_truth
0	2020-05-14-US-MTV-1	Pixel4	1273529463442	37.423549	-122.094006	37.423576	-122.094132
1	2020-05-14-US-MTV-1	Pixel4	1273529464442	37.423564	-122.094063	37.423576	-122.094132
2	2020-05-14-US-MTV-1	Pixel4	1273529465442	37.423573	-122.094098	37.423576	-122.094132
3	2020-05-14-US-MTV-1	Pixel4	1273529466442	37.423578	-122.094116	37.423576	-122.094132
4	2020-05-14-US-MTV-1	Pixel4	1273529467442	37.423571	-122.094115	37.423576	-122.094132

Figure 1: Extracting ground truths and aligning with train inputs.

Plotting the training data coordinates against the ground truth can be seen in Fig. 2.

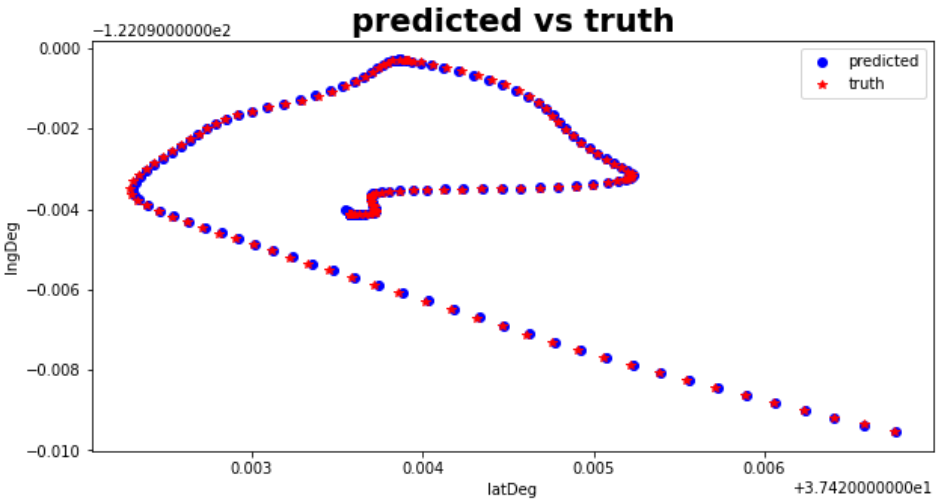


Figure 2: Plotting predicted vs truth coordinates.

2. One-hot Encoding

Usually, in machine learning (ML) it is recommend or required that you prepare your data in specific ways before fitting a machine learning model. Categorical data are variables containing label values instead of numeric values and the number of possible values is often limited to a fixed set. There are often referred to as nominal. A nominal category or a nominal group is a group of objects or data collectively grouped together on the basis of a particular characteristic or a qualitative property [11]. Many ML algorithms cannot directly make use of label data. They require all input variables and output variables to be numeric. This means that categorical data must be converted to a numerical form. For a categorical variable being an output variable, you may want to convert predictions by the model back into a categorical form. This is done in order to use them in some application.

A good way to achieve this is to use a one-hot encoding on categorical data. One-hot encoding refers to the process of splitting the column which contains numerical categorical data to many columns depending on the number of categories present in that column. This will result to each column containing “0” or “1” corresponding to which column it has been placed. However, One hot encoder only takes numerical categorical values, therefore, any value of string type must be label encoded (also called integer encoded) before one-hot encoded. For example, label encoding a gender column could be done as “Female” is 1, “Male” is 2.

The one-hot encoder then gives binary variables often called “dummy variables” in the form:
Female, Male
1 0
0 1

The one-hot encoding was applied to the "phoneName" variable (name of the phone used to collect the data) of the train and test data. The train and test columns were then aligned and in Fig 3, we can see the output of the one-hot encoding. The train and test data after one-hot encoding is illustrated in Fig 4. The next step after one-hot encoding the data is the implementation of the ML algorithms.

```
train_update.tail(2)
```

	Mi8	Pixel4	Pixel4Modded	Pixel4XL	Pixel4XLModded	Pixel5	SamsungS20Ultra
131340	0	0	0	0	0	0	1
131341	0	0	0	0	0	0	1

```
test_update.head(2)
```

	Mi8	Pixel4	Pixel4Modded	Pixel4XL	Pixel4XLModded	Pixel5	SamsungS20Ultra
0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0

Figure 3: Performing one-hot encoding.

```
print("train1_shape:",train1.shape)
train1.columns
train1.head()
```

```
train1_shape: (131342, 11)
```

	latDeg	lngDeg	latDeg_truth	lngDeg_truth	Mi8	Pixel4	Pixel4Modded	Pixel4XL	Pixel4XLModded	Pixel5	SamsungS20Ultra
0	37.423549	-122.094006	37.423576	-122.094132	0	1	0	0	0	0	0
1	37.423564	-122.094063	37.423576	-122.094132	0	1	0	0	0	0	0
2	37.423573	-122.094098	37.423576	-122.094132	0	1	0	0	0	0	0
3	37.423578	-122.094116	37.423576	-122.094132	0	1	0	0	0	0	0
4	37.423571	-122.094115	37.423576	-122.094132	0	1	0	0	0	0	0

```
print("test1-shape:",test1.shape)
test1.columns
test1.head()
```

```
test1-shape: (91486, 9)
```

	latDeg	lngDeg	Mi8	Pixel4	Pixel4Modded	Pixel4XL	Pixel4XLModded	Pixel5	SamsungS20Ultra
0	37.416586	-122.082022	0	1	0	0	0	0	0
1	37.416597	-122.082043	0	1	0	0	0	0	0
2	37.416602	-122.082056	0	1	0	0	0	0	0
3	37.416603	-122.082064	0	1	0	0	0	0	0
4	37.416605	-122.082067	0	1	0	0	0	0	0

Figure 4: Train and test data after one-hot encoding.

3. Implementing ML Algorithms

Several models were used to attempt to capture the relationship between datasets and overall positioning performance. These algorithms includes Bayesian Ridge (BR), logistical regression (LR), and a neural networks (NN). For the BR and LR approach, the one-hot encoding was used to prepare the data. However, for the NN approach, the one-hot was not applied. Each of the models were used and evaluated separately.

Preliminary tests, implementing all regressor algorithms for a quick performance analysis, showed that these were the most promising methods.

a). BR and LR Models

Each model was tuned using the one-hot encoded data, The input and output for the model was created from the latitude and longitude columns of the data. The data is then split into training and validation datasets using the library from sklearn model selection.

The BR model is now fit using the data. Similarly, the **the LR model** is fit using the data. The one-hot encoding is then

reversed for each phone data (one-hot decoding). The validation data is used to evaluate the models as seen in Fig. 5 and Fig. 6 in the results section which shows the evaluation score for each phone (50th, 95th and average percentile) for the BR and LR models respectively. The model is then applied to the complete test data set to predict the latitude and longitude coordinates. The evaluation of these results is done in kaggle.com as part of the Google competition requirements. These models are adapted from [12].

b). The NN model

The **NN model** used here is from the of sklearn python library. The model is adapted from [13].

Feature creation

The features generated are the aggregated features used for training the model. These features includes previous latitude and longitude, the corrected pseudoranges (correctedPrM) for each satellite from derived files shared by Google, and the latitude and longitude difference of the training data from the ground truth. These features are generated for both the train and test datasets.

The correlation between train data baseline coordinates and ground truth coordinates are seen in Fig. 5 below.

	latDeg_truth	lngDeg_truth	latDeg	lngDeg
latDeg_truth	1.000000	-0.89345	1.00000	-0.893449
lngDeg_truth	-0.893450	1.00000	-0.89344	1.000000
latDeg	1.000000	-0.89344	1.00000	-0.893440
lngDeg	-0.893449	1.00000	-0.89344	1.000000

Figure 5: Checking the correlation between current and truth coordinates.

Training the NN Model The NN model training is achieved using a skip connection in Keras on Tensor Processing Unit (TPU). TPU is highly-optimised for large batches and CNNs and has the highest training throughput. It is a custom build ASIC used to accelerate TensorFlow projects. The model summary is shown in Fig. 6 below.

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 41)]         0
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          5376        input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 128)          512         dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          16512       batch_normalization[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 128)          0           dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 128)          16512       dropout[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 128)          512         dense_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 128)          16512       batch_normalization_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128)          0           dense_3[0][0]
__________________________________________________________________________________________________
add (Add)                       (None, 128)          0           dropout_1[0][0]
                                                                 dropout[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 128)          16512       add[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 128)          512         dense_4[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 128)          16512       batch_normalization_2[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 128)          0           dense_5[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 2)            258         dropout_2[0][0]
==================================================================================================
Total params: 89,730
Trainable params: 88,962
Non-trainable params: 768
```

Figure 6: NN model summary.

4. Implementing Weighted Average of BR, LR and NN Models

The results from the three ML algorithms (BR, LR and NN) were integrated using a simple weighted average (SWA) to check on improving the results. A simple weight was used. First, equal weights were assigned to the model (i.e., 0.3333), the results

showed slight improvement compared to the individual ML models. Secondly, it was noticed that when different weights were used, an improvement was seen compared to when the same weights were used.

The different weights were assigned based on the accuracy of the technique. The weight of a certain ML method is the accuracy of that technique divided by the total accuracy (sum of all the models accuracy). Hence, when the average accuracy of a certain ML technique increases, its summation weight would increase. However, the total weight sum is 1.

The results are presented and discussed in table 1 in the next section.

IV. DISCUSSION OF RESULTS

This section presents the results and the evaluations of the results.

1. Evaluating the Models

The evaluation of the model is done for all the implemented ML algorithms.

For the BR and LR models:
The evaluation of the model is done on the validation datasets. The training data was split into train data and validation data. The Fig. 7 and Fig. 8 shows the evaluation of the validation data for the BR and LR models respectively.

For the BR model:

```
Evaluation details:
          phoneName   dist_50    dist_95  avg_dist_50_95
0              Mi8  1.590239   3.615539        2.602889
1           Pixel5  1.812045   7.208850        4.510447
2   Pixel4XLModded  1.379788   3.022998        2.201393
3           Pixel4  1.530467   9.404965        5.467716
4         Pixel4XL  1.384414   4.542033        2.963223
5  SamsungS20Ultra  4.382086  17.265322       10.823704
6     Pixel4Modded  1.798626   3.994491        2.896559


-----------------------------------------------------
Final Evaluation score: 4.495133096204458
-----------------------------------------------------
```

Figure 7: Phone level evaluation score for Bayesian Ridge.

For the LR model:

```
Evaluation details:
          phoneName   dist_50    dist_95  avg_dist_50_95
0              Mi8  1.583664   3.598068        2.590866
1           Pixel5  1.823719   7.209502        4.516611
2   Pixel4XLModded  1.380830   2.996374        2.188602
3           Pixel4  1.506424   9.446001        5.476212
4         Pixel4XL  1.396703   4.545425        2.971064
5  SamsungS20Ultra  4.360654  17.316828       10.838741
6     Pixel4Modded  1.807393   3.981278        2.894336


-----------------------------------------------------
Final Evaluation score: 4.496633060202297
-----------------------------------------------------
```

Figure 8: Phone level evaluation score for Linear Regression.

The Fig. 7 and Fig. 8 show the evaluation score for each phone in the 50th, 95th and average percentile. These are the computed

haversine distance estimation for BR and LR model respectively. The haversine distance estimation is the calculated great circle distance between two points on the earth.

A function was adopted from [14] to get the prediction and ground truth distance estimation (in meters). The inputs are array-like and specified in decimal degrees. This is therefore, the distance between the training/validation data baseline coordinates and the ground truth baseline coordinates.

We can see that the BR model has a better performance than LR model both at the phone level (that is for each phone in most cases) and at the final evaluation score which is the mean of the average distance for the 50th and 95th percentile (avg-dist-50-95).

For the test data, since we do not yet have the ground truth, the pre-evaluation was done using the predicted baseline coordinates and the actual baseline coordinates in the original data.

In the LR model on test data:
The prediction for test data, haversine distance is (latitude/longitude)-
haversine distance (smoothed): [0.90276 4.05308457]

The 2D error is then sqrt(0.9028*pow(2) + 4.0531pow(2)) = 3.6592.

In the BR model on test data:
The prediction for test data, haversine distance is (latitude/longitude)-
haversine distance (smoothed): [0.91345946 4.05016769]

The 2D error is then sqrt(0.9135*pow(2) + 4.0501pow(2)) = 3.6998.

For the NN model: The evaluation of the model is done by calculating the great circle distance between baseline coordinates and the ground truth in the case of the training datasets. For the test data, since the ground truth was not released, the pre-evaluation was done using the predicted baseline coordinates and the actual baseline coordinates in the original data.

The prediction for training data, haversine distance is (latitude/longitude)-
haversine distance (smoothed): [2.14165623 9.83846538]

The prediction for test data, haversine distance is (latitude/longitude)-
haversine distance (smoothed): [0.91779853 5.35367849]

2. Visualize NN Model Training History

We created plots from the collected history data for visualization of NN Model Training History. The plots are from training the model:

1. A plot of model accuracy (Fig. 9).
2. A plot of model loss (Fig. 10).
3. A plot of model learning rate (Fig. 11).

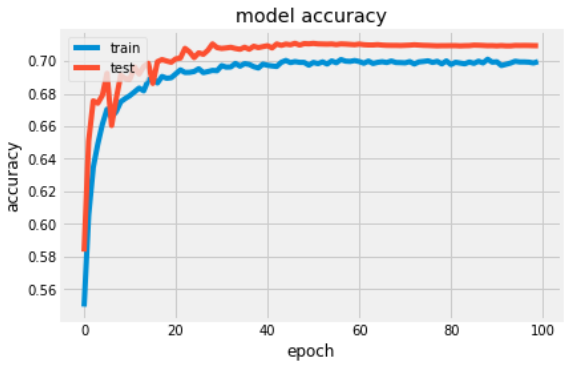


Figure 9: Plot of Model Accuracy on Train and Validation Datasets.

From Fig. 9, we can see that training the model a little more may not be effective as the trend for accuracy on both datasets has stopped rising but rather it moves horizontally for the last few epochs. We can also see that the training dataset has not yet been over-learned by the model, as both datasets show comparable skill.

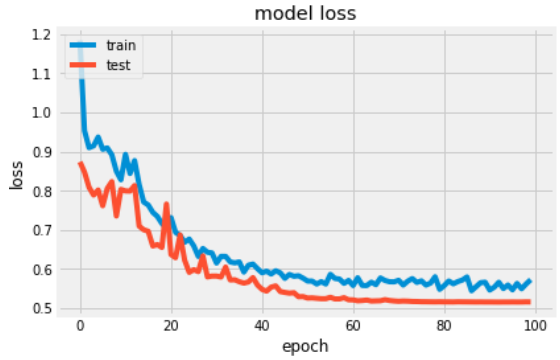


Figure 10: Plot of Model Loss on Training and Validation Datasets.

From Fig. 10, the model loss, we can see that the model performs comparably on both train and validation datasets (labeled as test). If these parallel plots start to show a consistent departure from each other, this could be a good indication to stop training at an earlier epoch.

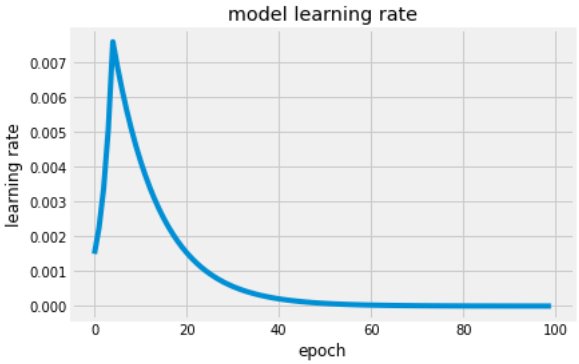


Figure 11: Plot of Model Learning Rate on the Datasets.

From Fig. 11, the model learning rate shows that most of the learning takes place at the first few epochs after which training the model may not lead to any improvement in the learning.

The history for the validation dataset is usually labeled as test by convention as it can be regarded as a test dataset for the model.

In all ML models, the exact evaluation of the models for the test data with the ground truth is done on kaggle.com as part of the google competition.

In Table 1, the results of the preliminary evaluation done on kaggle.com for the test data baseline prediction is presented for each ML model implemented (BR, LR, NN , SWA method).

Table 1: Results from evaluation of test data prediction on kaggle.com

Model List						
					SWA	
	WLS	Phone mean + BR	Phone mean + LR	Phone mean + NN	same weights	different weights
Accuracy in meters (m)	20.49	5.631	5.656	5.743	5.626	5.620

From table 1, we can see that the SWA performs better, followed by the BR, LR and NN model respectively. In the case of the SWA, when the average accuracy of a certain ML technique increases, its summation weight would increase, thereby, increasing the accuracy. This is seen in the accuracy improvement when different weights are used compared to using the same weights.

V. CONCLUSION AND FUTURE WORK

The data used in this paper was provided by Google under a competition hosted on kaggle.com for the ION GNSS+ 2021 conference. The Global Navigation Satellite System (GNSS) provides raw signals, which the GNSS chipset in smartphones uses to compute a position. The accuracy are usually not good enough and can create a "jumpy" experience for many applications relying on localization.

In this paper, we presented an algorithm that can improve the precision GNSS positioning and navigation accuracy on smartphones using a machine learning approach. First, we analyzed the data and pre-processed it for the model by identifying and removing outliers. Then we applied a "phone mean method" to the data in combination with some pre-selected ML models namely, bayesian ridge, linear regression, NN model and SWA method respectively to predict the latitude and longitude coordinates. The results show an improvement in the GNSS positioning and navigation accuracy when evaluated against the ground truth of the data collected using high end GNSS receivers for the same locations. The positioning accuracy obtained is in the range of 5.620 - 5.743 m (meters). This is an improvement to the WLS of 20.49 meters positioning accuracy.

In future publications (on-going research), we will develop a precision positioning algorithm for multi-constellation dual-frequency global navigation satellite systems (GNSS) receivers that would select a subset of satellites out of the tracked ones to achieve improved location accuracy with the help of machine learning. Receivers (especially low-cost receivers and smartphones) often have very limited channels and computational resources, therefore, the complexity of the algorithm needs to be kept low. We will use both the GDOP and measurement errors as criteria for satellite selection. From initial analysis, the trained models will effectively predict pseudorange and delta range residuals and also select the satellites making the most contribution to the desired GDOP value. The optimization of GDOP and the ability to filter out erroneous pseudorange and delta range measurements can lead to improved location accuracy. We will also work on integrating "outlier removal" to Kalman filters (KF) because if they are not removed, they would disturb linear KF heavily. Furthermore, we will first make use of the available raw GNSS data (including pseudorange rates, accumulated delta range - aka carrier phase, and base station data), and IMU data to generate a much better position (than WLS used in this study) and then the results will be further improved with ML.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Google for providing the datasets used in this study and give credits to some kaggle.com notebooks adapted in this study as reference in this paper.

REFERENCES

[1] P. Huang, C. Rizos, and C. Roberts, "Machine learning algorithm to forecast ionospheric time delays using global navigation satellite system observations," *GPS Solutions*, vol. 22, pp. 221–231, 2018. [Online]. Available: https://doi.org/10.1007/s10291-018-0776-0

[2] N. I. Ziedan, "Multipath and nlos signals identification and satellite selection algorithms for multi-constellation receivers," *Proceedings of the 29th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2016)*, pp. 521–533, 2018. [Online]. Available: https://doi.org/10.33012/2016.14844

[3] A. M. El-naggar, "An alternative methodology for the mathematical treatment of gps positioning," *Alexandria Engineering Journal*, vol. 50, no. 4, pp. 359–366, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016811000470

[4] E. Wang, C. Jia, S. Feng, G. Tong, H. He, P. Qu, Y. Bie, C. Wang, and Y. Jiang, "A new satellite selection algorithm for a multi-constellation gnss receiver," *in Proceedings of the 31st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2018), Miami, Florida*, pp. 3802–3811, 2018. [Online]. Available: https://doi.org/10.33012/2018.15993

[5] P. F. Swaszek, R. J. Hartnett, K. C. Seals, and R. M. Swaszek, "Tighter gdop bounds and their use in satellite subset selection," *in Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, pp. 637–649, 2019. [Online]. Available: https://doi.org/10.33012/2019.16869

[6] L. Mallika I, D. V. Ratnam, S. Raman, and G. Sivavaraprasad, "Machine learning algorithm to forecast ionospheric time delays using global navigation satellite system observations," *Acta Astronautica*, vol. 173, pp. 221–231, 2020. [Online].

Available: https://www.sciencedirect.com/science/article/pii/S0094576520302630

[7] A. Siemuri, H. Kuusniemi, M. S. Elmusrati, P. Välisuo, and A. Shamsuzzoha, "Machine learning utilization in gnss—use cases, challenges and future applications," in *2021 International Conference on Localization and GNSS (ICL-GNSS)*, 2021, pp. 1–6.

[8] G. M. Fu, M. Khider, and F. van Diggelen, "Android raw gnss measurement datasets for precise positioning," in *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+2020)*, September 2020, pp. 1925–1937.

[9] @t88take. (2021) Gsdc phones mean prediction. [Online]. Available: https://www.kaggle.com/t88take/gsdc-phones-mean-prediction

[10] M. Bodych. (2021) Demonstration of the kalman filter. [Online]. Available: https://www.kaggle.com/emaerthin/demonstration-of-the-kalman-filter

[11] G. Rugg and M. Petre, *A Gentle Guide To Research Methods*. Open University Press, 2006, no. ISBN 9780335219278. [Online]. Available: https://books.google.com/books?id=AFDIsoyH6MIC&pg=PA182

[12] Ramswaroop. (2021) Baseline appraoch - linear regression. [Online]. Available: https://www.kaggle.com/ramswaroopbhakar14/baseline-appraoch-linear-regression

[13] J.-Y. Lee. (2021) Google smartphone decimeter eda + keras (tpu). [Online]. Available: https://www.kaggle.com/jeongyoonlee/google-smartphone-decimeter-eda-keras-tpu#Model-Training

[14] stackoverflow.com. (2021) How can i quickly estimate the distance between two (latitude, longitude) points? [Online]. Available: https://stackoverflow.com/questions/15736995/how-can-i-quickly-estimate-the-distance-between-two-latitude-longitude-points#

Publication III

PEER REVIEWED

Application of Machine Learning to GNSS/IMU Integration for High Precision Positioning on Smartphones

Akpojoto Siemuri[1], Mahmoud Elsanhoury[1], Petri Välisuo[1], Heidi Kuusniemi[1,2], Mohammed S. Elmusrati[1]
[1] *School of Technology and Innovations, University of Vaasa, Finland*
[2] *Finnish Geospatial Research Institute, National Land Survey, Finland*

BIOGRAPHY

Akpojoto Siemuri received his B.Sc.(tech) degree in electrical and computer engineering from Federal University of Technology Minna, Nigeria in 2010, a M.Sc.(tech) degree in wireless industrial automation with a minor study in industrial management from the University of Vaasa, Finland in 2019. He is currently pursuing a Ph.D. degree in automation technology at the University of Vaasa. From 2018 to 2019, he was a Research Assistant in the Smart Energy Systems Research Platform (SESP) Project at the University of Vaasa, Finland. He is currently a Project Researcher in Digital Economy Research Platform, University of Vaasa. His research interest includes machine learning, GNSS technologies, smart devices, embedded systems, communication systems, and game theory.

Mahmoud Elsanhoury is currently pursuing the doctoral (Ph.D) degree in computer science and telecommunications engineering at the University of Vaasa, Finland. He received his M.Sc. (tech) degree in telecommunications engineering from Vaasa University in 2018, and the B.Sc. degree in telecommunications engineering from Alexandria University, Egypt in 2013. His current research interests cover ubiquitous indoor positioning systems, ultra-wideband (UWB) indoor localization, low-earth orbit (LEO) satellites for positioning, multisensor fusion technologies, Kalman filters, uncertain stochastic processes and machine learning algorithms.

Petri Välisuo is currently working as an Associate Professor (tenure track), sustainable automation, in the School of Technology and Innovation Management of University of Vaasa, Finland. He received M.Sc.(tech) degree in computer science from the Tampere University of Technology, Finland, and D.Sc.(tech) degree in automation technology from University of Vaasa, in years 1996 and 2011 respectively. He has authored and co-authored 27 peer reviewed and more than 10 other scientific publications. His research interests cover machine learning, IoT, positioning methods and other technologies relevant to industrial automation. He has been working 10 years in telecommunication industry before the research career at the University of Vaasa.

Dr. Heidi Kuusniemi is a professor in computer science and director of Digital Economy at the University of Vaasa in Finland. She is also a part-time research professor in satellite navigation at the Finnish Geospatial Research Institute. She has a M.Sc. (Tech.) degree (with distinction) from 2002 and a D.Sc. (Tech.) degree from 2005 in information technology, respectively, from Tampere University of Technology, Finland. She served as a member of the council of natural sciences and technology at the Academy of Finland 2019-2021 and was a member of the scientific advisory committee for GNSS (GSAC) at ESA. Her technical expertise and interests include GNSS reliability and resilience, estimation and data fusion, mobile precision positioning, indoor localization and PNT in new space.

Prof. Mohammed S. Elmusrati received his B.Sc. (with honors) and M.Sc. (with high honors) degrees in electrical and electronic engineering, University of Benghazi, Libya, in 1991 and 1995, respectively, and the Licentiate of Science in technology (with distinction) and the Doctor of Science in Technology (D.Sc.) degrees in automation and control engineering from Aalto University Finland, in 2002 and 2004, respectively. Currently, he is Full Professor and Head of the Digitalization Unit at the School of Technology and Innovations – University of Vaasa, Finland. His research interest includes wireless communications, artificial intelligence, machine learning, biotechnology, big data analysis, stochastic systems, and game theory. Elmusrati has published more than 130 papers, books, and book chapters. Prof. Elmusrati is an active member in different scientific societies such as Senior Member at IEEE, Member at Society of Industrial and Applied Mathematics (SIAM), and Member at Finnish Automation Society.

 https://doi.org/10.33012/2022.18375

ABSTRACT

This paper describes our solution for the Google smartphone decimeter challenge (GSDC), which was held from May to August 2022. The GSDC is a competition for improving positioning accuracy of smartphones. The global navigation satellite system (GNSS) data from smartphones have lower signal levels and higher noise in GNSS observations compared to commercial GNSS receivers. Therefore, it is difficult to directly apply the existing GNSS high-precision positioning methods like precise point positioning (PPP) and real-time kinematic (RTK). The smartphones used to collect the raw GNSS data have multi-constellation, dual-frequency GNSS receivers, and Inertial Measurement Unit (IMU) sensors. Multi-sensor fusion technology has become very prominent for seamless navigation systems due to its complementary capabilities to GNSS positioning. In this work, we developed a machine learning (ML) based adaptive positioning approach to estimate the positions of the smartphone by utilizing post-processed kinematic (PPK) precise positioning techniques to process the GNSS datasets. The ML model is used to predict the driving paths (highways, tree-lined streets, or downtown areas). Depending on the predicted driving path, PPK technique uses the carrier phase to compute the user position using differential corrections from known GNSS base stations. We then use of the Rauch–Tung–Striebel (RTS) smoother, which consists of a forward pass Kalman Filter (KF) and a backward recursion smoother to achieve a loosely coupled integration of GNSS and IMU measurements for positioning estimation of the smartphone. We refer to this method as LC-GNSS/IMU/ML using ML based adaptive positioning (MAP) real-time kinematic (RTK) post-processing algorithm (MAP RTK). This method is validated using reference data from GNSS survey-grade receivers provided with the training datasets. The final validation of this proposed method is done on Kaggle.com, the host of the GSDC competition. Using the proposed method, we estimated the location of the smartphone and tackled the competition. The final public score was 2.61 m, while the final private score was 2.29 m.

Keywords - Adaptive positioning; Machine Learning; Kalman Filter; Rauch–Tung–Striebel smoother; Inertial Measurement Units; Global Navigation Satellite Systems; Post-processed kinematic; Smartphones

I. INTRODUCTION

Multi-constellation global navigation satellite systems (GNSS) provide benefits such as the availability of more visible satellites that can be used to improve user positioning performance [1]. This is useful in challenging environments where GNSS signals could be partially or totally blocked and are affected by multi-path reflections, for example, in urban areas or places with dense foliage, this becomes very useful. Accuracy greater than 100 meters in the worst non-line-of-sight condition can improve to between 5 meters and sub-meter depending on the technique used for data collection and the receiver used (survey-grade or low-cost receivers). In 2016, the Android operating system released an application program interface to access raw GNSS measurement data from GNSS installed in smartphones [2]. As a result, high-precision positioning at the decimeter and centimeter levels on smartphones has attracted much attention [3, 4]. There have been growing interests in the use of low-cost receivers and GNSS chipsets on smartphones for positioning applications. However, they do not provide very precise accuracy as the high grade (survey-grade receivers), especially due to antenna constraints. Differential correction, a data collection technique, can be used to remove errors in GNSS data created by selective availability, ionospheric delay, tropospheric delay, and ephemeris errors. There are other factors that lead to error in positioning, such as multi-path propagation which causes ranging measurement errors [5]. Other factors (less correctable) that create an error in GNSS data include the large distance between the rover and the base station (about 10 mm degradation with every kilometer away from the base station when differential correction is considered), low SNR (signal to noise ratio), and low satellite elevation [5].

GNSS provides raw signals, which the GNSS chipsets in smartphones use to compute a position. Code phase utilization is one processing technique that gathers data via a C/A (coarse acquisition) code receiver, which uses the information contained in the satellite signals (aka the pseudo-random code) to calculate positions. After differential correction, this processing technique can result in 1-5 meter accuracy. Carrier phase utilization is another processing technique that gathers data via a carrier phase receiver, which uses the radio signal (aka carrier signal) to calculate positions. The carrier signal, which has a much higher frequency than the pseudo-random code, is more accurate than using the pseudo-random code alone. After differential correction, this processing technique can result in sub-meter accuracy depending on the GNSS receiver. The data used in this paper is from the Google smartphone decimeter challenge (GSDC). In GSDC, each dataset includes raw GNSS measurements collected in the US San Francisco Bay and other areas by several Android smartphone devices, together with the ground truth trajectories collected by high-grade GNSS and inertial navigation unit (IMU) integration system for reference.

Another point of interest is how machine learning (ML) can be used to improve the positioning of smartphones. ML is a very powerful tool in processing time-series data, as it can be applied in learning time-dependent patterns across multiple models. It is useful in analyzing hidden and unknown patterns and information in GNSS data. GNSS is a source of affordable big data useful for ML exploitation. Also, data storage and the growth of less expensive and more powerful processing capabilities have propelled the growth of ML [6]. The aim of using ML is not to generate an explicit formula for the distribution of the data; however, it is used to train an algorithm to detect the relationships between the features of a data set, directly from the data.

In this paper, we describe our solution used in this competition in which there were over 571 teams from all over the world

working on high-precision positioning of smartphones.

II. STRATEGY

Real-time kinematic (RTK)-GNSS technique, which is usually used for high-precision positioning, is difficult to implement in smartphones due to limitations of their antennas. This is because it is difficult to solve the integer ambiguities in the carrier phase measurements using smartphone antenna. Even in an open-sky environment with an accuracy of 3 m it is still difficult to reach decimeter level since there is a lot of noise and outliers in the observations. Therefore, a robust position estimation method is needed. In our work, we still attempt to make use of RTK-GNSS technique or in our case, post-processed kinematic (PPK) positioning techniques to process the GNSS datasets. A ML model is used to predict the driving paths (highways, tree-lined streets, or downtown areas) for adaptive positioning using PPK. In the PPK technique, the carrier phase is used to compute the user position making use of differential corrections. We then implement loosely coupled integration of GNSS and IMU measurements for position estimation, using Kalman filter (KF) algorithm and Rauch–Tung–Striebel (RTS) smoother. The initial results show improvements in the positioning accuracy. GNSS low-cost receivers often have limited channels and computational resources, therefore, the complexity of the algorithm had to be kept modest. Further validations will be performed to ensure the integrity of the proposed loosely coupled GNSS/IMU/ML (LC-GNSS/IMU/ML) method.

The steps or approach taken are as follows:

1. Data Analysis and Preparation
2. ML based prediction of driving path
3. PPK precise positioning techniques to process the GNSS datasets
4. GNSS/IMU integration making use of KF/RTS-based method

III. DATA ANALYSIS AND PREPARATION

The performance of smartphones satellite observation indicates that they cannot track satellites stably, also they can only observe a small number of satellites continuously with dual-frequency signals. Furthermore, the quality of the satellites observed by smartphones generally have a low carrier-to-noise ratio (C/NR) [7]. Multipath is another factor that significantly impaired the positioning accuracy of smartphones. Therefore, we first analyze the GNSS data of smartphones provided by GSDC to understand the observation quality of GNSS data.

At the ION GNSS+ 2022, Google released a total of 206 traces in the challenge, whose collection process is described in a paper published in the proceedings of ION GNSS+ 2020 [2]. The data were released twice, first the training data and next was the test data. The training datasets consisted of GnssLogger files, RINEX observation files, ground truth files, and device IMU/GNSS data files (GNSS intermediate values derived from raw GNSS measurements, provided for convenience). The test datasets include the same types of data and follow the same convention as the training dataset, except for the ground truth data that are not provided. The results of test datasets will be used as the prediction of the expected ground truth and will be evaluated using the unreleased ground truth. In both the training and test datasets, the derived values can be used to compute a corrected pseudorange which is a closer approximation to the geometric range from the phone to the satellite.

The driving paths of the provided GNSS data can be classified into three main categories: highways, tree-lined streets, or downtown areas. The train dataset ground truth track is shown in Figure 1. These pools of GNSS and IMU datasets collected from smartphones can be useful in developing high precision GNSS positioning using the accompanied high accuracy ground truth from high-grade GNSS receivers. Table 1 shows an overview of the weighted least square (WLS) positioning results, provided by the competition organizer for three categories of driving environments. The smartphone observed pseudorange is much noisier compared to commercial GNSS receiver, and GNSS code positioning using only pseudorange has limited positioning accuracy.

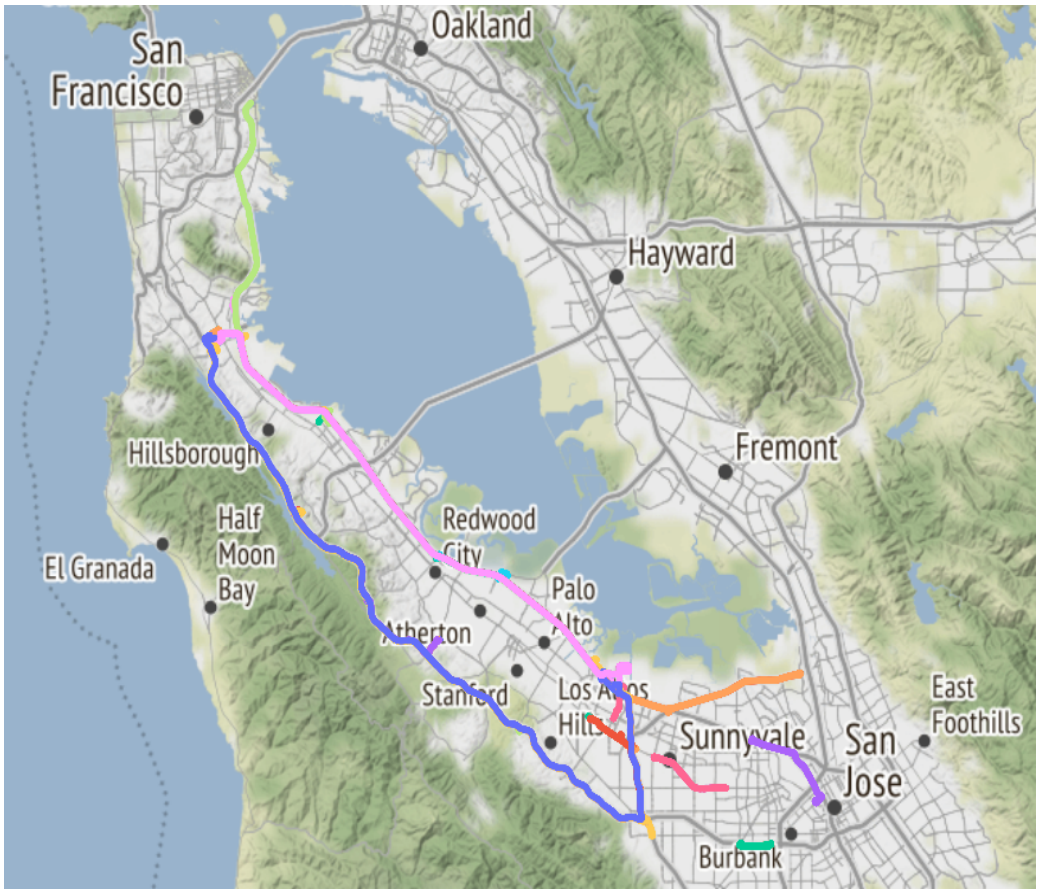


Figure 1: Driving trajectories included in training and some test data provided by GSDC.

The driving trajectories or paths are from the training data reference positions (ground truth) of the training data. Some new driving paths were included in the test data that are not present in the training data.

Table 1: Horizontal positioning error of the WLS baseline position in each driving path

Path	Highway	Tree-lined Street	Semi-Downtown
WLS baseline score	**4.3134 [m]**	**8.2553 [m]**	**8.0413 [m]**

IV. RELATED RESEARCHES

Precise point positioning using carrier phase measurements has been actively studied in the GNSS field. In [8] precise point positioning using carrier phase measurements with graph optimization is implemented and compared with Kalman filter based implementation. The use of RTK position estimation combined with IMU and other sensors have been studied [9, 10]. These papers showed an improvement in positioning accuracy compared to least-squares-based positioning. GNSS position estimation is highly dependent on the environment where the GNSS receiver is operating (open sky or urban canoys). In [3], the area of data collection was considered during position estimation. This paper is unique because it uses ML to predict the driving path of the data for an adaptive RTK position estimation. The Rauch–Tung–Striebel (RTS) smoother is used to achieve a loosely coupled integration of the GNSS carrier-phase and IMU measurements for positioning estimation in the smartphone.

V. PROPOSED METHOD

1. ML based Adaptive processing

Machine learning (ML) is applied to improve the positioning estimation of the KF/RTS-based GNSS/IMU integration using the ground truth data provided with GSDC training datasets. These ground truth data were collected using high-end survey-grade GNSS receivers. The driving paths of the provided GNSS data can be classified into three categories: highways (open sky) with line-of-sight (LOS), tree-lined streets, and downtown areas with multi-path/Non-LOS (NLOS). The training dataset is used to train the ML model to predict the driving paths. Different approaches has to be used in different areas. Especially in the downtown area where multipath signals caused by reflections and diffractions can significantly degrade the GNSS observations. Based on the predicted driving path, different PPK configurations are used to process the data for improved solutions. Parameters such as elevation angle of the satellite, C/NR, etc are considered based on the predicted driving path. This is done because GNSS measurements are easily disturbed by external influence such as multi-path which can be experienced in tree-lined streets, and downtown areas.

2. PPK precise positioning techniques

Real-time kinematic positioning (RTK) is the application of surveying to correct for common errors in current satellite navigation (GNSS) systems. It uses carrier phase measurements and the information content of the signal and relies reference station or interpolated virtual station to provide real-time corrections, which can give up to centimetre-level accuracy. In our case we use the post-process kinematic (PPK) for position estimation since the data is not collected in real-time. After the prediction of the driving path using the ML model, the data are then processed using PPK precise positioning techniques. The way the data is processed depends on the predicted driving path. First, we convert each phone GNSSLogger raw data file into a RINEX file that can then be processed with RTKLIBS programs [11]. RTKPOST [11] was first used to get an understanding of a couple of datasets and the right configurations to used to get a good PPK solution. Afterward, we batch process all of the data sets to get the position estimation. In the PPK technique, carrier phase data is used to compute the user position. The use of PPK requires differential correction, therefore some base observation were used to download raw observation data for the appropriate dates and times from some nearby CORS stations using NOAA National Geodetic Survey website. The SLAC, P222, and VDCY stations are used because they were reasonably close to all of the data collection rides. The satellite broadcast navigation data (BRDM files) for each data set was also downloaded from the International GNSS Service website including the clock navigation data, and satellite precise orbits [12]. These are used during the post-processing of the GNSS data.

3. Hardware clock bias

In GSDC dataset there are a couple of data sets that have corrupted carrier phase measurements. The initial RTKLIB solutions for these datasets are quite poor and are typicaly worse compared to the included Google baseline solutions which uses the WLS for pseudo-range measurements. These data sets can be identified by the *HardwareClockDiscontinuityCount* field in the raw Androiod log files [13]. If the final value in this field is larger than the initial value, then the carrier phase measurements will be corrupted by the clock discontinuities. The log files of the datasets are scanned using an algorithm to identify any with greater than one discontinuity[12]. The solutions for the listed dataset use the robust WLS solution in place of the poor RTK solutions [14].

4. GNSS/IMU integration using RTS smoothing algorithm

The Inertial Measurement Unit (IMU) is frequently used in robotics, vehicles, and, of course, in mobile phones. An IMU is usually made up of an accelerometer, gyroscope, and magnetometer. Tactical grade IMU is fully calibrated compared to low-cost ones which come with some factory calibrations stored in the IMU registers. The calibration only covers the scaling factors of each axes. The accelerometer measures the acceleration of a movement, the gyroscope measures the angular speed of the sensor, and the magnetometer measures the Earth's local magnetic field direction and magnitude.

Sensor fusion also referred to as data fusion, information fusion, or multi-sensor data fusion is used in the fusion of data from the accelerometer, magnetometer, and gyroscope. The fusion is performed to obtain position estimates and mitigate overall errors. Kalman filter is used for IMU and data fusion whereas numerical integration is required to obtain the position estimates. Basically, accelerometer values are fed to a dynamic model where they were numerically time-integrated twice to get the position, with the discretization and linearization (approximation) phases. The angular velocities measured from the gyroscope can be integrated with respect to time used to obtain the pitch and roll angles, which are later employed in the dynamic model to serve in the general state vector as fine-tuning values for the IMU localization.

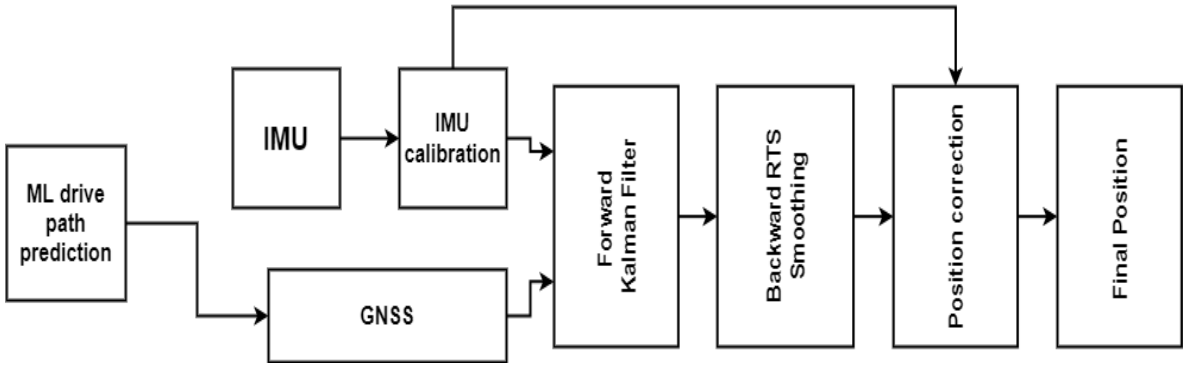


Figure 2: The flow diagram of MAP RTK post-processing algorithm.

GNSS measurements are easily disturbed by external influence such as multipath but the integration of GNSS with IMU is beneficial as the advantages and disadvantages of GNSS and IMU complement each other to enable accurate measurements in challenging areas. They are three types of IMU and GNSS integration namely, loosely coupled, tightly coupled, and ultra-tight coupled integration [15]. In this research we used the loosely coupled integration with KF/RTS, to integrate GPS and IMU in one system. We use the acceleration measured by IMU for inertial navigation calculation to get the data of position. We use them together with the information of GNSS for integration. The result is used as a prior observation of Kalman filtering while the covariance matrix and state transition matrix gained from forward Kalman filtering process is used for the reverse RTS smooth to the state estimate. The Rauch–Tung–Striebel (RTS) smoother, consists of a forward pass Kalman Filter (KF) and a backward recursion smoother to achieve the loosely coupled integration of GNSS and IMU measurements for positioning estimation in the smartphone. The forward classical Kalman filter is used to estimate the state of each moment, while the backward filter is to obtain more accurate state estimate on the basis of forward filter [16]. The flow diagram of MAP RTK post-processing algorithm is shown in Figure 2.

The state vector of the Kalman filter comprises both GNSS (longitute, latitude) and IMU (accelerometer) epochs, as follows in equation 1:

$$\mathbf{x} = [p^x \quad v_x \quad a_x \quad p^y \quad v_y \quad a_y]^T \tag{1}$$

where p^x, p^y are the (longitude, latitude) positions gained from GNSS. v_x, v_y are the non-measured speeds, and a_x, a_y are the IMU accelerations, respectively. Hence, the measurements are p^x, p^y and a_x, a_y. This makes the attributes of Kalman filter as follows in equation 2:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The Newtonian equations of motion for predicting the iterative position, velocity and acceleration are:

$$x_{k+1} = x_k + \Delta t \dot{x}_k + \frac{\Delta t^2}{2} \ddot{x}_k + \frac{\Delta t^3}{6} \dddot{x}_k$$

$$\dot{x}_{k+1} = \dot{x}_k + \Delta t \ddot{x}_k + \frac{\Delta t^2}{2} \dddot{x}_k$$

$$\ddot{x}_{k+1} = \ddot{x}_k + \Delta t . \dddot{x}_k$$

Where Δt is the time step, the state transition F matrix becomes:

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The Kalman algorithm proceeds as follows:

$$\begin{aligned} \mathbf{x_{k+1}} &= \mathbf{F_k x_k} + Q_k \\ \mathbf{y_k} &= \mathbf{H_k x_k} + R_k \end{aligned} \tag{4}$$

where Q_k, R_k are the process and measurements covariance matrices, and x_{k+1} and y_k are the new states update and the generated measurements update, respectively.

These filtered a-priori and a-posteriori state estimates $\hat{X}_k|k-1, \hat{X}_k|k$ and the covariances $\hat{P}_k|k-1, \hat{P}_k|k$ are saved for use in the backwards pass (for retrodiction). The smoothed state estimates $\hat{X}_k|n$, and covariances $\hat{P}_k|n$ are computed for the backward pass. The below recursive equations are used starting from the last time step and proceeding backwards in time [17].

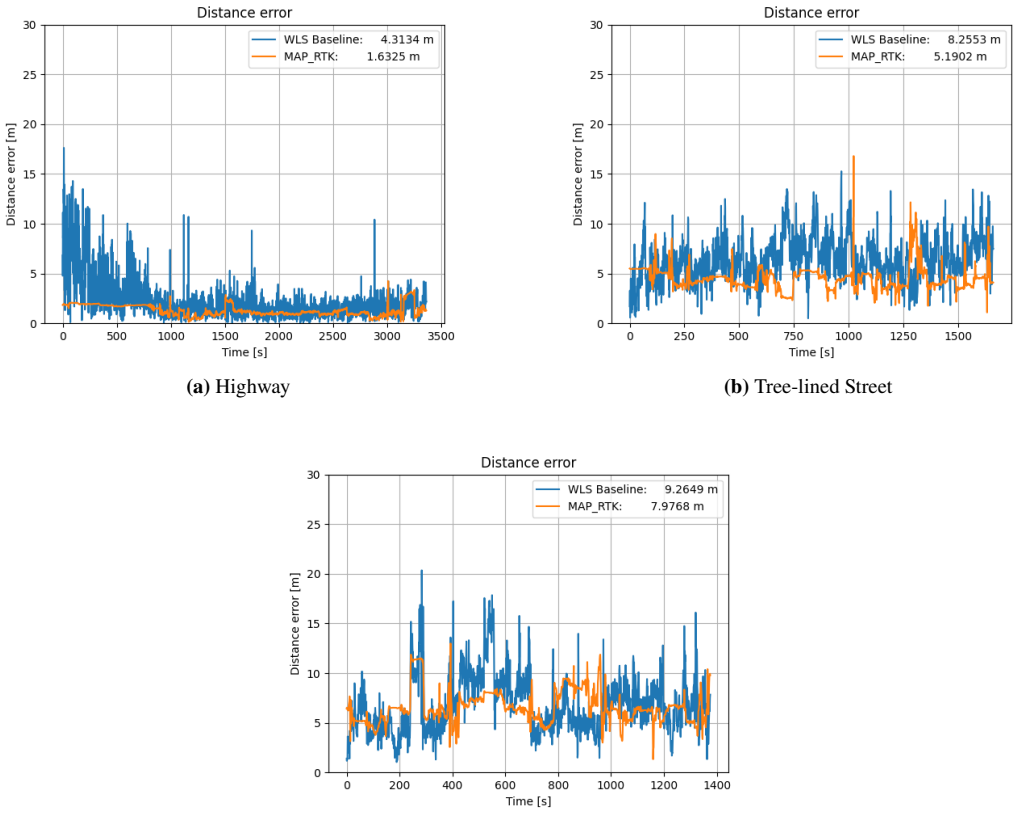
$$\hat{X}_k|_n = \hat{X}_k|_k + C_k(\hat{X}_{k+1}|_n - \hat{X}_{k+1}|_k) \tag{5}$$

$$P_k|_n = P_k|_k + C_k(\hat{X}_{k+1}|_n - \hat{X}_{k+1}|_k) \times C_k^T \tag{6}$$

where $C_k = P_k|_k F_{k+1}^T P_{k+1}^{-1}|_k$

$X_k|_k$ is the a-posteriori state estimate of timestep k and $X_{k+1}|_k$ is the a-priori state estimate of timestep $k+1$ which also applies to the covariance.

VI. EXPERIMENTS



(a) Highway

(b) Tree-lined Street

(c) Downtown

Figure 3: Comparison of positioning error between the WLS baseline and proposed method for each driving paths

The proposed method was evaluated using reference locations provided with the training dataset. In our studies, we made use of the carrier phase information of the data. The results of our model compared to the WLS solutions provided for the GSDC with respect to the ground truth is shown below. We can see an improvement in the estimated position for the respective driving paths as shown in Figure 3.

VII. RESULTS

We estimated the location of the smartphone and made evaluation of our method on the Kaggle platform. In our studies, we used the carrier phase information of the data. As a starting point, using GNSS only solution, initial results using our ML adaptive PPK positioning techniques on the 36 GSDC test datasets gave an accuracy of 2.62 meters for the public score on Kaggle platform. After the loosely coupled integration of the IMU sensor with GNSS navigation solutions, the score became 2.29 meters. From this result, we have shown that the proposed method (LC-GNSS/IMU/ML using MAP RTK) can be used to improve the position estimation of smartphones with relatively high accuracy.

VIII. CONCLUSION

Global Navigation Satellite System (GNSS) provides raw signals, which the GNSS chipsets in smartphones use to compute a position. The positioning accuracy provided by mobile phones is about 3-5 meters of positioning accuracy. This may be useful in some applications but in many cases, this can create a "jumpy" experience for many mobility applications relying on localization. In this paper, we employ a new LC-GNSS/IMU/ML method which makes use of the MAP RTK post-processing algorithm to process carrier-phase information, IMU sensors to improve the position estimates of smartphones. The results obtained show improvements in the positioning accuracy. Using the proposed method, we estimated the location of the smartphone and tackled the competition. The final public score was 2.61 m, while the final private score was 2.29 m. GNSS low-cost receivers and smartphones often have limited channels and computational resources, therefore, the complexity of the algorithm had to be kept modest. Further validations will be performed to ensure the integrity of the proposed method. In the future, we will look into factor graph and robust outlier detection for improvement of our solution.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Google for providing the datasets used in this study and give credits to some kaggle.com notebooks adapted in this study as referenced in this paper.

REFERENCES

[1] P. Huang, C. Rizos, and C. Roberts, "Machine learning algorithm to forecast ionospheric time delays using global navigation satellite system observations," *GPS Solutions*, vol. 22, pp. 221–231, 2018. [Online]. Available: https://doi.org/10.1007/s10291-018-0776-0

[2] G. M. Fu, M. Khider, and F. van Diggelen, "Android raw gnss measurement datasets for precise positioning," in *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+2020)*, September 2020, pp. 1925–1937.

[3] T. Suzuki, "First place award winner of the smartphone decimeter challenge: Global optimization of position and velocity by factor graph optimization," in *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+2021)*, September 2021, pp. 2974–2985.

[4] X. Zhang, X. Tao, F. Zhu, X. Shi, and F. Wang, "Quality assessment of gnss observations from an android n smartphone and positioning performance analysis using time-differenced filtering approach," in *Gps Solutions*, vol. 22, 2018, pp. 1–11.

[5] K. Elliott D. and H. Christopher J., *Understanding GPS/GNSS: Principles and Applications*. Artech House, 2017.

[6] A. Siemuri, H. Kuusniemi, M. S. Elmusrati, P. Välisuo, and A. Shamsuzzoha, "Machine learning utilization in GNSS—use cases, challenges and future applications," in *2021 International Conference on Localization and GNSS (ICL-GNSS)*, 2021, pp. 1–6.

[7] K. Zhang, F. Jiao, and J. Li, "The assessment of gnss measurements from android smartphones," in *China Satellite Navigation Conference, Springer*, vol. 22, 2018, p. 147–157.

[8] R. M. Watson and J. N. Gross, "Evaluation of kinematic precise point positioning convergence with an incremental graph optimizer," in *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2018, p. 589–596.

[9] D. Chen and G. X. Gao, "Probabilistic graphical fusion of lidar, gps, and 3d building maps for urban uav navigation," in

Navigation, Journal of the Institute of Navigation, vol. 66, 2019, p. 151–168.

[10] W. Li, X. Cui, and M. Lu, "A robust graph optimization realization of tightly coupled gnss/ins integrated navigation system for urban vehicles," in *Tsinghua Science and Technology*, vol. 23, no. 6, 2018, p. 724–732.

[11] T. Takasu and A. Yasuda, "Development of the low-cost rtk-gps receiver with an open source program package rtklib," in *The International Symposium on GPS/GNSS, Jeju, Korea*, 2009.

[12] E. Tim. (2021) Batch processing rtklib solutions with rnx2rtkp and python. [Online]. Available: https://rtklibexplorer.wordpress.com/2022/01/05/batch-processing-rtklib-solutions-with-rnx2rtkp-and-python/

[13] G. Developers. (2022) Gnss clock. [Online]. Available: https://developer.android.com/reference/android/location/GnssClock

[14] taroz1461. (2022) Carrier smoothing + robust wls + kalman smoother. [Online]. Available: https://www.kaggle.com/code/taroz1461/carrier-smoothing-robust-wls-kalman-smoother

[15] Cahyadi, Mokhamad Nur and Rwabudandi, Irene, "Integration of GNSS-IMU for increasing the observation accuracy in condensed areas (infrastructure and forest canopies)," *E3S Web Conf.*, vol. 94, p. 03015, 2019. [Online]. Available: https://doi.org/10.1051/e3sconf/20199403015

[16] S. Tian, C. Jiabin, S. Chunlei, and Y. Huan, "The application of r-t-s smoothing algorithm in the post-processing of the integrated navigation," in *2017 29th Chinese Control And Decision Conference (CCDC)*, 2017, pp. 197–201.

[17] H. E. RAUCH, F. TUNG, and C. T. STRIEBEL, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965. [Online]. Available: https://doi.org/10.2514/3.3166

Publication IV

A Systematic Review of Machine Learning Techniques for GNSS Use Cases

AKPOJOTO SIEMURI, Student Member, IEEE
KANNAN SELVAN
University of Vaasa, Vaasa, Finland

HEIDI KUUSNIEMI, Member, IEEE
University of Vaasa, Vaasa, Finland
Finnish Geospatial Research Institute, National Land Survey, Finland

PETRI VALISUO
MOHAMMED S. ELMUSRATI, Senior Member, IEEE
University of Vaasa, Vaasa, Finland

In terms of the availability and accuracy of positioning, navigation, and timing (PNT), the traditional Global Navigation Satellite System (GNSS) algorithms and models perform well under good signal conditions. In order to improve their robustness and performance in less than optimal signal environments, many researchers have proposed machine learning (ML) based GNSS models (ML models) as early as the 1990s. However, no study has been done in a systematic way to analyze the extent of the research on the utilization of ML models in GNSS and their performance. In this study, we perform a systematic review of studies from 2000 to 2021 in the literature that utilizes machine learning techniques in GNSS use cases. We assess the performance of the machine learning techniques in the existing literature on their application to GNSS. Furthermore, the strengths and weaknesses of machine learning techniques are summarized. In this paper, we have identified 213 selected studies and ten categories of machine learning techniques. The results prove the acceptable performance of machine learning techniques in several GNSS use cases. In most cases, the models using the machine learning techniques in these GNSS use cases outperform the traditional GNSS models. ML models are promising in their utilization in GNSS. However, the application of ML models in the industry is still limited. More effort and incentives are needed to facilitate the utilization of ML models in the PNT context. Therefore, based on the findings of this review, we provide recommendations for researchers and guidelines for practitioners.

Manuscript received 30 December 2021; revised 5 August 2022 and 28 October 2022; accepted 31 October 2022. Date of publication 3 November 2022; date of current version 6 December 2022.

DOI. No. 10.1109/TAES.2022.3219366

Refereeing of this contribution was handled by M. Tavana.

An earlier version of this article was presented in part at the 2021 International Conference on Localization and GNSS (ICL-GNSS) [DOI: 10.1109/ICL-GNSS51451.2021.9452295].

Author's address: Akpojoto Siemuri is with the University of Vaasa, Wolffintie 32, FI-65200 Vaasa PL 700, 65101 Vaasa, Finland (e-mail:akpo.siemuri@uwasa.fi). Kannan Selvan, is with the University of Vaasa, Wolffintie 32, FI-65200 Vaasa PL 700, 65101 Vaasa, Finland (e-mail: kannan.selvan@uwasa.fi). Heidi Kuusniemi is with the School of Technology and Innovations, University of Vaasa, Wolffintie 32, FI-65200 Vaasa PL 700, 65101 Vaasa, Finland (e-mail: heidi.kuusniemi@uwasa.fi) and Department of Navigation and Positioning, Finnish Geospatial Research Institute (FGI), National Land Survey of Finland (NLS). Petri Valisuo is with the School of Technology and Innovations, University of Vaasa, Wolffintie 32, FI-65200 Vaasa PL 700, 65101 Vaasa, Finland (e-mail: petri.valisuo@uwasa.fi). Mohammed S. Elmusrati is with the School of Technology and Innovations, University of Vaasa, Wolffintie 32, FI-65200 Vaasa PL 700, 65101 Vaasa, Finland (e-mail: moel@uwasa.fi). *(Corresponding author: Akpojoto Siemuri.)*

I. INTRODUCTION

THE growing complexity and dependency on global navigation satellite system (GNSS) technologies have increased the need for delivering high-performance GNSS solutions in terms of performance parameters, such as accuracy, availability, continuity, and integrity, at lower costs. Another additional performance indicator is the "Time To First Fix (TTFF)," which is used by some receiver manufacturers. GNSS performance prediction is a very important and essential activity to be carried out before the system is deployed so that the performance of the navigation system can be estimated, and the maintenance efforts and downtime can be significantly reduced. The early detection of faults and errors may lead to the timely correction of these faults [1].

There are various performance parameters addressed in the literature. These performance parameters as well as error/fault data can be used to develop and evaluate models used for the detection and correction of GNSS errors. These models can be used for estimating and predicting GNSS performance required by an application relying on a GNSS system in order for it to perform adequately.

There are some significant sources of errors for satellite-based positioning namely the ionospheric and troposphere effects, multipath, clock drift, receiver noise, interference, and hardware biases [2]. Ionospheric content interferes with GNSS signals, thereby, inducing errors for users when making calculations of their position from such signals. In addition, because GNSS was designed to operate in ideal line-of-sight (LOS) conditions, it has been observed to be highly influenced by the signals arriving at the receiver with multipath propagation in locations having a high possibility of reflection or refraction of the signal, such as urban areas. The GNSS signals are also vulnerable to radio frequency interference (RFI), due to their very low power at the Earth's surface. There are areas where GNSS signals are denied or not available (GNSS-denied environments) and this affects or even hinders the calculation of the user's position. All these can have a severe degrading effect on receiver performance [2].

The GNSS performance degradation is an area where machine learning (ML) finds its application as it deals with a nearly limitless quantity of data GNSS provides. ML has been used in many research works to propose and provide solutions to tackle GNSS challenges.

When compared to statistical methods, ML techniques enable us to identify tricky dependencies in data for which exploratory analysis has not allowed the proper determination of the shape of the underlying model [3]. The aim of using ML is not to generate an explicit formula for the distribution of the data. Rather, ML can be used to train an algorithm to learn the relation between the input features and the output. Furthermore, it is used for studying inter-relationships between features of a dataset. For example, if the features are continuous variables, you can use covariance to find their inter-relationship. Covariance is a measure of how much a feature is dependent on another. This learning method makes it possible to allow relaxation of the assumptions needed as seen in many statistical methodologies. Furthermore, the use of ML in GNSS context has seen increased interest by also several industries, such as Google's DeepMind AI, which learns to navigate cities without a map. When this is achieved in a real-world practical setting, it means artificial intelligence (AI) can recognize both objects and the type of the object, and relate them to the physical environment at various scales and distances [4]. Google has also found a new way to update its maps, by combining deep learning with Street View. This is done by combining the location data from Street View car's GNSS with address information and business names extracted from imagery. This could help in effectively mapping an entire city without any pre-existing knowledge of the layout or nomenclature [5]. In February 2020, Apple applied to the Federal Communications Commission for a license to install GPS testing equipment on its headquarters campus. It is thought that this move is related to the application filed by Apple Inc. with the U.S. Patent Office in August 2019, describing the company's "Machine Learning Assisted Satellite Based Positioning" [6]. Therefore, with this increased interest by industry and researchers alike in ML utilization in GNSS, we decided to make a systematic literature review (SLR) to analyze and compare the different ML algorithms, methods, and solutions used in the literature, as it can facilitate the development of new and efficient solutions in the utilization of ML techniques in GNSS. We perform a systematic review of studies between 2000 and 2021 to analyze and cite examples from the identified literature.

The rest of this article is organized as follows. Section II presents the methodology used in this systematic review. In Section III, we present and discuss the results of the review process. The implications for research and practice are presented in Section IV. While Section V describes the limitation of this article. Finally, Section VI concludes this article.

II. MATERIALS AND METHODS

In the planning, conducting, and reporting of the systematic review performed in this article, a process was adopted from [8], as illustrated in Fig. 1.

In the planning stage, the review protocol was developed, which includes the following steps:

1) identifying the research questions;

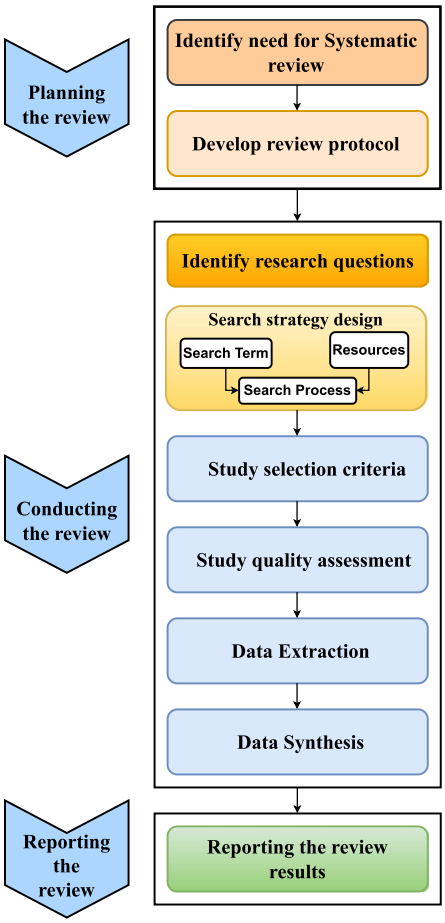


Fig. 1. Systematic review process adapted from [8].

2) design of the search strategy;
3) criteria for study selection;
4) quality assessment of the study;
5) data extraction;
6) data synthesis process.

Afterward, the results are then used in reporting the review.

The first step was the formation of the research questions that covered the issues to be addressed in the SLR. In the second step, the search strategy was described including the identification of search words and phrases and the selection of data sources from where the search will be performed in order to identify the candidate studies. The third step is used to determine the relevant studies. In this step, the criteria for the inclusion and exclusion of studies in the relevant study are performed. The fourth step scans the reference of the relevant papers for additional relevant studies, and then the quality assessment criteria are applied to the total relevant papers to derive the final selected papers used for the SLR. The fifth step involves the design of data extraction forms to collect the required information from the final selected papers in order to answer the research questions and in step six, we devise methods for data synthesis. The review protocol was developed through frequent meetings and consultations with senior researchers and professors.

TABLE I
Research Questions

#RQs	Research questions	Motivation
RQ1	Which ML algorithms have been applied to GNSS?	Identify the ML algorithms that have been used in GNSS. Practitioners and researchers can use the identified ML techniques as candidate solutions in their work.
RQ2	To which GNSS use cases have ML algorithms commonly been applied?	Identify GNSS use cases to which ML algorithms have been utilized. Practitioners can use this information to decide if their work requires the consideration of using an ML technique.
RQ3	What are the GNSS data sets used in the ML algorithm?	Identify GNSS data sets used in ML algorithms.
RQ4a	Do ML algorithms outperform non-ML algorithms?	In some of the existing studies, the proposed ML algorithms are compared with traditional non-ML GNSS algorithms in terms of estimation accuracy. RQ4a, therefore, aims to verify if ML algorithms perform better than non-ML algorithms. Estimation accuracy is the primary performance metric for ML models. The four aspects of estimation accuracy are accuracy metric, accuracy value, data set used for model construction, and model validation method.
RQ4b	Are there ML algorithms that significantly outperforms other ML algorithms?	Investigate which ML algorithms consistently outperform other ML algorithms. Therefore, RQ4b aims to identify the ML algorithms with relatively good performance.
RQ5	What are the strength and weaknesses of ML models applied to GNSS?	This aims at identifying the pros and cons of utilizing ML algorithms in GNSS. Having a full understanding of the characteristics of the candidate ML models, practitioners can make rational decisions on choosing the ML models that favor the GNSS use case in focus.
RQ6	What are the methods used for evaluation or validation of the ML algorithms?	List out the methods used for evaluation or validation of the ML algorithms to determine their accuracy and performance.

This review protocol helps in reducing the possibility and risk of research bias in the SLR. The following sections describe the research questions used and the steps taken during the period the SLR was conducted.

A. Research Questions (RQs)

This SLR is performed with the aim of providing and assessing results obtained from the studies done on the utilization of ML techniques in GNSS. This article extensively reviews studies between 2000 and 2021. From the final selected studies, first, we identify the different ML algorithms applied to GNSS (RQ1). Second, we identify GNSS use cases in which ML techniques are commonly used (RQ2). In the third research question (RQ3), the GNSS datasets used in ML techniques are identified. In RQ4a, the performance of ML techniques with traditional GNSS parameters/techniques is compared. This was done with the aim of determining if ML techniques are better than the traditional GNSS parameters/techniques. In RQ4b, the assessment of whether an ML technique outperformed other ML techniques in order to determine if an ML technique is consistently better than other ML techniques was addressed.

In research question RQ5, the strengths and weaknesses of different ML techniques are discussed to provide GNSS experts and researchers guidance regarding the selection of an appropriate ML technique based on the context of the GNSS application. Finally, in RQ6, the different methods used for the evaluation or validation of the ML algorithms are discussed. Furthermore, future guidelines are provided to GNSS technology experts and researchers regarding the application of ML techniques in GNSS. Table I presents more details on the research questions addressed in this SLR.

In Table XI in Appendix B, a summary of the ML algorithms utilized in GNSS and the GNSS use case they were applied to has been presented. Other details include year of publication, data type, the ML validation method, and the paper type (journal/conference).

B. Search Strategy

The search strategy comprises search terms, literature resources, and search process, which are detailed one by one as follows:

1) *Search Terms:* We formed sophisticated search terms by incorporating alternative terms and synonyms using the Boolean expression "OR" and combining main search terms using "AND." The following general search terms were used for the identification of literature:

GNSS AND "deep learning" OR GNSS AND "machine learning" OR GNSS AND "artificial intelligence" OR GNSS AND "random forest" OR GNSS AND "decision tree" OR GNSS AND "support vector machine" OR GNSS AND "neural network" OR GNSS AND "regression"

2) *Literature Resources:* After identifying the search terms, relevant digital portals were selected. The selection was restricted by the availability of digital portals at the home universities. The following electronic databases were used for the search. We also used relevant studies from The Institute of Navigation (ION).[1]

1) IEEE Xplore
2) Google Scholar
3) ScienceDirect
4) Crossref
5) Scopus
6) Institute of Navigation (ION)

We restricted the search from 2000 to 2021 to capture the ML and GNSS-related studies for the most recent two decades. The initial search to identify the literature for the review was performed after which the candidate studies were determined from the full-text papers by removing duplicate and irrelevant papers. The search was limited to only publications in journals and conferences. However, we included one paper from 1995 because we found it to be very relevant to our review process.

3) *The Search Process:* To facilitate the use of ML techniques in GNSS, it is necessary to systematically review the performance of these ML techniques and their usage from existing literature and studies. To the best of the authors' knowledge, there is no systematic review that focuses on ML techniques utilized in GNSS use cases except for our conference publication presented at the 2021 International Conference on Localization and GNSS (ICL-GNSS) [7].

To achieve this aim, we extensively searched through some relevant digital libraries to identify studies to answer the research questions. The final selected studies were selected based on the quality assessment of the studies and their relevance. Fig. 2 illustrates the stages involve in this SLR.

1) *Search phase 1:* Search each electronic database separately and then go through each paper title and abstract to check its relevance based on the search

[1] [Online]. Available: https://www.ion.org

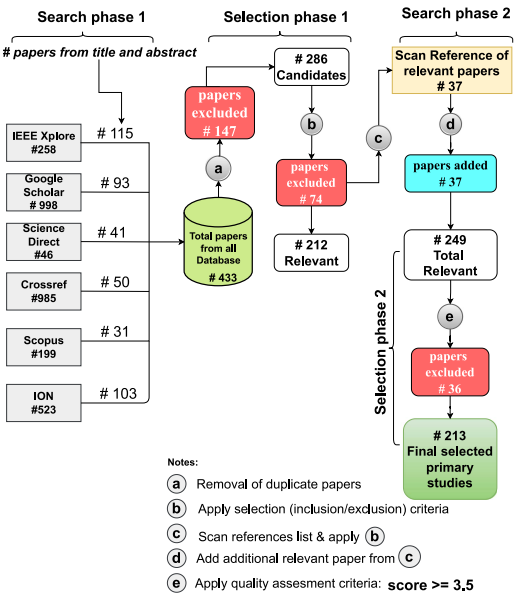


Fig. 2. Stages in the systematic review process.

term (*i.e., # papers from title and abstract*). Afterward, gather the returned papers for each electronic database together and remove duplicate papers to form a set of candidate papers.

2) *Search phase 2:* Scan the reference lists of the relevant papers to find extra relevant papers and then, if any, add them to the set of relevant papers.

C. Study Selection

Search phase 1 resulted in 433 retrieved papers. Since not all of the retrieved papers would provide the information useful to address the research questions raised by this review, further filtering is needed to identify the relevant papers. This is the aim of the study selection. Specifically, as illustrated in Fig. 2, the study selection process consists of two phases. It is important to note that in each selection phase, two researchers conducted the selection independently. If there were any disagreements between them on the selection criteria, a group meeting involving all researchers was held to make a decision.

1) *Selection phase 1:* Apply the inclusion and exclusion criteria (defined below) to the candidate papers so as to identify the relevant papers, which provide potential data for answering the research questions.
2) *Selection phase 2:* Apply the quality assessment criteria (defined in the next section) to the relevant papers so as to select the papers with acceptable quality, which are eventually used for data extraction.

The final literature to study was selected after following the criteria for inclusion and exclusion listed below, which had been refined through pilot selection. We carried out the study selection by reading the titles, abstracts, or full text of the papers.

Inclusion Criteria:

1) Studies utilizing ML techniques in GNSS use cases.
2) Studies combining ML and non-ML techniques in GNSS application.
3) Studies utilizing ML techniques for GNSS/non-GNSS integration such as inertial navigation system (INS), ultra-wideband (UWB), and wireless fidelity (Wi-Fi).
4) Comparative studies that compare different ML models or compare the ML model with the non-ML model in GNSS use cases.
5) Only journal versions of papers will be included: for studies with both conference versions and journal versions.
6) Only the most complete and latest paper will be included for duplicate publications of the same study.

Exclusion Criteria:

1) Studies based only on non-ML techniques applied to GNSS.
2) Studies using ML techniques in a context other than in GNSS.
3) Studies based on ML techniques used with only non-GNSS techniques, such as INS, UWB, and WIFI.
4) Similar studies, that is, studies by the same author in conference as well extended versions in the journal. However, if the results were different in both studies, they were retained.
5) Review studies (mini-reviews), editorials, news.
6) Short communications, encyclopedia, book chapters, case reports, conference info.

Using the above steps, we identified 212 relevant studies for inclusion in the SLR process. We then used search phase 2 to include more papers from the reference lists of the relevant studies, which produced an extra 37 studies. Therefore, a total number of 249 relevant studies were identified for further processing and analysis.

D. Quality Assessment Criteria

We formed a quality questionnaire for assessing the relevance and strength of the relevant studies. The quality criteria were developed by considering the suggestions given in [9]. Table II presents the quality assessment questions. The questions are ranked 1 (yes), 0.5 (partly), and 0 (no). The final score is obtained after adding the values assigned to each question. A study could have a maximum score of 6 and a minimum score of 0.

Two independent researchers ranked the quality questions for each relevant study and consulted other researchers in case of any disagreement. Finally, after thorough reviews, discussions, and brainstorming sessions, a final decision about the inclusion/exclusion for each study was made. To ensure the reliability of the findings of this review, we considered only the relevant studies with acceptable quality, i.e., with quality scores equal to or greater than 3.5, for the

TABLE II
Quality Assessment Questions

#Q	Quality factors	Yes	Partly	No
Q1	Are the ML algorithm used in the research clearly described?			
Q2	Are the GNSS use cases to which the ML algorithms are utilized clearly defined?			
Q3	Are the data set for model construction clearly defined and are the evaluation or validation methods for the ML algorithms mentioned?			
Q4a	Are there any comparison done between non-ML vs ML?			
Q4b	Are there any comparison done between ML vs other ML?			
Q5	Are the strengths and weaknesses of the ML algorithms specified?			

TABLE III
Data Extraction Form

ID
Authors
Paper Title
Year of Publication
DOI
Machine Learning Technique Used (RQ1)
Study Application (Uses case) (RQ2)
Data Category (Real or Simulated)
Data Used (RQ3)
Performance (ML vs. non-ML) (RQ4a)
Performance (ML vs. ML) (RQ4b)
Strength and Weakness (RQ5)
Validation Method (RQ6)
Results
Paper Abstract
Digital Library
Paper Type (Type)
Publication Venue

subsequent data extraction and synthesis. Accordingly, we further dropped 36 papers with a quality score of less than 3.5 in selection phase 2 (see Fig. 2). Finally, after the application of the quality assessment criteria stated in selection phase 2 to the total number of relevant studies, we identified 213 papers as the final selected studies used for this SLR. The quality scores of all 213 selected studies are presented in Table VII in Appendix A. These 213 studies were then used in the data extraction form. The more detailed list of these 213 selected papers can be found in the data extraction form in Table XI in Appendix B.

E. Data Extraction and Data Synthesis

A form is filled out for each of the final selected studies. The purpose of using the data extraction form is to determine which research question was satisfied by a final selected study. We summarized author's name, title, publishing details, dataset details, independent variables (metrics), and the ML techniques. The details of which specific research questions were answered by each final selected study were present in the data extraction card. These data extraction cards were used to collect information from the final selected studies. Two independent researchers collected the information required for each final selected study from the data extraction card. The two researchers then matched their results and if there is any disagreement between the two, other researchers are consulted to resolve these disagreements. The resultant data are saved into a excel file for further use during the data synthesis process. Table III shows the data extraction form used to collect the data from the selected studies.

The basic objective while synthesizing data is to accumulate and combine facts and figures from the final selected studies in order to formulate a response and resolve the research questions [10]. Collection of a number of studies that state similar and comparable viewpoints and results help in providing the research evidence for obtaining conclusive answers to the research questions. We scrutinized and evaluated both the quantitative data, which include values of various performance metrics like area under the receiver operating characteristic (ROC) curve (AUC), prediction accuracy, and qualitative data, which include strengths and weaknesses of the ML methods, categorization of various ML methods, feature subselection methods, and datasets used. We utilize a number of techniques to synthesize data collected from our final selected studies. In order to answer the research questions, we used visualization techniques, such as line graphs, box plots, pie charts, and bar charts. We also used tables for summarizing and presenting the results.

F. Threats to Validity

The three main threats to the validity of the process implemented in this review are presented from the following standpoints: bias from study selection, possible inaccuracy in the data extraction process, and publication bias. Therefore, based on the research questions and the aim of this review, the search terms were generated. It was noticed that some studies used different terms in titles that may not be related to the research questions or aim of our review process. As a result, it is possible that there were biases in the search strategy. The study selection process was performed by two independent researchers. However, some relevant studies that were found may have been excluded during the selection phase, but it was a minimal number of records.

The inclusion and exclusion criteria were used to select the studies to meet the aim of the review. These criteria were agreed on by all the authors to meet the scope of the study. Another possible threat to the validity of this review is publication bias. Based on this, it is more likely that positive results on ML models will be published than negative results, or researchers may put forward their methods as outperforming other ML or non-ML methods. This can therefore lead to an overestimation of the performance of ML models. However, this may be limited by the inclusion of studies that did not implement new ML models, but just did comparisons between ML models and other models.

TABLE IV
Publication Type Distribution

Paper Type	Number of Studies	Percentage
Journal	123	57.75
Conference	87	40.85
PhD thesis	1	0.47
Msc thesis	1	0.47
Book-Chapter	1	0.47
Total	**213**	**100**

TABLE V
Quality Levels of Relevant Studies

Quality level	**# Of studies**	**Percent**
Very high (5 <= score <= 6)	46	18.47
High (3.5 <= score <= 4.5)	167	67.07
Medium (2.5<= score <= 3)	14	5.62
Low (1.5<= score <= 2)	19	7.63
Very low(0 <= score <= 1)	3	1.20
Total	**249**	**100**

To reduce the inaccurate data extraction bias, a specialized card was utilized for data extraction (see Table III). In addition, the study selection process was undertaken in its entirety by two independent researchers (extractor and checker) with other researchers resolving disagreements by discussion among all researchers.

III. RESULTS AND DISCUSSIONS

In this section, the findings of this review are presented and discussed. We begin by presenting an overview of the selected studies. We then report and discuss, one by one in separate sections, the findings of the review based on the research questions. We interpret the review results not only within the context of the research questions but also in a broader context that is closely related to the research questions. Furthermore, some related works are also presented to support the findings.

A. Overview of the Final Selected Studies

In this review, we identified 213 primary studies that applied ML in GNSS use cases (see Table VIII). These papers were published between 2000 and 2021.

A total of 122 (57.55%) papers were published in journals, 87 (41.04%) papers were published in conference proceedings, and 3 (1.42%) papers (one Ph.D. thesis, one M.Sc. thesis, and one book chapter (see Table IV). The publication venues of the selected studies are presented in Table VI while the distribution of the studies over publication year is shown in Fig. 3.

From Fig. 3, it is interesting to see how the number of publications has been expanding over the years. The types of the selected studies belong to experiment research except for one survey research [7], and one case study research was found [11]. Although most of the selected studies used one form of validation dataset to validate ML models, it does not follow that the validation results sufficiently reflect the real situations in the industry. In fact, the lack of sufficient case studies and surveys from the industry may imply that the application of ML techniques in GNSS is still immature.

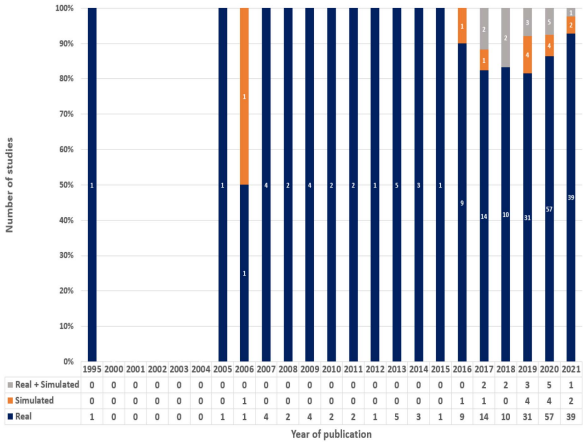


Fig. 3. Distribution of the studies over publication year.

Regarding the quality of the selected studies, we used the quality score of the study which must be equal to or above 3.5 (with a perfect score for the quality assessment being 6), before it is included in the review. As shown in Table VII, about 85.14% (213 of 249) of the selected studies are in high- or very high-quality level.

B. Types of ML Techniques (RQ1)

From the selected studies, we were able to identify some of the main types of ML techniques that had been applied to GNSS use cases. They are listed as follows:

1) Decision tree (DT)
2) Random forest (RF)
3) Regression analysis (linear/logistic)
4) K-nearest neighbor (KNN)
5) K-means clustering
6) Naive Bayes (NB)
7) Extreme learning machine (ELM)
8) Gaussian process regression (GPR)
9) Support vector machine (SVM)
10) Neural networks (NNs) a.k.a. artificial neural network (ANN)

a) Recurrent neural network (RNN)
b) Long short-term memory (LSTM), a special kind of RNN
c) Convolutional neural network (CNN)
d) Multilayer perceptron (MLP)
e) Back propagation neural network (BPNN)
f) Deep neural network (DNN)
g) General regression neural network (GRNN)
h) Radial basis function neural network (RBF-NN)
i) Bidirectional recurrent neural networks (BRNN)
j) Deep belief network (DBN)
k) Time-delay neural network (TDNN).

Among the above-listed ML techniques, NNs, DTs, and SVMs were the three most frequently used techniques; they together were adopted by about 84% of the selected studies, as illustrated in Fig. 4. The top three algorithms

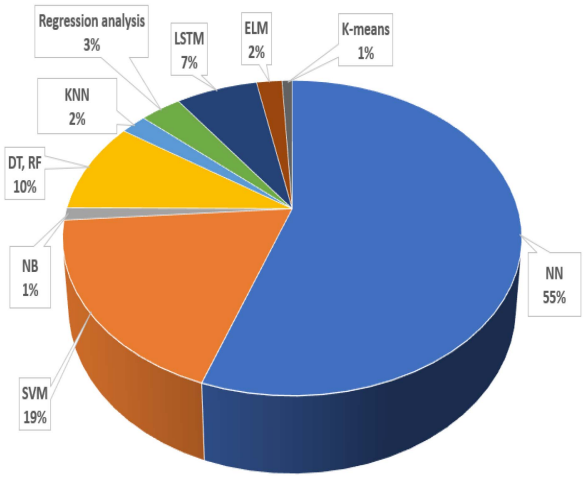


Fig. 4. Summary of ML algorithms mostly utilized in GNSS.

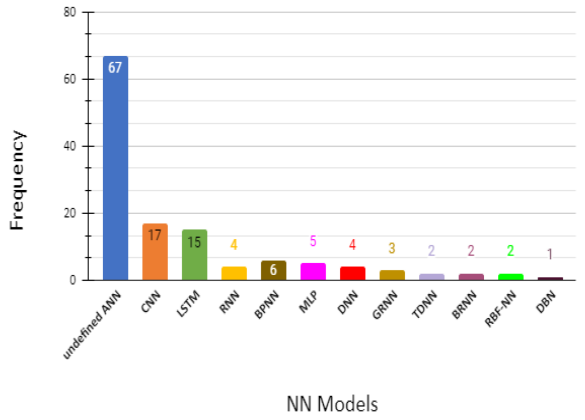


Fig. 5. Distribution of the NN model's terminologies as used in the selected studies.

have the following percentages: NNs (55%); SVMs (19%); DTs (10%). Considering only the selected ones, this shows that DL models have been utilized the most in GNSS out of all ML models. The distribution of the NN models as shown in Fig. 5 is based on the terminologies used in the reviewed literature. This shows that ANN (58%), CNN (15%), and LSTM (18%) have been used more frequently by researchers in various GNSS applications. Note that "undefined ANN" is used to categorize the ANN models where the author of the selected paper was not specific on the category of the ANN used. Fig. 5 is used to represent the number of times these NN model terminologies are used in the selected papers. We present the terminologies used in the selected papers. From this, we can see that, for example, MLP, BPNN, and DNN are all related but have been counted separately based on their reference in the selected papers used for the review.

MLP is known as the foundation architecture of DNNs. However, notice that MLP and DNN are separated in Fig. 5. This is done in this article in other to keep track of how the NN models' terminologies are being used. For example, although DNNs could be seen as a subset of MLP, they are different in their structure. The structure of both MLP and DNN consists of an input layer, hidden layers, an output layer, an activation function, and a set of weights and biases. However, DNN has a higher number of hidden layers stacked together for processing and learning from data, while MLP has few numbers of hidden layers. Similarly, LSTM is a special kind of RNN that is suitable for classification, processing, and prediction using time series data. This is because time series can have lags of unknown duration between important events. LSTMs were designed to take care of the vanishing gradient problem that can be experienced when training traditional RNNs. It is difficult to train RNNs that require long-term memorization, but LSTM performs better with these kinds of datasets as it has more additional special units that can hold information longer.

C. GNSS Use Cases for ML Algorithm (RQ2)

ML utilization in GNSS is becoming more popular among researchers. The ML algorithms were used for classification, clustering, forecasting, and anomaly detection depending on the GNSS use case. In this section, we present some of the GNSS use cases in which ML algorithms have been utilized based on the selected studies used in this review.

1) *GNSS Signal Acquisition:* Signal acquisition is the process of assessing the presence of GNSS signals and providing a rough estimate of the parameters of the incoming signal: the Doppler frequency and the code delay. It is the first step performed by a GNSS receiver. The outcome of this process decides if a particular satellite signal is present or not in the received signal, and it also gives a rough estimate of its associated code delay and Doppler frequency if present [12]. This acquisition process is implemented by all GNSS receivers [2]. This process is achieved through the evaluation of the cross ambiguity function (CAF), usually in a discrete-time domain. CAF is a 2-D function that is related to the correlation between the received signal and local code for every possible delay/Doppler pair. The CAF can be assumed to be an image and has certain traits that can be utilized in identifying the presence or absence of the signal from a specific satellite. With this knowledge, a data-driven model can be trained such as 1) an MLP, which is an NN architecture with moderate complexity been widely applied in ML literature; and 2) a Convolution Neural Network (CNN), having the ability to recognize complex nonlinear phenomena, at the expense of a much larger complexity compared to MLPs [12].

2) *Signal Detection and Classification:* The detection and classification of GNSS signals are very useful as it helps to differentiate the various types of signals based on how they are being affected or not by the environment or media of propagation. GNSS signals can be classified into, for example, LOS, non-line-of-sight (NLOS), and multipath [13]. A multipath signal is a GNSS signal bouncing off a reflective surface prior to reaching the GNSS receiver antenna. This means multipath interference occurs when an RF signal from a transmitter arrives at a receiver through two

or more routes. Multipath is one of the major sources of a GNSS error that lead to unacceptable pseudorange errors and affects positioning. Multipath causes distortions of GNSS signals; therefore, it needs to be detected, excluded, or corrected. NLOS signals are reflected signals arriving at a receiver even when the LOS signal is blocked. Detecting the characteristics of the acquired signal enables the system to decide how to treat the signal based on the evaluated effect it would have on the GNSS positioning solution. New designs of receivers can help mitigate against multipath to some extent but this is not the case for all receivers, for example, low-cost receivers and GNSS-enabled smartphones. Most of the market-ready solutions currently available for multipath detection and mitigation are based on stochastic modeling, spatial geometry modeling, advanced techniques in data processing, and special hardware designs [3]. These models need to be able to accurately and reliably classify LOS, multipath, and NLOS signals. However, when there is an outside event that is too complex to be modeled and that does not fit the mathematical assumptions used to develop the statistic model, these solutions become ineffective. ML methods enable us to relax assumptions attached to the statistical methodology. In [14] (SVM), [15] (LSTM), [16] (DT, SVM, and KNN), [17] (SVM, NN), [18] (CNN), [19] (SVM), and [13] [gradient-boosting decision tree (GBDT), DT, KNN, and adaptive network-based fuzzy inference system (ANFIS)], ML have been applied to signal detection and classification. An ANN model capable of processing the structure of the autocorrelation function (ACF) was used for the detection of evil waveforms (EWFs) in [20] (ANN). EWFs are a rare perturbations occurring at the stage of signal generation. Detecting this type of distortion postcorrelation traditionally involves hand-crafted structure tests on a densely sampled ACF. These are designed for specific scenarios; therefore, they lack flexibility compared to data-driven methods. In [13], compared with DT, KNN, and ANFIS, a robust GBDT was employed for GNSS signal reception classification. It made use of the carrier-to-noise-density ratio (C/N0), pseudorange residuals, and satellite elevation angle as the input features to improve the performance of the signal classification at the receiver. Similarly, in [16], compared with SVM, and KNN, a DTs-based classifier has used the satellite elevation and C/N0-R-L ratio as the input features to improve the performance of the signal classification at the receiver. While in [21], GNSS interference signal recognition based on CNN and fusion time–frequency features was implemented. The accurate detection and classification of GNSS signals can help in the improvement of the GNSS positioning accuracy especially in urban areas where the GNSS signals are heavily impacted by the environment [19].

3) *Earth Observation and Monitoring:* Earthquakes detection—ML algorithms have been used in Earth observation and monitoring applications, such as in [22], where an alternative ML approach to the prediction and detection of earthquakes and the determination of their magnitude has been proposed. Results show that the ANN process achieved an accuracy of 85.71% in validation assessment to predict the earthquake approximately 3 h before the seismic event. Other techniques applied to the monitoring of earthquakes include seismographic stations, Interferometric Synthetic Aperture, strong-motion measurements, and gravity measurement [23]. Hurricane tracking—ML has also been applied to hurricane tracking using a convolutional neural network (CNN) in [24]. The trained CNN regression model has achieved accuracy with less than 1.5 pixels errors in x and y coordinates, i.e., 0.2% error on average.

Sea ice detection/sensing/thickness estimation—ML-aided sea ice monitoring methods make use of spaceborne global navigation satellite system-reflectometry (GNSS-R) data. These data collect information about sea ice concentration (SIC), sea ice thickness (SIT), etc. In [25], sea ice detection/sensing/thickness estimation is done using an NN. On average, using this method, the accuracy for sea ice detection is about 98.4%. It was found that when GNSS-R delay-Doppler maps (DDMs) data are adequately preprocessed, CNNs and NNs share similar accuracy; otherwise, the former outperforms the latter. Furthermore, it was concluded that CNNs were more tolerant to the data format changes than ANNs [26]. In [27], support vector regression (SVR) and CNN are used. Comparisons showed good consistency between the derived and reference SIT, with correlation coefficients of 0.95 and 0.90 and root mean square differences of 5.49 and 7.97 cm for SVR and CNN, respectively. While in [28], among NN, CNN, and NN-FS, the NN-FS (FS means feature selection) showed the best performance, and it was the closest to that of SVM-FS. Through experiment, it was found that SVM-FS produced fewer false alarms compared to NN-FS during the analysis of false detection of ice under different sea conditions in terms of wind speed. In [29], DT- and RF-based methods have been used and achieved an overall accuracy of 97.51% and 98.03%, respectively, in the Arctic region and 95.46% and 95.96%, respectively, in the Antarctic region. Furthermore, a regression NN is used in [30] to train thin ice and full-range models having a mean absolute error (MAE) of 6.5 and 23 cm, respectively.

In the estimation of snow depth (SD), a DBN was used in [31]. The results showed that the DBN SD retrieval model estimates SD more accurately than linear methods and CNN models. Specifically, R increased from 0.81 to 0.85, MAE decreased from 11.15 to 9.55 cm, and root mean square error (RMSE) decreased from 17.96 to 15.40 cm.

Soil moisture (SM), wind speed retrieval, and vegetation water content (VWC)—SM retrieval is a vital activity in various applications such as hydrology and agriculture. GNSS-R is a new type of remote sensing technology also used for SM retrieval. In [32], random forest (RF) was used for GNSS-R SM retrieval. While in [33] RF, SVM, gradient boosting DT (XGBoost), and ANN was used. The XGBoost model performed best with an RMSE of 0.052 cm. The proposed algorithm can be applied to other training and testing problems that could benefit from it such as hydrology and agriculture where accurate SM estimates play an important role. Another study that used the XGboost ML-aided method in GNSS-R SM retrieval/estimation is

seen in [34]. The results showed a good correlation with the statistical analysis of ground-truth measurements.

Other ML-based methods include Bayesian regularization neural network (BRNN) used in [35]. RF and SVM used in [36], SVM used in [37], ANN in [38], and ANN, RF, and SVM in [39].

For GNSS-R wind speed retrievals and estimation, NN was used in [40], [41], [42], and [43]. When compared to an LS-based approach, the derived model shows a significant improvement of 20% in the RMSE. While for wind speed retrievals from cyclone global navigation satellite system (CYGNSS), ANN was used in [44]. The comparison highlights that the ANN approach outperforms the baseline approach for both low and high wind speeds (ANN RMSD improves by 15%) and removes most of the geographical biases between baseline winds and wave-watch 3 model winds seen in monthly maps of wind speeds. The ANN improvement increases for increasing wind speed [44]. Similar accuracy was found in [45] and [46], where ANN outperforms the traditional approach for wind speed retrieval. In [47], RF was used in down-scaling GNSS-R-based VWC, which is recognized as an important parameter in vegetation growth study. The results showed that the RF model outperformed the traditional methods of multiple linear regression (MLR) and kriging interpolation in the cross-validation results (R of MLR is only about 0.4, and that of the cross-validation of kriging interpolation is only 0.3). Using the RF method, the results decreased a lot with R decreasing by 0.2 and RMSE increasing by 0.015 for cross-validation results. Similarly, in [48], RF-method, BPNN, and GRNN were used in monitoring the variation of VWC. Among the three ML methods, the results of RF were the best, followed by those of GRNN and BPNN [48].

Other areas of Earth monitoring—An RF algorithm has also been used in the prediction of dam displacement [49]. While in [50], SVR was used in monitoring the urban heat island effect, which has been widely studied because of its impacts on the environment and human well-being.

Other areas of Earth monitoring in which ML has been applied include using GNSS position time series to predict the land subsidence or upheave in an area. This is done by predicting the next GNSS position time series using algorithms like MLP, Bayesian NN (BNN), RBF, KNN, GRNN, SVR, GP, and classification and regression trees (CART) [51], [52], [53]. Another Earth monitoring application is the nowcasting of severe weather events and summer storms from the combination of vertically integrated water vapor with vertical profiles of wet refractivity derived from GNSS tomography. In [54], RF was used.

ML models have been applied to other environmental remote sensing applications such as landslide monitoring/prediction, estimating nearshore water depths, weather forecast by monitoring and forecasting precipitable water vapor (PWV), and forecast hourly intense rainfall [11], [55], [56], [57], [58], [59], [60], [61], [62].

4) *GNSS Navigation and Precise Positioning:* Location-based services can be utilized in many aspects, namely, tracking, health care monitoring, and intelligent transport systems (ITS). ML has also been applied in this area with the aim of improving GNSS navigation/positioning in several scenarios. GNSS outage for very short periods may not represent a relatively big problem for a position estimate, as an INS can be integrated to produce position estimates. However, a long outage period means the bias error from motion sensors will start to increase and position estimate accuracy is significantly reduced as well [63]. Therefore, long GNSS signal outages could harm vehicles' position estimates and become a risk for ITS and its users. Nowadays, GNSS is successfully implemented to achieve precise positioning in both indoor and outdoor environments [64], [65]. Such precise positioning is essential for safe operations [66], [67]. ML has been applied in ITS to estimate the GNSS position error by aiding motion sensor units in providing a more accurate position estimate during periods of outage or blockage of the GNSS signal [63]. For land vehicle navigation applications, an ANN model and an efficient hybrid methodology based on Dempster–Shafer theory augmented by an SVM (DS-SVM) were implemented in order to effectively fuse GNSS and INS data [68]. Kalman filtering (KF) is used for linear systems and extended KF (EKF), i.e., linearized KF can be implemented but may cause filter divergence under high dynamic conditions. In [68], test results indicate that the proposed DS-SVM algorithm effectively compensated and reduced positional inaccuracies over the regular ANN model and traditional KF/EKF methods during GNSS availability and outage conditions for low-cost inertial sensors.

In [69], LSTM is used to achieve a GNSS network-based real-time kinematic improvement. The GNSS sensor only provided the RMSE of about 3.8 m compared to the LSTM model, which significantly improved to about 0.45 m. GNSS position error estimation is done in order to improve the positioning accuracy after error correction. In [63], DT and SVM were implemented, with the average RMSE for SVM being around 31% less than the one seen in the best results in RMSE for DTs (28 versus 41 cm). Features considered in training the models included elevation, azimuth, constellation type, and carrier-to-noise ratio.

In [70], fully connected NNs (FCNNs) and LSTM are combined and used to predict the GNSS satellite visibility and pseudorange error in an urban area based on the available GNSS measurements. These combined networks achieve satisfactory performance on both satellite visibility and pseudorange error predictions, which have 80.1% overall accuracy and a 4.9-m average difference from the labeled pseudorange error (reference). A least-squares SVM (LSSVM) technique is used in [71] for GNSS navigation with dynamic model real-time correction. The results show that the proposed LSSVM-KF algorithm can adequately adapt to time-variant dynamics and performs well for real-time correction. In [72], a new ML-based architecture [LR, linear discriminant analysis (LDA), SVM, KNN, CART, and Gaussian Naive Bayes (NB)] combines classical observables for local hazard detection, with the outcome of advanced Receiver Autonomous Integrity Monitoring

(RAIM) in order to find out if a given point of a railway is suitable for safe and reliable use of GNSS for train positioning. The results show that the setup of the classifier can be driven by the features of the electromagnetic environment on which the train shall operate. Using the daily records taken by the GNSS receivers, the map of the local hazards and the effects of their combinations can be learned. This can mitigate the risks derived by significant changes in the environment (for example, constructing new buildings along a railway) or by the activation of new RF sources (such as new 5G RAN nodes). ML models have been applied to other positioning and navigation applications, such as regional mapping of the geoid [73] and human mobility analysis on large-scale mobility data, which has contributed to multiple applications, such as urban and transportation planning, disaster preparation and response, tourism, and public health [74]. Other applications include location prediction using GPS trackers to, for example, locate missing people with dementia [75], improving GNSS Positioning from smartphones [76], [77], [78], [79], improving GPS code phase positioning accuracy in urban environments [80], improving accuracy of differential GPS correction prediction in the position domain [81], and improving kinematic GNSS positioning accuracy with low-cost GNSS receiver in urban environments [19]. Another study used the combination of Genetic Algorithms (GA) and NNs for exploring the navigation satellite constellation design tradespace to speed up the constellation performance computation and for an improved GNSS integrity [82]. In [83], a wavelet neural network (WNN) is employed for orbit approximation to obtain a continuous orbit function. This is because the orbit function is essential in positioning and navigation tasks. Therefore, the advantage of continuity is that it can also be used during GNSS signal interruptions.

5) *GNSS-Denied Environments and Indoor Navigation:* The foundation for a context-adaptive system is scenario recognition and it is a major contributor to seamless indoor and outdoor positioning technology. This allows the system to "understand" its environment, and then apply the appropriate strategies to achieve accurate, continuous, and reliable positioning in the different scenarios [84]. Scenario recognition is essential for seamless indoor localization and robust positioning in complex environments. RNN-based scenario recognition with multiconstellation GNSS measurements on a smartphone was implemented in [85]. Here, a complex environment was divided into four categories (deep indoor, shallow indoor, semioutdoor, and open outdoor) and the influence of multiconstellation satellite signals on scenario recognition performance based on a hidden Markov model algorithm was analyzed in detail prior to the application of RNN for the scenario recognition. The experimental results show high recognition accuracy in both isolated scenarios and transition environments, with an overall accuracy of 98.65%.

In the case of situation and context awareness, Guinness [86] implemented an ML-based approach [SVM, ANN, LR, BN, DT, NB, instance-bases learning with parameter k (IBk)], and locally weighted learning (LWL) for sensing mobility contexts using smartphone sensors. The aim is to develop techniques that continuously and automatically detect a smartphone user's mobility activities, including walking, running, driving, and using a bus or train, in real time or near-real time (<5 s). The main aim of this study was to investigate if an ML algorithm exists that can produce a classifier having both high performance (with respect to class prediction rate) and low computational complexity. The results show that several existing ML algorithms achieved performance above 95% accuracy; however, DT algorithms were the only ones that also had relatively low computational complexity (for classification) [86].

In indoor navigation, NLOS and multipath are caused by indoor furniture and flat surfaces of the walls and ceilings, and it is quite severe. Since detecting NLOS and multipath is a classification problem, deep learning can be used to tackle this problem. In [87], a deep learning approach [NLOS and multipath detecting network (NMDN)] is used for indoor NLOS and multipath detection. The datasets used are generated by a GNSS software receiver using an intermediate frequency signal collected from an indoor pseudolite system. This method is compared with two SVMs, which are the traditional methods for classification, and shows an improvement of up to 45% in overall classification accuracy. In [88], NN fingerprinting and GNSS data fusion are done to improve localization in an indoor environment. The NN-based positioning fusion model was able to reduce the positioning error by up to 49%, having submeter accuracy in the uncertainty-free scenarios and 1.75 m mean positioning error in the 5-dB uncertainty case.

6) *GNSS Anomaly Detection and Atmospheric Effects:* GNSS has been used recently in the understanding of the atmospheric effects on the Earth because of the analysis of the ionospheric behavior. This ionospheric behavior can be derived through the determination of the total electron content (TEC) derived from GNSS data processing. TEC is a representation of the electron density in the signal trajectory between the satellite and the receiver on the Earth's surface. A good understanding of the tropospheric wet delay and ionospheric scintillation effects on GNSS signals has been of great interest both in the fields of science and industry. Ionospheric scintillation is the rapid fluctuation of the amplitude and phase of radio frequency signals (for example, GNSS), propagating through the ionosphere. In GNSS receivers, signal acquisition and tracking can be heavily impacted by strong scintillation, resulting in degradation in accuracy and continuity of GNSS performance. It is difficult to predict and model scintillation because there are different reasons for the occurrence of this phenomenon, for example, solar activity, magnetic storms, local electric fields, conductivity, and wave interaction to name a few [89].

Ionosphere analyses: ML models have been employed for the detection of scintillation, as illustrated in [89] using SVM-based algorithm. It used cross-validation to determine the optimal hyperparameters with a detection accuracy is 96%, which is increased by 1.5% compared to the previous implementation. In [90], the overall accuracy in the

validation performance is around 92%, which demonstrates the good performance of the SVM detector on phase scintillation detection.

In [91], Ridge Regression, Long short-term memory (LSTM), Classification Neural Network, Autoencoder Classification Neural Network, and LSTM Autoencoder Classification Neural Network were implemented, and compared. The results showed that the durability of the LSTM Autoencoder Classification Neural Network model over the span of a year can predict irregularities up to 3 hours in advance with an accuracy of 92%. Other studies include [92] and [93] using ANN model, [94], [95] and, [96] using DT model, [97], [98], [99], and [100] using SVM model, [101] and [102] using RBF SVM model (RBF-SVM), and [103] using an NN model.

The time delay of the GPS (L1 and L2) signal in the ionosphere is one of the propagation path delays and it depends on the TEC of the atmospheric layer. This delay contributes to a potential source of error in time measurements and can produce an error range in tens of meters. ML algorithms have been used to predict ionospheric time delays from GNSS observations like in [104] and [105] using an NN model. It used an extrapolation methodology combining two types of input data, observed TEC, and environmental parameters. The NN method yielded the best accuracy when compared to the least square regression (LSR) model and bi-harmonic spline (BHS), which is a pure spatial extrapolation method.

Others include [106] and [107] using GPR, and [108] using SVM. Prediction of tropospheric wet delay from GNSS observations has also been performed using ML algorithms like in [109], [110], [111], [112], and [113] using ANN, and [114] using BP NN.

Tropospheric analyses: In [115], ANN has been used to precisely identify interfered radio occultation (RO) events from GNSS RO measurements widely used in the prediction of weather, climate, and space weather, particularly in the area of tropospheric analyses. The estimation of tropospheric wet delay is of great importance for real-time weather forecasting applications. The characteristics and effects of ANN in tropospheric analyses can be validated by comparing the predicted zenith wet delay values using ANN with the values estimated from GNSS observations and meteorological data. ANN is useful in finding hidden relationships between features in a dataset, such as the relationship between the GNSS signal delay and PWV as in the case of tropospheric analyses. It has the ability to learn, recall, and generalize from the given data by suitable assignment and adjustment of weights. Due to significantly reduced computation time compared to the traditional SVM method, ANN is useful for real-time RO applications, such as extreme weather prediction or data assimilation [115].

7) *GNSS Security: Spoofing and Jammer Attacks:* A user device receiving false signals and believing it to be authentic could prompt dangerous behavior due to the false position or timing fixes. An example was mentioned in [116], where GPS spoofing was used to misdirect a hovering drone into an unplanned dive and to steer a yacht off course. Therefore, defenses against spoofing are aimed at detecting an attack in order to warn the attacked receiver that its navigation fix and clock offset are unreliable. The second reason for defense against spoofing is to recover a reliable navigation and timing solution. This has been achieved at the pseudorange level with receivers employing RAIM by using an inconsistent set of five or more pseudoranges to allow the receiver to detect an unsophisticated spoofer that broadcasts one or more false signals with no attempt to achieve a believable consistency. However, in response to increased efforts to defend against spoofing, advanced forms of GNSS spoofing have been conceived and interest in GNSS spoofing has increased with an example of such spoofing called "in the wild," which is an actual malicious spoofing attack. Various defense strategies that have been developed to deal with self-consistent spoofing have been beaten by these advanced spoofings [116]. Therefore, we have seen researchers attempting to use ML to detect and defend against spoofing attacks, for example, in [117] using the ANN model. Using features such as pseudorange, PR, Doppler shift, etc., an increase in the ANN model detection performance is observed. For one feature, the highest accuracy of 73.2% is achieved with the pseudorange. With two features (pseudorange and Doppler shift), the best performance with an accuracy of 92.6%, a probability of detection of 85.2%, and a probability of false alarm of 0% is obtained. As the number of features increases, the accuracy also increases. Another application is the C/N0 abnormity detection method for GPS antispoofing using an ANN model [20].

In [118], using MLP trained with particle swarm optimization (PSO), the simulation results showed that spoofing attack detection improved approximately 4 and 2% in comparison with the results achieved through classification based on Bayes-optimal rule and multihypothesis Bayesian classifier mentioned in the literature review. The feature vector used by this method includes received signal power and correlation function distortion. It uses these features to try to classify received signals as jammed, spoofed, multipath, or interference-free signals.

While in [119], using RNN based on LSTM, the error of the forecasting model over the testing dataset is in the order of 0.0006%. The model is able to detect GPS anomalies using signal imperfections, Doppler shift deviations higher than 4 Hz, and detecting mobile spoofing devices at higher speeds of 2.5 km/h.

In [120] SVM classification (C-SVM) with principal component analysis (PCA) was implemented. The overall success rate of the proposed approach was 97.8%, and the cross-validation error was slightly higher. Using PCA, the relations among the selected variables were analyzed. These variables include lock time, pseudorange, carrier Doppler frequency, receiver clock bias, receiver clock drift, C/N, code variance, multipath correction, carrier multipath correction, full carrier phase [cycles], carrier variance, and spoofing indication.

Research has also been done on the use of ML to detect and classify interferences and jammer signals in order to

defend against jammer attacks. Types of jamming signals include audio jamming, narrow-band jamming, pulse jamming, sweep jamming, spread spectrum jamming, and the combination of each two of the above jamming signals. Jammer signals are a type of interference signal; therefore, the detection and classification of GNSS interference signals can help identify if the interference is an intentional and malicious (jammer or spoofing signal) attack or if it is unintentional, which can still cause GNSS positioning errors.

For intentional jammer attacks, studies have been conducted using ML to detect and classify GNSS interference such as in [121] using a twin SVM algorithm (TWSVM) for real-time interference monitoring. Implementing SVM in the interference monitoring of GNSS signals meets the requirement of objectivity and accuracy. However, the training speed of standard SVM does not satisfy the requirement of being in real time for interference monitoring. The experimental results indicate that the TWSVM model [121] is faster than the standard SVM in training speed (the training speed of TWSVM is at the millisecond level and the classification speed of TWSVM is at the microsecond level) and can be used in practice.

While for unintentional attacks like solar radio bursts that interfere with the GNSS signal, the SVM algorithm was used for the detection of the interference. In [122], SVM and CNN were used for jammer signal classification in GNSS bands. The results showed that with a small library of images and not excessively complex parameters/network layer architectures, a high mean classification accuracy was obtained for SVM (94.90%) and CNN (91.36%). It used a dataset[2] composed of 61 800 different images and containing different jammer types including the no-interference scenario and also used randomly generated parameters for these interference types.

While in [18], CNN is used for jammer signal classification. Here, the proposed CNN method has both robustness and good accuracy in jamming signals classification. This is seen in the results for single jamming signal classification, the proposed CNN method correctly classifies almost 100% of jamming signals. While for coexisting situations (more than one jamming signal), the lowest classification accuracy is up to 92%. Other studies include [123] using LSTM with CNN, [124] using LR, KNN, NB, DT, and SVM algorithms, [125] using a multilayer NN, and [126] using SVM.

8) *GNSS/INS Integration:* KF is widely used as a data-fusion algorithm in navigation. The integration of GNSS and non-GNSS systems, such as INS, makes use of KF for its GNSS/INS calibration systems. When GNSS cannot supply measurement updates normally, the filter time would be increased. In such cases, the divergence of INS error is fast without GNSS information correction. This could be detrimental depending on the use case, such as the concealment of unmanned underwater vehicles and unmanned aerial vehicles (UAVs). ML algorithms have been used in several studies to mitigate such scenarios. In general, when GNSS is active, the ML model is used to learn the divergence characteristics of the INS error under several basic conditions depending on the area of application (vehicles, UAV, etc.). If there is a disturbed GNSS signal, the ML model is used to correct the position error of the INS in order to improve the navigation accuracy.

In [127], the BPNN algorithm was used for GNSS/INS integration to overcome the GNSS outages. The simulation experiments used BPNN to compensate for the KF algorithm. The results showed an improved accuracy of the integrated navigation when GNSS is unavailable. The proposed method is able to maintain low-level deviations for about 9 min. Within this short navigating mission without a GNSS signal, the reliability and feasibility of the UAV can be verified. Similarly, in [128], the BPNN algorithm was used for GNSS/INS integration. The results show that the BPNN model can efficiently predict the increment of position and compensate for the accumulation of INS errors during GNSS outages.

In [129], a CNN-based adaptive KF is implemented to achieve the GNSS/INS integration. The estimator can output the system noise covariance matrix by windowed inertial measurements. The experimental results show that the proposed algorithm has a better performance (three times lower RMSE value) compared to classical KF and Sage–Husa adaptive filter in highly dynamic conditions. While a CNN-LSTM method is used in [130], here the results of the CNN-LSTM model compared with the EKF navigation method, show significant improvement in the navigation accuracy and the alignment time. It has a final attitude accuracy better than 0.2°, an alignment time of 10 s, and a position accuracy better than 3 m. This can meet the requirements of low-cost vehicle flexibility. While in [131], an ensemble learning algorithm (ELM) is used for INS/GPS navigation. To validate the performance of the proposed method, the results are compared with an adaptive network-based fuzzy inference system (ANFIS) and an EKF method. The result of ELM outperforms ANFIS and EKF by approximately 50% and 70%, respectively. This suggests a promising prospect for the use of ELM in the field of positioning in the absence of GPS signals using low-cost MEMS-based inertial sensors.

NN models have also been implemented to mitigate the GNSS signal outage in GNSS/INS integration in [132]. The study uses an NN model to do online learning of the system behavior during the time intervals when there are no satellite outages. It then takes advantage of this learning by applying it during periods of outages. The fixed velocity (stationary) results showed that the MAE after a 30-s outage was about 400 m without the NN model, but after the NN model was used for error compensation, it was approximately 100 m. For a constant-velocity trajectory, the position accuracy was about 500 m without NN error corrections and close to 100 m with the NN error corrections applied.

An SVM-based GNSS/INS integrated was utilized in [133] for land vehicle navigation. Here, the proposed

[2][Online]. Available: https://doi.org/10.5281/zenodo.3370934

SVM-based GNSS/INS integrated provided a 45%–73% improvement on rms positioning compared with the KF approach. It also outperformed BPNN by 8%–32%, and ELM by 46%–67% on rms positioning. The positioning improvements in maximum position accuracy were 26%, 66%, and 78% compared with KF, BPNN, and ELM methods, respectively.

GNSS can also be integrated into other devices such as cameras [134] using the CNN algorithm. The RMSE errors obtained remain low even at high added bias values of 100 and 200 m. The highest RMSE values were observed at low/medium biases because it is more difficult to detect and exclude low biases than high biases. Other ML-based GNSS/INS integration studies include [135] using RNN, [136] using RBF-NN, and [137], [132], [138], [139], and [140] using NN models.

9) *Satellite Selection:* Location accuracy is a result of two main factors namely, the satellite location-dependent geometric dilution of precision (GDOP) and the pseudorange measurement inaccuracies [141]. Generally, the more visible satellites available the better the positioning performance. The benefit of multiconstellation GNSS is that more visible satellites can be used to improve user positioning performance. However, not all satellite would contribute to the positioning performance because of some GNSS error like satellite clock error or high C/NR, etc. [142]. Therefore, selecting the optimal sets of satellites from all possible visible satellite combinations is important. This selection is done with the aim of minimizing either GDOP or weighted GDOP (WGDOP) as a criterion. Other selection criteria used in researches include elevation angle, C/NR, and range errors. ML has been applied in this kind of study to implement an ML-based satellite selection algorithm as seen in [143] using PointNet and VoxelNet networks, and [144] using an NN model. The study by Simon et al. [144] was one of the earliest research (from our database search in Section II) that utilized DL algorithm for satellite selection. Usually, a satellite subset is chosen by minimizing a quantity known as geometric dilution of precision (GDOP). However, in [143], the satellite selection algorithm is a satellite segmentation problem, having a specified input channel for each satellite and two class labels, one for selected satellites and the other for those not selected. The satellite segmentation algorithm is used to ensure that a fixed number of satellites with the minimum GDOP or WGDOP value can be segmented from any feeding order of input satellites. Whereas, in [144], an NN model is used to predict the GDOP without the usual resource consuming computation of the trace of the inverse of the measurement matrix. The NN model does this by learning the functional relationships between the entries of a measurement matrix and the eigenvalues of its inverse.

10) *LEO Satellites: Orbit Determination and Positioning:* The application of GNSS to the precise orbit determination (POD) of low-Earth-orbit (LEO) satellites has been beneficial in the development of many new space applications in the area of navigation, telecommunication, remote sensing, and Earth observation systems. These applications

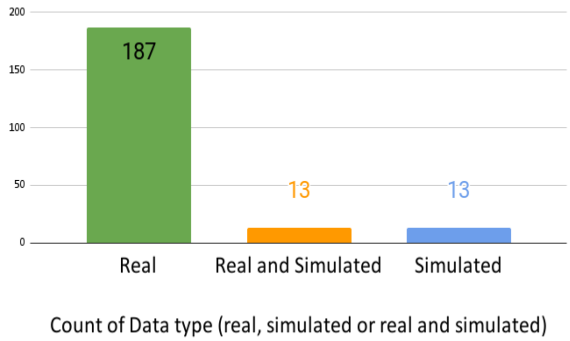


Fig. 6. Datasets for ML utilization in GNSS.

can benefit from the precise tracking of satellites orbits using onboard GNSS receiver data. With recent advancements, GNSS receivers have been designed to meet the POD requirements and have been implemented on many satellites, depending on the objectives of their missions, requiring accurate knowledge of their orbits. The performance of the POD process can be affected by the measurement environment, the technique used, and the mission application of the satellite. Furthermore, besides accuracy, the need to reduce the latency in achieving a precise solution has been of interest. This is beneficial to many end-user applications as it provides faster access to the required orbit solutions [145]. ML for orbit determination of LEO satellites has been implemented by some studies [146], [147], [148], [149]. In [146] and [149], TDNN, which is a type of feed-forward NN (FFNN), and LSTM are used for simultaneous tracking and navigation with LEO satellites. An NN is implemented in [147] and [148] to mitigate GNSS multipath for LEO positioning applications. It was noticed that very limited studies have been done on the use of ML for POD of LEO satellites. This may be due to the fact that most studies related to GNSS navigation are based on improving positioning accuracy and mitigation of GNSS errors. Additionally, LEO satellites with GNSS receivers have only recently emerged, and thus not many studies have yet focused on their POD via GNSS accounting for little number of research on LEO satellites using ML models and GNSS.

D. Datasets Used by the ML Models (RQ3)

A variety of datasets have been used in ML utilization in GNSS studies. Fig. 6 shows the percentage of studies using different datasets. Some of the data used were simulated data [150], [151] while others were real data [23], [152], [153]. The utilized datasets can be publicly available (for free) or private in nature not shared by researchers; therefore, results on such datasets cannot be verified and such studies are not replicable. The major category of datasets used and their sources are as follows.

1) *Simulated data:* These datasets include, CU SeNSe Lab [154], and a customized software-defined radio (SDR)-based GNSS data grabber and software receiver [155].

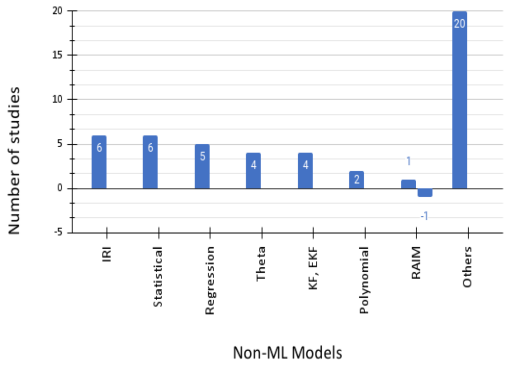


Fig. 7. ML versus non-ML (bars above zero line indicate that ML models are more accurate, while the bars below zero line indicate that non-ML models are more accurate).

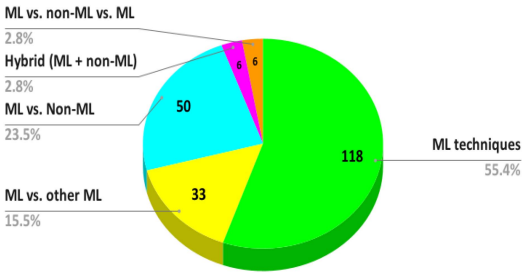


Fig. 8. Utilization of ML in GNSS.

2) *Real data:* These datasets include GNSS raw data from smartphones (it also include data collected by individuals.) [156], vertical total electron content (VTEC) data from the National Oceanic and Atmosphere Administration [23], and GPS data over International GNSS Service (IGS) stations [152], [153].
3) *Combining real and simulated data:* The combination of real and simulated data are used [3], [120], [157].

Different kinds of devices were used for the collection of the data used for research. These include low-cost receivers such as u-blox, smartphones, and high-end receivers, for example, Trimble, Novatel, and Javad Triumph VS receivers. The type of research determined the location for data collection. These locations include open sky environments (LOS), urban canyon (multipath/NLOS prone), indoors (GNSS-denied environment), and laboratory-generated data.

In GNSS, even with the same equipment and the same data collection path, the data collected at different times should be regarded as different dataset due to the change in satellite's geometry. Therefore, in the GNSS field, it is rare to see researchers use the same dataset for different objectives or even more, the same objective. However, in research, replicability is important; therefore, it would be a good practice to create a database where researches can store their research data for easy access in order for other researchers to be able to replicate their results. This way, the ML model used can be trained and evaluated by other researchers making use of the shared data.

E. ML Versus Non-ML Models (RQ4a)

In some of the selected studies, ML models versus non-ML models performance were compared (as seen in Fig. 7). The ML models have been compared with several conventional non-ML models: regression model [14], [80], [101], [158], [159], brute force approach [143], traditional statistical approaches [60], [94], [160], [161], [162], [163], classical KF [129], Bayes-optimal rule [118], least square (LS)-based approach [40], Saastamoinen model [110], autoregressive model and a traditional LEO propagation model (EKF-STAN) [146], conventional wind speed retrieval method [43], maximum-likelihood power-distortion (PD-ML) [164], BERNESE 5.2 [114], CYGNSS [44], hydrostatic-seasonal-time model [49], statistical theta method [51], [52], [53], [165], MAPGEO2004 geoid model [73], GNSS-IR SM [58], autoregressive and autoregressive moving average [166], ERA-Interim—a global atmospheric reanalysis (now ERA5 reanalysis) [107], empirical linear algorithms (LRM and LLM) [59], International Reference Ionosphere (IRI) 2016 model [167], NeQuick and IRI-2001 global TEC model [168], [169], [170], EKF-based integration scheme [171], CODE Global Ionospheric Maps (GIMs) [172], autoregressive integrated moving average (ARIMA), and quadratic polynomial models [173], LSR and BHS [105], linear interpolation method and inverse distance weighted interpolation method [112], KF [138], [139], polynomial model [93], [174], IRI-2001 model [175], conventional systems (RAIM) [126], [176], EGNOS [103], and IRI-2012 model [177].

In Fig. 7, we present the studies that compared the performance of ML with non-ML models. Majority of the studies concluded that ML models outperformed non-ML models except for one study (GNSS/INS integration [126]), where the SVM model has a similar performance to conventional systems (RAIM), but suffers from faster degradation due to the tightly coupled fusion algorithm. Specifically, Fig. 8 shows that 23.47% (50 of 213) of studies did a comparison between non-ML and ML models [14], [31], [40], [43], [44], [45], [51], [52], [53], [58], [59], [60], [73], [80], [93], [94], [103], [105], [107], [110], [112], [114], [126], [129], [130], [131], [138], [139], [143], [146], [158], [159], [160], [161], [162], [163], [164], [165], [166], [168], [169], [170], [171], [172], [173], [174], [175], [176], [177], [178].

The percentage of studies that made comparison between ML and other ML models was 15.49% (33 of 213) [12], [13], [16], [17], [26], [27], [28], [29], [33], [36], [39], [56], [63], [68], [77], [86], [87], [91], [108], [115], [122], [123], [124], [135], [179], [180], [181], [182], [183], [184], [185], [186], [187]. Whereas 55.39% (118 of 213) of the studies did not do any comparison but presented the result of ML model(s) used [3], [11], [15], [18], [19], [20], [21], [22], [24], [25], [30], [32], [34], [35], [37], [38], [41], [42], [46], [47], [48], [50], [54], [55], [57], [61], [62], [69], [70], [72], [74], [75], [76], [78], [79], [81], [83], [85], [88], [89], [90], [92], [95], [96], [97], [98], [99], [100], [102],

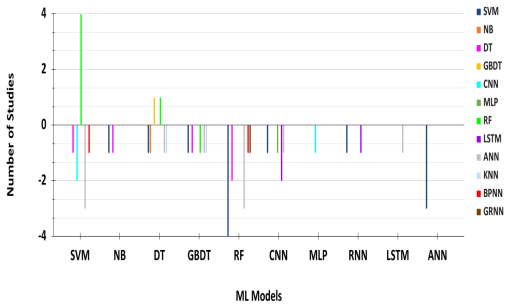


Fig. 9. Comparison between ML models (bars above zero line indicate that models in vertical axis are more accurate, whereas the bars below zero line indicate that models in horizontal axis are more accurate).

[104], [106], [109], [111], [113], [117], [119], [120], [121], [125], [127], [128], [132], [133], [134], [136], [137], [140], [144], [147], [148], [157], [188], [189], [190], [191], [192], [193], [194], [195], [196], [197], [198], [199], [200], [201], [202], [203], [204], [205], [206], [207], [208], [209], [210], [211], [212], [213], [214], [215], [216], [217], [218], [219], [220], [221], [222], [223], [224], [225], [226], [227], [228], [229], [230], [231], [232], [233].

Furthermore, 2.82% (6 of 213) of studies compared one or more non-ML model with one or more ML model [49], [71], [101], [167], [234], [235], while 2.82% (6 of 213) of studies implemented a hybrid between ML and non-ML algorithm [82], [118], [149], [236], [237], [238]. These hybrid implementation studies claim their performance is better than ML only and non-ML only implementations. The comparison of the utilization of ML in GNSS are presented in Fig. 8.

F. ML Versus Other ML Models (RQ4b)

For the comparisons between different ML models, we compared the accuracy of many ML models applied to the same use case. This comparison is done for studies that implemented more than one ML model for a particular GNSS use case. All ML models applied to a specific GNSS use case are analyzed for comparison based on accuracy. In general, five significant comparison results can be found from the selected studies. First, RF, GBDT, CNN, RNN, and ANN are more accurate than LSTM and SVM respectively. Some studies showed that DT, GBDT, and RF-based classifier obtained a higher accuracy than KNN, CNN, ANN, and SVM respectively. Another ML model with significant performance was the SVM in its various forms (SVM, LSSVM-KF, and C-SVM) performing better than CNN, ANN, and DT. These performance comparisons between ML models are provided in Fig. 9.

The reasons for some ML models such as DT, GBDT, and RF performing better than NN models (ANN, CNN, and LSTM) may be due to the following:

1) Very few studies conducted experiments to compare ML with other ML models for the same GNSS use case.
2) All the GNSS use cases demonstrating the superiority of NN models over other ML models come from the studies of NN models; therefore, it is possible that some of them have a bias toward NN model and these studies may be overly optimistic.
3) The use cases used for this comparison differ slightly from each other and as a result, the type of ML algorithm used varies and it may not be beneficial to do a cross-comparison of ML model across GNSS use cases.

Furthermore, for ML models that are rarely compared with each other, it is difficult to determine which is more accurate.

G. Strength and Weakness of ML Models Applied to GNSS (RQ5)

Given the various GNSS use cases, we may have to concern ourselves with selecting the appropriate ML models for a specific GNSS use case. By investigating the characteristics of the candidate ML models (to be more precise, the ML techniques), such concern can be addressed. The aim is to identify the strengths and weaknesses of the ML techniques. To this aim, we have extracted the strengths and weaknesses of ML techniques and synthesized them based on the type of the ML technique. The different studies assessed the ML techniques in different ways and from different aspects based on the GNSS use case the ML model was applied to, therefore, we decided to synthesize in general, the common strengths and weaknesses that were mentioned by at least two studies (see Table IX). The list comprises of the most common algorithm from the selected studies. However, in Table X, we present the strength and weaknesses as mentioned by some of the authors of the selected studies.

The topics about model selection, model application, and model combination, which are closely related to the characteristics of ML techniques and estimation contexts have been discussed in some existing works. With respect to model selection, a tree-form framework to select appropriate ML models was proposed [239]. The design of the framework is based on criteria, such as dataset size, uncertainty, causality, and applicability. These preliminary criteria can be extended to include more criteria related to ML models. With respect to applying ML models to GNSS use cases, no study has proposed a framework or procedure; however, Zhang and Tsai [240] proposed a general procedure for applying ML models that consists of the following steps: problem formulation, problem representation, data collection, domain theory preparation, performing the learning process, analyzing and evaluating learned knowledge, and dealing with the knowledge base. This procedure is also applicable to GNSS-related tasks. In the aspect of model combination, MacDonell and Shepperd [241] stated that the combination of a set of diverse techniques can improve estimation accuracy in cases where no dominant technique can be found. In [242], an SLR was conducted where it was also found that combining models would usually produce better estimates than when the models are used individually. Other studies [243] have

provided strong support for this possibility although in the field of software development effort. Studies have presented the opinion that combining two or more ML techniques may potentially enhance the power of the estimation model. This can hold true in the field of GNSS, as shown in [77].

Although ML techniques have been proved in some studies to be effective for some GNSS use cases (contexts), they do not always perform well on all GNSS use cases. This implies that an absolutely "best" ML model (technique) does not appear to exist, and the performance of a particular ML model depends heavily on the contexts it applies to. Therefore, in order to choose appropriate ML techniques and apply them to real-world GNSS use case efficiently, the characteristics of the candidate ML techniques as well as the contexts of the GNSS use case needs to be well understood. In [244], it was noted that selecting the best estimation method "in a particular context" is more beneficial than selecting the "best" estimation method in general.

H. Evaluating/Validating ML Models (RQ6)

Since ML model is data-driven, both building the model and its validation rely extremely on the training data. Therefore, when evaluating the estimation accuracy of an ML model, the training dataset on which the model is built and validated, must be taken into account, as well as the employed validation method. Various historical datasets were used to build and validate the ML models identified in this review. The most frequently used datasets together with their relevant information has been presented in the previous sections.

Regarding the type of data used for evaluating the model, the studies made use of either simulated data, real data, or semisimulated data to evaluate the model depending on the GNSS use case. The real data were collected using high-grade GNSS receivers to be used as reference/ground truth. In some studies where GNSS ground truth/reference data are not available (for example, indoor applications), predefined known trajectories were used. During the trajectory recording, stops are made at certain way-points and the current location is recorded as a reference point (ground truth). For the studies that used simulated data, the simulation was done to be used as the accurate reference data for the model evaluation. The aim of testing with simulated data is to validate the proposed method in a well-controlled environment, where all the parameters (for example, multipath/direct-only signals) can be manually defined and clearly labeled.

With respect to validation methods, the metric explains the performance of an ML model. The chosen metrics influence how the performance of ML algorithms is measured and compared. They influence how we weigh the importance of different characteristics in the results. Furthermore, the ML model may give satisfying results when evaluated using a metric like *accuracy score* but may have poor results when evaluated against other metrics such as "*logarithmic loss*" or any other such metric. Hence, it is very much important to choose the right metric to evaluate the ML model. The metric used depends on if it is a classification, regression, or clustering problem. Some examples of classification, regression, and clustering metrics are listed as follows.

1) *Classification Metrics:*
   a) Accuracy.
   b) Logarithmic loss.
   c) ROC, AUC.
   d) Confusion matrix.
   e) Classification report.
2) *Regression Metrics:*
   a) MAE.
   b) Mean Squared Error.
   c) RMSE.
   d) Root mean squared logarithmic error.
   e) R square.
   f) Adjusted R square.
3) *Clustering Metrics:*
   a) Silhouette score.
   b) Rand index.
   c) Adjusted rand index.
   d) Mutual information.
   e) Calinski–Harabasz index.
   f) Davies–Bouldin index.

From the selected studies, Holdout, n-fold cross-validation ($n > 1$), and comparison with another algorithm were mostly used. Specifically, the numbers (percentages) of the studies that used these three validation methods are 2 (3.84%) for Holdout, 20 (38.46%) when compared with another algorithm, and 30 (57.69%) for n-fold cross-validation. Furthermore, accuracy metric should also be considered in evaluating the ML models. Different metrics measure the accuracy from different aspects and effort estimation accuracy can be measured using various metrics. It was found from the selected studies that RMSE, mean square error (MSE), ROC curves, and Standard-Deviation (StD) were the most popular accuracy metrics. Specifically, the numbers (percentages) of the studies that used these three metrics are 45 (56.25%) for RMSE, 7 (8.75%) for MSE, 9 (11.25%) for ROC, and 12 (15%) for StD.

More details can be seen in Table XI in Appendix B. It was also seen that when evaluating an ML model, we can make use of other well established model or methods for the evaluation.

IV. IMPLICATIONS FOR RESEARCH AND PRACTICE

This review has found that the studies on the application of ML techniques such as FFNN, RBF-NN, DBN, and SVR to GNSS use cases are still limited. Some ML techniques have not even been applied in the GNSS domain. Researchers are therefore encouraged to explore the possibilities of using the unapplied ML techniques to new GNSS use case context. In order to identify these unapplied ML techniques and to use them more efficiently in GNSS,

researchers should keep track of the related disciplines like ML, artificial intelligence, data mining, and statistics. Because these disciplines can provide valuable insights and methods to address GNSS challenges.

High-quality historical GNSS dataset with detailed descriptions of features and data collection process is essential for building and validating ML models. This review has shown that, on one hand, most of the available GNSS datasets are GNSS observations. On the other hand, some of the studies used simulated GNSS datasets. These data varied and their means of collected also varied from study to study. To address this and thereby promote ML utilization in GNSS research, we suggest that researchers share their GNSS datasets in the research community after the removal of confidential information.

In the comparison of different ML models for GNSS use cases, the limited number of relevant studies and the nonuniform experimental designs may account for inconclusive and/or unclear results. Therefore, aside from performing more research and experiments, it will benefit the research community if a uniform experimental framework for evaluating the performance of different ML models utilized for a particular GNSS use case is developed. Without the use of such a uniform framework, the comparison results for different ML models may vary when using different datasets, or different experimental approaches even for the same GNSS use case.

In the case of the implications for practitioners, this review has found that very few of the selected studies focus on industry practice, for example, [76], [77], [78]. This may be evidence that the application of ML models in the real world of the GNSS industry is still quite limited. Therefore, we suggest that practitioners should cooperate with researchers to investigate the possibility of applying the promising ML models in their practices. For example, RF, SVM, CNN, and ANN have been investigated most extensively in academia; therefore, these can first be taken into consideration by practitioners, as a useful complement to the existing traditional GNSS model. However, because of the limited number of studies comparing ML models and traditional GNSS (non-ML) models found from this review, we recommend using both ML and non-ML models in parallel at the early stage of practice, which has been shown to have promising accuracy by some studies [82], [118], [149], [236], [237], [238]. The replacement of the non-ML (traditional GNSS) models with ML models can only be considered when the ML model performs significantly and consistently better than the existing traditional GNSS model.

This review has shown that different ML techniques are beneficial to different GNSS use cases. Therefore, prior to decision making concerning choosing an ML model to implement, practitioners need to know the contexts of the GNSS use case; in addition, they need to understand the characteristics of the ML models of interest. Usually, the degree to which the GNSS use case matches the characteristics of the chosen ML model can have a direct and significant impact on the performance of the ML model. This means that given the estimation contexts of a GNSS use case, we have to select ML models appropriate for the contexts. This concern can be addressed by investigating the candidate ML models or, more precisely, the ML techniques based on their characteristics, mainly reflected by the strengths and weaknesses of the ML techniques. The characteristics of the ML techniques are mainly associated with four types of estimation contexts namely: 1) small dataset, 2) outliers, 3) categorical features, and 4) missing values. For example, DT is prone to overfitting on small training dataset, while ANN and DT cannot deal with missing values. Second, ANN cannot deal with categorical features in their standard forms but will work as long as the categorical features have been quantified (or to avoid misleading information for methods using distance metrics, the categorical features should be encoded to suitable form, of which one-hot-encoding is often used). Third, similarly, ANN and DT cannot deal with missing values in their standard forms but will work as long as the missing values have been imputed.

In addition to the characteristics summarized above, there are some other distinct characteristics of ML techniques, which may be considered when choosing appropriate ML models. These are as follows: DT are intuitive and are easy to understand, while ANN has the ability to learn complex functions; however, it requires a large amount of data for training and may suffer from overfitting, with weak explanatory ability; BNN is capable of learning causal relationships. The topics about model selection, model application, and model combination have been discussed by existing works [239]. These topics are closely related to the characteristics of ML techniques and estimation contexts. With respect to model selection, a tree-form framework to select appropriate ML models was proposed in [239]. The framework is designed using criteria such as dataset size, uncertainty, causality, and applicability. These are preliminary criteria and can, therefore, be extended to include more criteria related to ML models. Furthermore, this framework was done for selecting the appropriate ML techniques for the prediction of software development costs. However, it can still be adapted to GNSS use cases as the characteristics of ML are the same irrespective of the field of application.

V. LIMITATIONS OF THIS REVIEW

This review considered accuracy metrics (e.g., RSME) and validation when evaluating the performance of ML models or comparing ML models with other models. Accuracy metrics are the most important metrics and were used by most of the studies. However, besides accuracy metrics, other performance metrics, such as generalization ability and interpretability, were ignored in this review. These may also be important, especially when selecting appropriate models for given GNSS use case. However, the summarized strengths and weaknesses of ML models, i.e., the outcomes of RQ5, are helpful to identify the appropriate ML models, which may alleviate this limitation to some extent. Another limitation in this review is that only 50 out of the 213 selected studies compare non-ML and ML

techniques. Thus, the non-ML versus ML comparison is not definitely conclusive. Furthermore, while comparing different ML techniques, each of the selected study made use of different experimental settings including datasets, feature reduction methods, and preprocessing methods. This review has revealed contradictions in some results of the comparisons between ML models and conventional non-ML models and between different ML models; therefore, it is difficult to establish which model is more accurate for a GNSS use case. To improve the chances of identifying the model which is more accurate for a GNSS use case, the comparison of the ML algorithms should be done on common grounds. Some of the comparison parameters may include: time complexity (how much time the algorithm takes), space complexity (how much memory an algorithm needed to run in terms of the input size), sample complexity (number of training examples needed to train the network in order to guarantee a valid generalization), bias-variance tradeoff, online and offline, parallelizability, parametricity, etc. [245]. The authors will consider these parameters in future work. Furthermore, it would be important to use the same data. Section III-D shows that there are plenty of different datasets used by different researchers. Therefore, we suggest the development of a common test bench to study the utilization of ML algorithm in GNSS. Another limitation is the insufficient number of studies reporting the desired comparisons, which may have led to these inconsistencies. In general, drawing conclusions from a large number of studies is more likely to be reliable. There is also a possibility that although we have exhaustively searched all the stated digital libraries, we may have missed a suitable study. In conducting this review, we have assumed that all the studies are impartial, and where this is not the case, it then poses a threat to this study.

Some of the mentioned strengths and weaknesses of the approach used in the studies were retrieved directly from the selected studies (see Table IX). This means that it is possible that some of them may just represent the authors' opinions and, therefore, may be unreliable. To increase the reliability of the synthesized results for RQ5, we take only the synthesized strengths and weaknesses supported by two or more selected studies. Therefore, care has to be taken in dealing with any inferences drawn from these synthesized results.

TABLE VI
Publication Venues of Selected Studies

Publication Venue	Paper Type	# Studies	Percentage
The Institute of Navigation	Conference	44	20.66
Remote Sensing	Journal	15	7.04
Sensors	Journal	12	5.63
Advances in Space Research	Journal	9	4.23
GPS Solutions	Journal	5	2.35
arXiv	Journal	5	2.35
International Geoscience and Remote Sensing Symposium	Conference	5	2.35
IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing	Journal	5	2.35
Advancing Earth and Space Science	Journal	5	2.35
IEEE Access	Journal	4	1.88
IEEE/ION Position, Location and Navigation Symposium (PLANS)	Conference	3	1.41
International Conference on Localization and GNSS (ICL-GNSS)	Conference	3	1.41
IEEE Transactions on Geoscience and Remote Sensing	Journal	3	1.41
The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences	Conference	3	1.41
Geodesy and Geodynamics	Journal	3	1.41
Journal of Atmospheric and Solar-Terrestrial Physics	Journal	3	1.41
International Conference on Intelligent Transportation Systems (ITSC)	Conference	2	0.94
Journal of Sensors	Journal	2	0.94
IEEE Transactions on Aerospace and Electronic Systems	Journal	2	0.94
IEEE Geoscience and Remote Sensing Letters	Journal	2	0.94
International Conf. on Sensing, Measurement Data Analytics in the era of Artificial Intelligence (ICSMD)	Conference	2	0.94
Remote Sensing of Environment	Journal	2	0.94
Materials Today: Proceedings	Journal	2	0.94
IEEE Internet of Things Journal	Journal	2	0.94
Applied Soft Computing	Journal	2	0.94
NAVIGATION, Journal of the Institute of Navigation	Journal	2	0.94
Information Fusion	Journal	2	0.94
Neurocomputing	Journal	1	0.47
ITS World Congress	Conference	1	0.47
IET Radar, Sonar Navigation	Journal	1	0.47
University of Nottingham	PhD thesis	1	0.47
International Conf. on Technology Management, Operations and Decisions (ICTMOD)	Conference	1	0.47
International Conf. on Signal Processing, Communications and Computing (ICSPCC)	Conference	1	0.47
Advanced Information Management, Communicates, Electronic and Automation Control Conf. (IMCEC)	Conference	1	0.47
IEEE Annual Consumer Communications Networking Conference (CCNC)	Conference	1	0.47
KTH ROYAL INSTITUTE OF TECHNOLOGY	Msc Thesis	1	0.47
RFI Workshop - Coexisting with Radio Frequency Interference (RFI)	Journal	1	0.47
Defence Technology	Journal	1	0.47
Asia-Pacific Conf. on Intelligent Robot Systems (ACIRS)	Conference	1	0.47
International Conf. on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT)	Conference	1	0.47
International Computer Conference, Computer Society of Iran (CSICC)	Conference	1	0.47
International Conf. on Computer Applications Information Security (ICCAIS)	Conference	1	0.47
Acta Astronautica	Journal	1	0.47
Engineering Science and Technology, an International Journal	Journal	1	0.47
Measurement	Journal	1	0.47
International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)	Conference	1	0.47
International Conf. on Acoustics, Speech and Signal Processing (ICASSP)	Conference	1	0.47
International Conf. on Signal, Information and Data Processing (ICSIDP)	Conference	1	0.47
IEEE Intelligent Vehicles Symposium (IV)	Conference	1	0.47
Global Oceans 2020: Singapore – U.S. Gulf Coast	Conference	1	0.47
European Navigation Conference (ENC)	Conference	1	0.47
IEEE Aerospace Conference (50100)	Conference	1	0.47
International Conf. on Artificial Intelligence in Information and Communication (ICAIIC)	Conference	1	0.47
IEEE Sensors Journal	Journal	1	0.47
Cognitive Communications for Aerospace Applications Workshop (CCAAW)	Conference	1	0.47
Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)	Conference	1	0.47
International Symposium onTelecommunications (IST)	Conference	1	0.47
Forum on Cooperative Positioning and Service (CPGP)	Conference	1	0.47
European Signal Processing Conference	Conference	1	0.47
IEEE International Conf. on Wireless for Space and Extreme Environments (WiSEE)	Conference	1	0.47
International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)	Conference	1	0.47
IEEE Sensors Journal	Journal	1	0.47
Animals	Journal	1	0.47
Satellite Navigation	Journal	1	0.47
Journal of Hydrology	Journal	1	0.47
Wireless Personal Communications	Journal	1	0.47
Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography	Journal	1	0.47
Leibniz International Proceedings in Informatics (LIPIcs)	Journal	1	0.47
International Journal of Remote Sensing	Journal	1	0.47
ISPRS International Journal of Geo-Information	Journal	1	0.47
IOP Conference Series: Materials Science and Engineering	Conference	1	0.47
Mechatronics	Journal	1	0.47
Journal of Geodesy	Journal	1	0.47
International Association of Geodesy Symposia book series	book-chapter	1	0.47
Applied Sciences	Journal	1	0.47
Advances in Artificial Neural Systems	Journal	1	0.47
International Journal of Electronics	Journal	1	0.47
International Journal of Computer Applications	Journal	1	0.47
Alexandria Engineering Journal	Journal	1	0.47
IEEE Aerospace and Electronic Systems Magazine	Journal	1	0.47
International Conference on Automation and Computing (ICAC)	Conference	1	0.47
Iranian Journal of Electrical Electronic Engineering	Journal	1	0.47
International Conference on Robotics and Biomimetics (ROBIO)	Conference	1	0.47
International Symposium on Signal Processing and Information Technology (ISSPIT)	Journal	1	0.47
Annals Geophysics (AG)	Journal	1	0.47
Studia Geophysica et Geodaetica	Journal	1	0.47
Others	Journal	5	2.35
Total		**213**	**100**

VI. CONCLUSION

This systematic review investigated ML utilization in GNSS. The type of ML technique, the performance accuracy of the ML model, the comparison between different models (including ML model versus non-ML model, and ML model versus other ML model), and the GNSS contexts (use case) in which the ML models were presented. An extensive literature search for relevant studies published in the period 2000–2021 have been performed and which identified 213 primary studies (referred to as "selected studies") that are pertaining to the six research questions (RQs) raised in this review.

The principal findings of this review are summarized as follows.

1) *RQ1*—The ML algorithms that have been applied in GNSS use cases are presented in Table XI in Appendix B. Among them, RF, SVM, ANN, and

TABLE VII
Quality Levels of Relevant Studies

ID	Authors	Q1	Q2	Q3	Q4a	Q4b	Q5	Score
S1	Simon et al.	1	1	0	0	1	1	4
S3	Machado et al.	1	1	1	0	1	0	4
S4	Bhatt et al.	1	1	0.5	0	1	0	3.5
S5	Socharoentum et al.	1	1	1	0	1	0	4
S6	Z. Zhou et al	1	1	1	1	1	1	6
S7	Jiao et al.	1	1	1	0	1	0	4
S9	Favenza et al.	1	1	0.5	0.5	0.5	0	3.5
S10	Y. Quan	1	1	1	0	1	0	4
S11	Yang et al.	1	1	0.5	0	1	0	3.5
S12	L.-T. Hsu	1	1	1	0	1	1	5
S14	Suzuki et al.	1	1	1	0	1	0.5	4.5
S15	Jiao et al.	1	1	1	0	1	0	4
S16	B. GUERMAH et al.	1	1	1	0	1	0	4
S18	Y. Liu et al.	1	1	1	0	1	0	4
S19	Y. Quan et al.	1	1	1	0	1	1	5
S20	P. Huang et al.	1	1	1	1	1	0	5
S21	Gogliettino et al.	1	1	1	0.5	0.5	0.5	4.5
S22	Zhidong Zhang et al	1	1	1	0	1	0	4
S23	Z. Liu et al.	1	1	1	0	1	0	4
S25	H.-U. Kim and T.-S. Bae	1	1	1	0	1	1	5
S26	N. Linty et al.	1	1	0.5	0	1	0.5	4
S27	M. R. Manesh et al.	1	1	1	0	1	0.5	4.5
S28	ALEJANDRO KURATOMI	1	1	1	0	1	0	4
S29	Q. Liu et al.	1	1	1	0	1	0	4
S30	R. Morales Ferre et al.	1	1	1	0	1	0.5	4.5
S31	A. LOUIS	1	1	0.5	0	1	0	3.5
S32	K. Lamb et al.	1	1	1	0	1	1	5
S33	S. Semanjski et al.	1	1	1	0	1	1	5
S34	R. Orus Perez	1	1	1	0	1	1	5
S35	R. Sun et al.	1	1	1	0	1	1	5
S36	D. Brum et al.	1	1	1	0	1	0	4
S37	H. Dai et al.	1	1	1	0	1	1	5
S38	Z. Zou et al.	1	1	1	0.5	1	1	5.5
S39	Chang et al.	1	1	0.5	0.5	0.5	0	3.5
S40	E. Munin et al.	1	1	1	0	1	0.5	4.5
S42	Li et al.	1	1	0.5	0.5	0.5	0.5	4
S43	Borhani-Darian et al.	1	1	1	0	1	0	4
S44	P. Borhani-Darian et al.	1	1	1	0	1	0	4
S45	S. Tohidi et al.	1	1	0.5	0.5	0.5	0	3.5
S46	Munin et al.	1	1	1	0	0.5	0	3.5
S47	M. Alshaye et al.	1	1	1	0	1	0	4
S48	Yunxiang Liu et al.	1	1	1	0	1	0	4
S49	L. Mallika I et al.	1	1	1	0	1	0	4
S50	Suzuki et al.	1	1	1	0	1	0	4
S51	M. O. Selbesoglu	1	1	1	0	1	0	4
S52	Y. Xia et al.	1	1	1	0	1	0	4
S53	Q. Yan et al.	1	1	1	0	1	0.5	4.5
S54	R. Calvo-Palomino et al.	1	1	1	0	1	0	4
S55	Haiyu et al.	1	1	1	0	0.5	0	3.5
S56	S. Semanjski et al.	1	1	1	0	1	0	4
S57	S. Semanjski et al.	1	1	1	0	1	0	4
S58	F. Dovis et al.	1	1	0.5	0	1	0	3.5
S59	Y. Jia et al.	1	1	1	0	1	0	4
S60	E. I. Adegoke et al.	1	1	1	0	1	0	4
S61	Y. Jia; et al.	1	1	1	0	1	0	4
S62	Q. Yan et al.	1	1	1	0	1	0	4
S63	M. Asgarimehr; et al.	1	1	1	1	1	0	5
S65	L. Miotti et al.	1	1	1	1	1	0	5
S67	Y. Liu et al.	1	1	1	0	1	0	4
S68	T. Mortlock et al.	1	1	1	1	1	0	5
S69	L. Mengying et al.	1	1	1	0	1	0	4
S71	A. Lwin et al.	1	1	1	0	0.5	0	3.5
S73	S. J. Cho et al.	1	1	1	0	1	0	4
S74	J. Wang et al.	1	1	1	0	1	0	4
S75	X. Chu et al.	1	1	1	0	1	0	4
S76	G. Zhang et al.	1	1	0.5	0	1	0	3.5
S77	D. R. Kartchner et al.	1	1	0.5	0	1	0	3.5
S79	R. Klus et al.	1	1	1	0	1	0	4
S80	M. Y. Klimenko et al.	1	1	0.5	0.5	0.5	0	3.5
S82	Y. Liu et al.	1	1	1	1	1	0.5	5.5
S84	A. Hu et al.	1	1	0.5	0	1	0	3.5
S86	A. R. Kazemi et al.	1	1	0	0.5	1	0.5	4
S87	Y. Yang et al.	1	1	0.5	1	1	0	4.5
S88	Q. Yan et al.	1	1	1	0	1	0	4
S89	J. -. De Boer et al.	1	1	0.5	0	1	1	4.5
S90	J. Reynolds et al.	1	1	1	1	1	1	6
S91	R. Imam et al.	1	1	1	0	1	0	4
S92	S. Li et al.	1	1	0.5	0	1	0	3.5
S93	Y. Su et al.	1	1	0.5	1	1	0	4.5
S94	Z. Ã–ZDEMÄ°R et al.	1	1	0.5	0	1	0	3.5
S95	W. Ye et al.	1	1	1	0	0.5	0	3.5
S98	Y. Luo et al.	1	1	0.5	0.5	1	0.5	4.5
S99	Y. Wang et al.	1	1	1	1	1	0	5
S100	Q. Yan et al.	1	1	1	0	1	0	4
S101	L. Cong et al.	1	1	1	0	1	0	4
S102	L. He et al.	1	1	1	0	1	0	4
S103	J Mendez-Astudill et al.	1	1	1	0	1	0	4
S104	M Kiani et al.	1	1	1	1	1	0	5
S105	M Kiani et al.	0.5	1	0.5	1	1	0	4
S106	Liu et al.	1	1	0.5	0.5	0.5	0	3.5
S107	ES Fogarty et al.	1	1	1	0	1	0	4
S108	K Maschera et al.	1	1	1	0	1	0	4
S110	M Łoś et al.	1	1	1	0	1	0	4
S111	Y Jia et al.	1	1	1	0	1	0	4
S112	M Kiani et al.	1	1	1	1	1	0	5
S113	Alessandro Neri et al.	1	1	0.5	0	0.5	0.5	3.5
S114	Y Zhu et al.	1	1	1	0	1	0	4
S115	Y Liu et al.	1	1	1	0	1	0	4
S116	H Xu et al.	1	1	1	0	1	0	4
S117	V Senyurek et al.	1	1	1	0	1	0	4
S118	Y Jia et al.	1	1	1	0	1	0	4
S119	Q Yuan et al.	1	1	1	0	1	0	4
S121	A Lwin et al.	1	1	1	0	0.5	0	3.5
S122	H Liu et al.	1	1	1	0	1	0	4
S124	J Wang et al.	1	1	1	1	1	0	5
S125	N Liu et al.	1	1	1	0	1	0	4
S126	HU Kim et al.	1	1	1	0	1	0	4
S127	S Li et al.	1	1	1	0	1	0	4
S130	M Kaselimi et al.	1	1	1	0	1	0	4

TABLE VII
(Continued.)

S131	K Kasantikul et al.	1	1	1	0	1	0	4
S132	J Hu et al.	1	1	1	0	1	0	4
S132	J Hu et al.	1	1	1	0	1	0	4
S133	Dongchan et al.	1	1	0.5	0	0.5	1	4
S134	V Otugo et al.	1	1	1	0	1	0	4
S136	X Zou et al.	1	1	1	0	1	0	4
S137	O Eroglu et al.	1	1	1	0	1	0	4
S138	M Kiani et al.	1	1	1	1	1	1	6
S139	E. Piccolomini et al.	1	1	1	0	1	0	4
S140	M. Moses et al.	1	1	1	0	1	0	4
S141	Veronez et al.	1	1	1	1	1	0	5
S142	Y Liang et al.	1	1	1	0	1	0	4
S144	Y Shi et al.	1	1	1	0	0.5	0	3.5
S143	M Zeybek et al.	1	1	1	1	1	0	5
S145	L Zhao et al.	1	1	0.5	0	1	0	3.5
S146	M Kaselimi et al.	1	1	1	1	1	1	6
S147	S Miyazawa et al.	1	1	1	0	0.5	0	3.5
S148	Shamshiri et al.	1	1	1	1	1	1	6
S149	Surisetty et al.	1	1	1	1	1	0	5
S150	Mutchakayala et al.	1	1	1	0	1	0	4
S151	Wojtusiak et al.	1	1	1	0	1	0	4
S152	Osah et al.	1	1	1	0	1	0	4
S153	Salar et al.	1	1	1	0	1	0	4
S155	Xia et al.	1	1	1	1	1	0.5	5.5
S156	Li et al.	1	1	1	0	1	0	4
S157	Mohammed et al.	1	1	1	0	1	0	4
S158	Okoh et al.	1	1	1	1	1	0	5
S159	Rafatnia et al.	1	1	1	1	1	0	5
S160	Sahu et al.	1	1	1	1	1	0	5
S161	Sivavaraprasad et al.	1	1	1	1	1	0	5
S162	Ferreira et al.	1	1	1	1	1	1	6
S163	P. Prešeren et al.	1	1	1	0	1	1	5
S164	Rahimi et al.	1	1	1	1	1	0	5
S165	Kemal Tütüncü et al.	1	1	1	1	1	1	6
S166	R.E. Guinness et al.	1	1	1	0	1	0	4
S167	N. Yamaga et al.	1	1	1	1	1	0	5
S168	Q. Yuan et al.	1	1	1	0	1	0	4
S169	L. Li et al.	1	1	1	0	1	0	4
S170	B. Huang et al.	1	1	1	1	1	0.5	5.5
S171	M. Kim et al.	1	1	1	1	1	0	5
S172	B. Zhang et al.	1	1	1	0	1	0	4
S173	C. Herbert et al.	1	1	1	0	1	0	4
S174	Taro Suzuki et al.	1	1	1	0	1	0	4
S175	Marco Mendonça et al.	1	1	1	0	1	0	4
S179	Selbesoglu et al.	1	1	1	1	1	0	5
S180	Mohamad Orabi et al.	1	1	1	0.5	0.5	0	4
S181	Yu Jiao et al.	1	1	1	0	1	0	4
S183	Hany Ragabet et al.	1	1	1	0	1	0	4
S185	J. Merwe et al.	1	1	0.5	0	1	0	3.5
S186	Yung-Cheng et al.	1	1	1	0.5	1	0	4.5
S187	Hamad Yousif et al.	1	1	1	0.5	0.5	0.5	4.5
S188	J. Wang et al.	1	1	0.5	0.5	0.5	0	3.5
S190	T. Désert et al	1	1	1	0.5	1	0.5	5
S191	Li He et al	1	1	0.5	0	1	0.5	4
S192	W. Vigneau et al.	1	1	1	0	1	0.5	4.5
S194	Ramos-Bosch et al.	1	1	1	0	1	0	4
S195	Chengquan Xu et al.	1	1	1	0.5	1	0	4.5
S196	Heekwon No et al	1	1	1	0	1	0.5	4.5
S197	Qiming Zhong et al	1	1	1	0	1	0.5	4.5
S198	Chengjun Guo et al	1	1	1	0	1	0	4
S199	N. HARBAOUI et al	1	1	1	0	1	0	4
S200	S. Kozhaya et al	1	1	0.5	0	1	0	3.5
S201	Adyasha Mohanty et al	1	1	1	0	1	0	4
S202	Kahn-Bao Wu et al	1	1	1	0	1	0	4
S203	A. Kanhere et al	1	1	1	0	1	1	5
S204	A. Siemuri et al	1	1	1	0	1	0	4
S207	G. Caparra et al	0.5	1	1	0	0.5	0.5	3.5
S208	N. Ziedan	1	1	1	0	0.5	0	3.5
S209	A. Gomez et al	1	1	1	0	1	0	4
S210	Lei Liu et al	1	1	1	0	0.5	0	3.5
S211	Yunxiang Liu et al	1	1	0.5	0	1	0	3.5
S213	Quoc-Huy Phan et al	1	1	1	0	0.5	0	3.5
S214	Azami Hamed et al	1	1	1	0	1	0	4
S215	Nadali Zarei	1	1	0.5	0	1	0.5	4
S216	A. Noureldin et al	1	1	0.5	0	1	0	3.5
S217	Yu Jia et al	1	1	1	0	1	0.5	4.5
S218	E. Abdolkarimi et al	1	1	0.5	0	1	0	3.5
S220	A. Elnaggar	1	1	1	0.5	0.5	0	4
S221	Yiming Quan et al	1	1	1	0	1	0	4
S222	Habarulema et al	1	1	1	0.5	0.5	0.5	4.5
S223	J. Habarulema et al	1	1	1	0	0.5	0.5	4
S224	R. Sharaf et al	1	1	1	0	1	0	4
S225	C. Pikridas et al	1	1	0.5	0	1	0	3.5
S226	P. Benevides et al	1	1	1	0	1	0	4
S227	G. Panice et al	1	1	0.5	0.5	1	0	4
S228	Rui Sun et al	1	1	1	0.5	0.5	0	4
S229	Mosavi et al	1	1	1	0	0.5	0	3.5
S231	Li Jing et al	1	1	0.5	0	0.5	0.5	3.5
S232	Yimin Zhou et al	1	1	1	0	1	0	4
S233	Guangcai Wang et al	1	1	0.5	0	1	0	3.5
S234	Fangni Lei et al	1	1	1	0	1	0	4
S235	Zhengxie Zhang et al	1	1	1	0	1	0	4
S236	Li et al.	1	1	1	0	1	0	4
S237	Wu et al	1	1	0.5	0	1	0	3.5
S238	Savas et al	1	1	0.5	0.5	0.5	0.5	4
S239	Liu et al	1	1	1	0	1	1	5
S241	David et al	1	1	1	0	0.5	1	4.5
S242	Huang et al	1	1	1	0	1	1	5
S243	Yilmaz et al	1	1	0.5	0	1	0	3.5
S244	Habarulema et al	1	1	1	0	0.5	0	3.5
S245	Sabzehee et al.	1	1	1	0	1	0	4
S246	Leandro et al	1	1	1	0	1	1	5
S247	Wang et al	1	1	0.5	0	1	1	4.5
S248	Lyu et al	1	1	1	0	1	1	5
S249	Shafiee et al	1	1	1	0	1	1	5

TABLE VIII
Selected studies

ID	Authors	Research Questions Addressed (#RQ)						Ref.	ID	Authors	Research Questions Addressed (#RQ)						Ref.
S1	Simon et al.	1	2			4b	5	[144]	S127	S Li et al.	1	2	3		4b		[209]
Q3	Machado et al.	1	2	3		4b		[189]	S130	M Kaselimi et al.	1	2	3		4b		[210]
S4	Bhatt et al.	1	2	3		4b		[68]	S131	K Kasantikul et al.	1	2	3		4b		[45]
S5	M. Socharoentum et al.	1	2	3		4b		[180]	S132	J Hu et al.	1	2	3		4b		[179]
S6	Z. Zhou et al.	1	2	3	4a	4b	5	[71]	S133	Dongchan et al.	1	2	3		4b	5	[211]
S7	Jiao et al.	1	2	3		4b		[97]	S134	V Otugo et al.	1	2	3		4b		[92]
S9	Favenza et al.	1	2	3	4a	4b		[94]	S136	X Zou et al.	1	2	3		4b		[212]
S10	Y. Quan et al.	1	2	3		4b		[3]	S137	O Eroglu et al.	1	2	3		4b		[38]
S11	Yang et al.	1	2	3		4b		[190]	S138	M Kiani et al.	1	2	3	4a	4b	5	[166]
S12	L.-T. Hsu et al.	1	2	3		4b	5	[191]	S139	E Loli Piccolomini et al.	1	2	3		4b		[15]
S14	Suzuki et al.	1	2	3		4b	5	[192]	S140	M Moses et al.	1	2	3		4b		[213]
S15	Jiao et al.	1	2	3		4b		[98]	S141	MR Veronez et al.	1	2	3	4a	4b		[73]
S16	B. GUERMAH et al.	1	2	3		4b		[16]	S142	Y Liang et al.	1	2	3		4b		[57]
S18	Y. Liu et al.	1	2	3		4b		[89]	S143	Y Shi et al.	1	2	3	4a	4b		[58]
S19	Y. Quan et al.	1	2	3		4b	5	[158]	S144	M Zeybek et al.	1	2	3		4b		[11]
S20	P. Huang et al.	1	2	3	4a	4b		[143]	S145	L Zhao et al.	1	2	3		4b		[214]
S21	Gogliettino et al.	1	2	3	4a	4b	5	[159]	S146	M Kaselimi et al.	1	2	3	4a	4b	5	[167]
S22	Zhidong Zhang et al.	1	2	3		4b		[193]	S147	S Miyazawa et al.	1	2	3		4b		[74]
S23	Z. Liu et al.	1	2	3		4b		[137]	S148	Shamshiri et al.	1	2	3	4a	4b	5	[107]
S25	H.-U. Kim and T.-S. Bae	1	2	3		4b	5	[69]	S149	Surisetty et al.	1	2	3	4a	4b		[59]
S26	N. Linty et al.	1	2	3		4b	5	[95]	S150	Mutchakayala et al.	1	2	3		4b		[215]
S27	M. R. Manesh et al.	1	2	3		4b	5	[117]	S151	Wojtusiak et al.	1	2	3		4b		[75]
S28	ALEJANDRO KURATOMI	1	2	3		4b		[63]	S152	Osah et al.	1	2	3		4b		[216]
S29	Q. Liu et al.	1	2	3		4b		[87]	S153	Salar et al.	1	2	3		4b		[217]
S30	R. Morales Ferre et al.	1	2	3		4b	5	[122]	S155	Xia et al.	1	2	3	4a	4b	5	[168]
S31	A. LOUIS	1	2	3		4b		[194]	S156	Li et al.	1	2	3		4b		[46]
S32	K. Lamb et al.	1	2	3		4b	5	[195]	S157	Mohammed et al.	1	2	3		4b		[111]
S33	S. Semanjski et al.	1	2	3		4b	5	[120]	S158	Okoh et al.	1	2	3	4a	4b		[169]
S34	R. Orus Perez	1	2	3		4b	5	[104]	S159	Rafatnia et al.	1	2	3	4a	4b		[172]
S35	R. Sun et al.	1	2	3		4b	5	[13]	S160	Sahu et al.	1	2	3	4a	4b		[170]
S36	D. Brum et al.	1	2	3		4b		[22]	S161	Sivavaraprasad et al.	1	2	3	4a	4b		[171]
S37	H. Dai et al.	1	2	3		4b	5	[135]	S162	Ferreira et al.	1	2	3	4a	4b	5	[173]
S38	Z. Zou et al.	1	2	3	4a	4b	5	[129]	S163	P. Pavlovčič Prešeren et al.	1	2	3		4b	5	[83]
S39	Chang et al.	1	2	3	4a	4b		[82]	S164	Rahimi et al.	1	2	3	4a	4b		[60]
S40	E. Munin et al.	1	2	3		4b	5	[181]	S165	Kemal Tütüncü et al.	1	2	3	4a	4b	5	[239]
S42	Li et al.	1	2	3	4a	4b	5	[161]	S166	R.E. Guinness et al.	1	2	3		4b		[86]
S43	Borhani-Darian et al.	1	2	3		4b		[182]	S167	N. Yamaga et al.	1	2	3	4a	4b		[160]
S44	P. Borhani-Darian and P. Closas	1	2	3		4b		[12]	S168	Q. Yuan et al.	1	2	3		4b		[186]
S45	S. Tohidi and M. R. Mosavi	1	2	3	4a	4b		[118]	S169	L. Li et al.	1	2	3		4b		[218]
S46	Munin et al.	1	2	3		4b		[196]	S170	B. Huang et al.	1	2	3	4a	4b	5	[174]
S47	M. Alshaye et al.	1	2	3		4b		[24]	S171	M. Kim et al.	1	2	3	4a	4b		[105]
S48	Yunxiang Liu et al.	1	2	3		4b		[183]	S172	B. Zhang et al.	1	2	3		4b		[61]
S49	L. Mallika l et al.	1	2	3		4b		[106]	S173	C. Herbert et al.	1	2	3		4b		[30]
S50	Suzuki et al.	1	2	3		4b		[197]	S174	Taro Suzuki et al.	1	2	3		4b		[17]
S51	M. O. Selbesoglu	1	2	3		4b		[109]	S175	Marco Mendonça et al.	1	2	3		4b		[240]
S52	Y. Xia et al.	1	2	3		4b		[85]	S179	Mahmut Oguz Selbesoglu et al.	1	2	3	4a	4b		[112]
S53	Q. Yan and W. Huang	1	2	3		4b	5	[26]	S180	Mohamad Orabi et al.	1	2	3	4a	4b		[163]
S54	R. Calvo-Palomino et al.	1	2	3		4b		[119]	S181	Yu Jiao et al.	1	2	3		4b		[90]
S55	Haiyu et al.	1	2	3		4b		[198]	S183	Hany Ragabet et al.	1	2	3		4b		[219]
S56	S. Semanjski et al.	1	2	3		4b		[238]	S185	J. Rossouw van der Merwe et al.	1	2	3		4b		[124]
S57	S. Semanjski et al.	1	2	3		4b		[199]	S186	Yung-Cheng et al.	1	2	3	4a	4b		[138]
S58	F. Dovis et al.	1	2	3		4b		[200]	S187	Hamad Yousif and Ahmed El-Rabbany	1	2	3	4a	4b	5	[164]
S59	Y. Jia et al.	1	2	3		4b		[32]	S188	Jianguo Jack Wang et al.	1	2	3	4a	4b		[139]
S60	E. I. Adegoke et al.	1	2	3		4b		[201]	S190	T. Désert et al	1	2	3	4a	4b	5	[103]
S61	Y. Jia et al.	1	2	3		4b		[33]	S191	Li He et al	1	2	3		4b	5	[20]
S62	Q. Yan and W. Huang	1	2	3		4b		[27]	S192	W. Vigneau et al.	1	2	3		4b	5	[147]
S63	M. Asgarimehr et al.	1	2	3	4a	4b		[40]	S194	Pere Ramos-Bosch et al.	1	2	3		4b		[148]
S65	L. Miotti et al.	1	2	3	4a	4b		[110]	S195	Chengquan Xu et al.	1	2	3	4a	4b		[175]
S67	Y. Liu et al.	1	2	3		4b		[41]	S196	Heekwon No et al	1	2	3		4b	5	[220]
S68	T. Mortlock and Z. M. Kassas	1	2	3	4a	4b		[146]	S197	Qiming Zhong et al	1	2	3		4b	5	[221]
S69	L. Mengying et al.	1	2	3		4b		[99]	S198	Chengjun Guo et al	1	2	3		4b		[21]
S71	A. Lwin et al.	1	2	3		4b		[35]	S199	Nesrine HARBAOUI et al	1	2	3		4b		[222]
S73	S. J. Cho et al.	1	2	3		4b		[184]	S200	Sharbel E. Kozhaya et al	1	2	3		4b		[149]
S74	J. Wang et al.	1	2	3		4b		[202]	S201	Adyasha Mohanty et al	1	2	3		4b		[134]
S75	X. Chu et al.	1	2	3		4b		[42]	S202	Kahn-Bao Wu et al	1	2	3		4b		[223]
S76	G. Zhang et al.	1	2	3		4b		[70]	S203	Ashwin V. Kanhere et al	1	2	3		4b	5	[76]
S77	D. R. Kartchner et al.	1	2	3		4b		[123]	S204	Akpojoto Siemuri et al	1	2	3		4b		[77]
S79	R. Klus et al.	1	2	3		4b		[88]	S207	Gianluca Caparra et al	1	2	3		4b	5	[78]
S80	M. Y. Klimenko et al.A. V. Veitsel	1	2	3	4a	4b		[203]	S208	Nesreen I. Ziedan	1	2	3		4b		[79]
S82	Y. Liu et al.	1	2	3	4a	4b	5	[43]	S209	Annabel R. Gomez et al	1	2	3		4b		[91]
S84	A. Hu et al.	1	2	3		4b		[115]	S210	Lei Liu et al	1	2	3		4b		[224]
S86	A. R. Kazemi et al.	1	2		4a	4b	5	[165]	S211	Yunxiang Liu et al	1	2	3		4b		[225]
S87	Y. Yang et al.	1	2	3	4a	4b		[114]	S213	Quoc-Huy Phan et al	1	2	3		4b		[226]
S88	Q. Yan et al.	1	2	3		4b		[25]	S214	Azami Hamed et al	1	2	3		4b		[236]
S89	J. -. De Boer et al.	1	2	3		4b	5	[132]	S215	Nadali Zarei	1	2	3		4b	5	[237]
S90	J. Reynolds et al.	1	2	3	4a	4b	5	[44]	S216	AboelmagdNoureldin et al	1	2	3		4b		[140]
S91	R. Imam et al.	1	2	3		4b		[96]	S217	Yu Jia et al	1	2	3		4b	5	[100]
S92	S. Li et al.	1	2	3		4b		[47]	S218	E. S. Abdolkarimi et al	1	2	3		4b		[131]
S93	Y. Su et al.	1	2	3	4a	4b		[49]	S220	Aly M.El-naggar	1	2	3	4a	4b		[93]
S94	Z. Ã–ZDEMÄ°R et al.	1	2	3		4b		[204]	S221	Yiming Quan et al	1	2	3		4b		[227]
S95	W. Ye et al.	1	2	3		4b		[205]	S222	Habarulema et al	1	2	3	4a	4b	5	[176]
S98	Y. Luo et al.	1	2	3	4a	4b	5	[162]	S223	John Bosco Habarulema et al	1	2	3		4b	5	[228]
S99	Y. Wang et al.	1	2	3	4	4b		[14]	S224	R. Sharaf et al	1	2	3		4b		[136]
S100	Q. Yan et al.	1	2	3		4b		[28]	S225	Christos Pikridas et al	1	2	3		4b		[113]
S101	L. Cong et al.	1	2	3		4b		[133]	S226	Pedro Benevides et al	1	2	3		4b		[62]
S102	L. He et al.	1	2	3		4b		[206]	S227	G. Panice et al	1	2	3	4a	4b		[126]
S103	J Mendez-Astudill et al.	1	2	3		4b		[50]	S228	Rui Sun et al	1	2	3	4a	4b		[80]
S104	M Kiani et al.	1	2	3	4a	4b		[51]	S229	Mosavi et al	1	2	3		4b		[81]
S105	M Kiani et al.	1	2	3	4a	4b		[52]	S231	Li Jing et al	1	2	3		4b	5	[187]
S106	Liu et al.	1	2	3	4a	4b		[101]	S232	Yimin Zhou et al	1	2	3		4b		[127]
S107	ES Fogarty et al.	1	2	3		4b		[207]	S233	Guangcai Wang et al	1	2	3		4b		[128]
S108	K Maschera et al.	1	2	3		4b		[185]	S234	Fangni Lei et al	1	2	3		4b		[229]
S110	M Łoś et al.	1	2	3		4b		[54]	S235	Zhengxie Zhang et al	1	2	3		4b		[108]
S111	Y Jia et al.	1	2	3		4b		[34]	S236	Li et al.	1	2	3		4b		[121]
S112	M Kiani et al.	1	2	3	4a	4b		[53]	S237	Wu et al	1	2	3		4b		[18]
S113	Alessandro Neri et al.	1	2	3		4b	5	[72]	S238	Savas et al	1	2	3	4a	4b	5	[177]
S114	Y Zhu et al.	1	2	3		4b		[29]	S239	Liu et al	1	2	3		4b	5	[230]
S115	Y Liu et al.	1	2	3		4b		[102]	S241	David et al	1	2	3		4b	5	[231]
S116	H Xu et al.	1	2	3		4b		[208]	S242	Huang et al	1	2	3		4b	5	[232]
S117	V Senyurek et al.	1	2	3		4b		[39]	S243	Yilmaz et al	1	2	3		4b		[188]
S118	Y Jia et al.	1	2	3		4b		[36]	S244	Habarulema et al	1	2	3		4b		[233]
S119	Q Yuan et al.	1	2	3		4b		[48]	S245	Sabzehee et al.	1	2	3		4b		[178]
S121	A Lwin et al.	1	2	3		4b		[37]	S246	Leandro et al	1	2	3		4b	5	[234]
S122	H Liu et al.	1	2	3		4b		[55]	S247	Wang et al	1	2	3		4b	5	[235]
S124	J Wang et al.	1	2	3	4a	4b		[31]	S248	Lyu et al	1	2	3		4b	5	[19]
S125	N Liu et al.	1	2	3		4b		[130]	S249	Shafiee et al	1	2	3		4b	5	[125]
S126	HU Kim et al.	1	2	3		4b		[56]									

TABLE IX
Strengths and Weaknesses of Mostly Used ML Techniques

Strengths	Weaknesses	Studies Ref.
DT and ensemble DTs 1. Decision trees can learn non-linear relationships and are fairly robust to outliers. 2. It can avoid overfitting by pruning or making use of ensembles. 3. Intuitive and easy to understand.	1. Unconstrained, individual trees are prone to overfitting as they can keep on branching until the training data is memorized.	[13, 16, 29, 63, 94–96, 124, 180, 201]
DL models 1. Perform very well on images, audio, and text data. 2. They can be updated easily with new datasets using batch propagation. 3. Their architectures (that is, the number of layers and the structure of the layers) can be adapted to fit many types of problems. 4. Capable of dealing with noisy data. 5. Hidden layers reduce the need for feature engineering.	1. Require a very large amount of data (therefore, cannot be used as a general-purpose algorithm). 2. Computationally intensive to train. 3. Require much more experience to tune them (that is, hyperparameters tunning). 4. Weak explanatory ability.	NN models: [3, 12, 17, 18, 20, 21, 25, 27, 30, 38–41, 43–46, 56, 60, 62, 73, 77, 78, 81, 82, 86, 88, 92, 93, 103, 105, 109–113, 115, 118, 123, 125, 130, 132, 134, 136, 138–140, 144, 147, 148, 160, 163–165, 169–171, 173, 175, 176, 178, 184, 186, 188, 189, 195–197, 203, 210, 212, 213, 227, 232–234, 236, 237] and LSTM: [15, 56, 69, 74, 91, 97, 123, 130, 149, 167, 174, 217, 225]
LSTM (a special type of RNN) 1. Easy to implement. 2. Scaleable with the dataset. 3. Perform well even with small data sets.	**LSTM (a special type of RNN)** 1. Outperformed by NN models that have been properly trained and tuned. 2. LSTMs take longer to train. 3. LSTMs require more memory to train.	
SVM 1. Fairly robust against overfitting, especially in high-dimensional space. 2. Used to model non-linear decision boundaries with many kernels to choose from.	1. Trickier when tuning because picking the right kernel is very important in the process. 2. They also don't scale well to larger datasets.	[14, 16, 17, 19, 28, 33, 36, 37, 39, 68, 72, 86, 89, 90, 97–102, 108, 120, 121, 126, 133, 162, 168, 184, 192, 198, 199, 207, 208, 223, 238, 240]

CNN and their combinations have been used most frequently.

2) *RQ2*—The ML algorithms were used for classification, clustering, forecasting, and anomaly detection depending on the GNSS use case. Some of these GNSS use cases include signal acquisition, signal detection and classification, Earth observation, GNSS/INS integration, anomaly detection, and spoofing and jamming detection, etc.
3) *RQ3*—Some of the data used were simulated data gotten from different simulation tools and SDRs, while others were real data collected using GNSS receivers. The utilized datasets can be publicly available (for free) or private in nature (not shared by researchers; therefore, results on such datasets cannot be verified and such studies are not replicable).
4) *RQ4a and RQ4b*—Regarding ML performance, in general, ML model is more accurate than non-ML model, which has been supported by most studies. Regression model and other traditional GNSS models depending on the GNSS use case were compared with ML models in 50 out of 213 selected studies.
5) *RQ5*—Different ML techniques have different strengths and weaknesses and thus favorable to different GNSS applications. The strengths and weaknesses of the studies based on the implemented ML algorithms were extracted from the selected studies and presented in this review. This was done because the quality assessment process of the selected studies can help ensure that they are from studies with acceptable quality. The strength and weaknesses from a list comprising of the most implemented algorithm in the selected studies was also presented.
6) *RQ6*—Regarding validation, it has been shown from the selected studies that for the validation of the ML models, most studies made use of Holdout, *n*-fold cross-validation, and comparison with another algorithms. While for accuracy metric, most made use of RMSE, ROC, MSE, and StD. Furthermore,

TABLE X
Strengths and Weaknesses of the approaches used in the Selected Studies

ID	ML Technique Used	Strengths	Weaknesses	Ref.
S6	Least-squares support vector machine (LSSVM) technique	Unlike classical KF, it adaptively identifies the dynamic model bias, which is then used to compensate for the dynamic model.		[71]
S12	Support Vector Machine (SVM)		Classifier developed is applied only for static applications	[191]
S19	Convolutional Neural Network (CNN)	The proposed methods can be used in carrier phase-based kinematic positioning, including RTK applications.		[158]
S21	MLP, Autoencoder network	Autoencoder approach is very promising as it needs samples not affected by errors.		[159]
S25	Long-Short Term Memory (LSTM)		Needs to be elaborated to incorporate sensors with more features as input data and the strategy on how to balance the weight between sensor data.	[69]
S26	Decision tree	Machine learning helps in facilitating the work of analyzing big sets of GNSS data affected by amplitude scintillation		[95]
S27	Neural Network (NN)		Need to include online learning in the neural network or use unsupervised algorithms	[117]
S30	Support vector machine (SVM)		Classification will be more accurate if the complexity of the training layers is increased	[122]
S32	Convolutional Neural Network (CNN)	A novel methodology which predicts GNSS phase scintillations 1 hour in advance.		[195]
S33	C-Support Vector Machines (C-SVM)	Correlation analysis is a good approach for the selection of variables that serve as an input for the supervised machine learning approach.		[120]
S34	Neural Network (NN)		Applicability should be assessed if other techniques such as convolutional neural networks are used and the hardware requirements for analyzing large amounts of data.	[104]
S35	Gradient boosting decision tree (GBDT)	Removing NLOS based on the proposed method can improve the static positioning accuracy to some extent		[13]
S37	Recurrent neural network (RNN)	Anticipates proposed method can be applied in the field of multi-sensors integrated navigation system.		[135]
S38	Convolutional Neural Network (CNN)		The limitation of the proposed algorithm is that one trained estimator is bound to specific sensors. The neural network model should be retrained when used on a new navigation system.	[129]
S40	Convolutional Neural Network (CNN)		Proposed algorithm needs to be validated with the real signals	[181]
S42	Deep Neural Network (DNN)		The investigated NN model requires a multi-correlation scheme, thus involving an increased computational cost when compared to standard methods.	[161]
S53	Convolutional Neural Network (CNN)	The usage of filters in the convolution layer reduces the noise in the DDM.		[26]
S82	Neural Network (NN)		Further investigation on the relationship between the wind speed or the wind vector and the last hidden layer neurons can potentially provide a better understanding of the underlying physical meaning of the network	[43]
S86	Neural Network (NN)	Classification algorithm outperforms the original PD- ML detector and PSO-NN classifier, but it comes with more computation complexity		[165]
S89	Neural Network (NN)	NN-aided GNSS/MEMS integration provides more accurate position estimates than GNSS/MEMS without NN corrections.		[132]
S90	Artificial Neural Network (ANN)		The ANN methodology has difficulty to estimate unusual occurrences of very low or very high wind speed and the need for a large training set to obtain accurate retrievals over a time frame of several months or years.	[44]
S98	Support vector machine (SVM)	The proposed method does not require the radio telescope, and can achieve all-time, all-weather detection by processing large quantities of data at the same time and the detection results demonstrate whether multiple stations are affected by SRBs.		[162]
S133	Deep Neural Network (DNN)	SVM has low performance than the NN model while RNN performance is the best, but needs initial time and takes a long time to train and DNN has less performance than RNN but does not require initial time and is faster to train.		[211]
S138	Generalized Regression NN	It is purely a mathematical model with high accuracy, up to centimeter level. Including observation accuracy and the physical conditions of the environment may lead to a more accurate algorithm, capable of achieving higher accuracies, possibly up to the millimeter level.		[166]
S146	Long-Short Term Memory (LSTM)	Inclusion of selected features in the supervised LSTM algorithm and that LSTM networks are equipped with memory cell which holds information content in the input STEC data, gives superiority and an extra boost to the proposed method. Non-linearity and long-term prediction are additional advantages of the proposed LSTM method.		[167]
S148	GP regression	The approach has a good generalization capability even with a small set of training samples. Wider range of sampling results in better generalization capabilities.		[107]
S155	Support vector machine (SVM)		The model cannot accurately predict the ionospheric TEC in high years of solar activity	[168]
S162	Neural Network (NN)		The NN model does not have the ability to calibrate vTEC by itself, it relies on data provided by the calibration technique.	[173]
S163	Wavelet Neural Network (WNN)		The problem of optimal wavelet network adjustment remains and because of that wavelet function selection should always be based on practical experimentation and trial and error tests.	[83]
S170	Long-Short Term Memory (LSTM)	SL-LSTM method achieves a better long-term prediction accuracy and stability than the other three methods. The quality of satellite clock bias prediction is better than that of other three methods.		[174]
S187	Neural Network (NN)		The ANN consumes computer resources while training large dataset	[164]
S190	Neural Network (NN)	The proposed method is effect on an equatorial region		[103]
S191	Artificial Neural Network (ANN)	Installing an antenna with a non-trivial and confidential radiation map, the security of the GPS signal for a specific fixed receiver can be increased.		[20]
S192	Neural Network (NN)		The performances and complexity of both algorithms has been analyzed and those results are to be finalized to assess the cost of integration of those techniques inside a LEO GNSS receiver.	[147]
S196	Quantile Regression	The proposed model is well overbounding up to desired probability achieving better position accuracy, lower alarm rate, and tighter protection level.		[220]
S197	Bayesian Filter	Filtering has a greater impact on the results of the mobile positioning with significant movement compared to static positioning		[221]
S203	Deep Neural Network (DNN)	Using pseudorange residuals and LOS vectors from the initial position guess as inputs and NED position corrections as outputs to the DNN improves the numerical conditioning of the DNN and provides global applicability to the algorithm.		[76]
S207	Neural Network (NN)	The method does not require changes in the architecture of the GNSS receivers and can be deployed as a software service.		[78]
S215	Neural Network (NN) trained with PSO, NPSO, GA, and ICA		GPS GDOP approximation and classification are time-consuming and unreliable using existing approaches	[237]
S217	Support vector machine (SVM)		New phase scintillation detectors based on phase observations need to be developed	[100]
S222	Neural Network (NN)		The availability of historical data affects the NN model	[176]
S223	Feed Forward Neural Networks (FFNN)	NN model makes more accurate predictions on TEC than GPS-derived TEC due to the availability of data within the NN model from nearby stations.		[228]
S231	Ensemble learning algorithm (LS-Boost or Bagging)		The accuracy of the proposed algorithm is not good as SINS error does not have noticeable divergence in a short period of time	[187]
S238	K-means clustering		Implementation parameters of the algorithm must be well optimized.	[177]
S239	convLSTM	convLSTM-based architecture forecasts an entire regions' ionospheric irregularity occurrence and intensity values		[230]
S241	Neural Network (NN)	MSEs were very low for most of the months, hence accuracy is high		[231]
S242	Radial basis function (RBF) Neural Network (RBF-NN)	RBF network is a reliable and alternate tool for the ionospheric TEC forecast of single station		[232]
S246	Neural Network (NN)		The TEC values for each station based on the technique used is not an optimal approach as it depends on ambiguity term	[234]
S247	Wavelet Neural Network (WNN)		The architecture of WNN employed is not guaranteed to be the best and unique design due to the theoretical limitations inhering in ANN	[235]
S248	Support vector machine (SVM)	The proposed weight scheme is superior to the traditional weight scheme as it can better model the GNSS measurement NLOS error in urban environments.		[19]
S249	n Multi-layer NN	Spoofing detection using the algorithm after training is low-cost, easy to implement, and reliable		[125]

TABLE XI
Algorithms, GNSS Use Case, Data Type, and Validation Method

ID	Authors	Year	ML Technique	Study Application	Data type	Accuracy/Validation	Type	Ref.
S1	Simon et al.	1995	Neural Network (NN)	Satellite selection	Real	Not mentioned	Journal	[144]
S3	Machado et al.	2011	Artificial Neural Networks (ANN)	Earth monitoring	Real	Root-mean-squared (RMS) and standard-deviation (StD)	Conference	[189]
S4	Bhatt et al.	2012	Support Vector Machines (SVM)	GNSS navigation	Real	not mentioned	Conference	[68]
S5	Socharoentum et al.	2016	Logistic Regression (LR), SVM, Naïve Bayes (NB), and Decision Tree (DT)	NLOS Detection	Real	Validation data set	Conference	[180]
S6	Z. Zhou et al	2016	Least-squares SVM (LSSVM)	GNSS navigation	Real	k-fold cross-validation	Journal	[71]
S7	Jiao et al.	2016	SVM	Ionospheric scintillation	Real	Receiver operating characteristic (ROC) curves and confusion matrices and hold-out for validation.	Conference	[97]
S9	Favenza et al.	2017	DT	GNSS Scintillation	Real	Cross-validation, and F-score	Conference	[94]
S10	Y. Quan	2017	ANN, Convolutional Neural Network (CNN), Random Forest (RF)	Multipath Detection	Real and Simulated	Not mentioned	PhD thesis	[3]
S11	Yang et al.	2017	Deep Neural Network (DNN)	GNSS Interference	Real	Not mentioned	Conference	[190]
S12	L.-T. Hsu	2017	Recurrent Neural Network (RNN)	Multipath Detection	Real	Not mentioned	Conference	[191]
S14	Suzuki et al.	2017	SVM	Multipath Detection	Real	Using different numbers of correlation outputs	Conference	[192]
S15	Jiao et al.	2016	SVM	Ionospheric scintillation	Real	Five-fold cross-validation, receiver operating characteristic (ROC) curves and confusion matrices	Journal	[98]
S16	B. GUERMAH et al.	2018	DT, SVM and KNN	LOS/Multipath Signal Classifier	Real	Not mentioned	Conference	[16]
S18	Y. Liu et al.	2018	SVM	Ionospheric scintillation	Real	10-folds cross validation, and validation datasets	Conference	[89]
S19	Y. Quan et al.	2018	Convolutional Neural Network (CNN)	Multipath Detection	Real and Simulated	Simulated and Real GPS data and compared with existing multipath mitigation methods in position domain.	Journal	[158]
S20	P. Huang et al.	2018	PointNet and VoxelNet networks	Satellite selection	Real	Test data from 220 IGS stations.	Journal	[143]
S21	Gogliettino et al.	2019	Multi-Layer Perceptron (MLP), autoencoder network, LR	GNSS security	Simulated	ROC, and area under the curve (AUC)	Conference	[159]
S22	Zhidong et al	2019	NN	GNSS/INS integration	Real	Not mentioned	Conference	[193]
S23	Z. Liu et al.	2019	NN	GNSS/INS integration	Simulated	Validation test with the help of STM32.	Conference	[137]
S25	H.-U. Kim et al.	2019	LSTM	GNSS positioning	Real	Not mentioned	Journal	[69]
S26	N. Linty et al.	2019	DT	Ionospheric scintillation	Real	Not mentioned	Journal	[95]
S27	M. R. Manesh et al.	2019	NN	Detection of GPS Spoofing Attacks	Real	K-fold cross validation	Conference	[117]
S28	A. KURATOMI	2019	DT, SVM	GNSS Position Error Estimated	Real	Root mean square error (RMSE)	Msc Thesis	[63]
S29	Q. Liu et al.	2019	NLOS and multipath detecting network (NMDN)	Indoor Navigation	Real	Support vector machine (SVM)	Journal	[87]
S30	Ferre et al.	2019	SVM	Jammer Classification	Real	Validation dataset	Journal	[122]
S31	A. LOUIS	2019	NN	Evil waveforms (EWF) detction	Real	ROC curves	Journal	[194]
S32	K. Lamb et al.	2019	CNN	Ionospheric scintillation	Real	Not mentioned	Journal	[195]
S33	S. Semanjski et al.	2019	C-Support Vector Machines (C-SVM)	Detection of GNSS Signal Spoofing	Real and Simulated	Validation dataset	Conference	[120]
S34	R. Orus Perez	2019	NN	Ionospheric delay	Real	Not mentioned	Journal	[104]
S35	R. Sun et al.	2020	Gradient Boosting Decision Tree (GBDT), Traditional Decision Tree (DT), distance weighted KNN, adaptive network-based fuzzy inference system (ANFIS)	Signal classification	Real	RMSE value	Journal	[13]
S36	D. Brum et al.	2020	ANN	Earth monitoring	Real	Matthews Correlation Coefficient (MCC), and Mean square error (MSE)	Journal	[22]
S37	H. Dai et al.	2020	RNN, Extreme Learning Machine (ELM)	GNSS/INS integration	Real and Simulated	RMSE value	Journal	[135]
S38	Z. Zou et al.	2020	CNN	GNSS/INS integration	Real	Not mentioned	Conference	[129]
S39	Chang et al.	2020	Genetic Algorithm (GA), NN	GNSS integrity	Real	Averaged and the standard deviation	Conference	[82]
S40	E. Munin et al.	2020	CNN	Multipath Detection	Real	Accuracy percentage	Conference	[181]
S42	Li et al.	2020	DNN	GNSS signal correlation	Simulated	MSE	Conference	[161]
S43	Borhani-Darian et al.	2020	Multi-layer perceptron (MLP), CNN	GNSS spoofing attack	Simulated	ROC curves	Conference	[182]
S44	Borhani-Darian et al.	2020	MLP, CNN	GNSS Signal Acquisition	Real	ROC curves	Conference	[12]
S45	S. Tohidi et al.	2020	MLP trained by Particle Swarm Optimization (PSO)	Detection of GPS Spoofing Attacks	Real	Compared with results achieved via classification based Bayes rule.	Conference	[118]
S46	Munin et al.	2020	Deep CNN	Multipath Detection	Real and Simulated	ROC curves	Conference	[196]
S47	M. Alshaye et al.	2020	CNN	Earth monitoring	Simulated	MSE	Conference	[24]
S48	Yunxiang Liu et al.	2020	RF, SVM	GNSS abnormaly detection	Real	Validation dataset, and cross-validation	Conference	[183]
S49	L. Mallika I et al.	2020	Gaussian Process Regression (GPR)	Ionospheric delay	Real	Mean Absolute Error (MAE), MeanAbsolute Percentage Error (MAPE), Mean Square Error (MSE), Root Mean Square Error (RMSE), and correlation coefficient.	Journal	[106]
S50	Suzuki et al.	2020	CNN	Multipath Detection	Real	Cross-validation	Conference	[197]
S51	M. O. Selbesoglu	2020	ANN	Tropospheric delay	Real	Comparison with the values estimated from Global Navigation Satellite System observations	Journal	[109]
S52	Y. Xia et al.	2020	RNN	Indoor Navigation	Real	Using new test sets.	Journal	[85]
S53	Q. Yan et al.	2020	CNN	Earth monitoring	Real	Reference SIC data	Journal	[26]
S54	Calvo-Palomino et al.	2020	LSTM	Detection of GNSS Signal Spoofing	Real	Not mentioned	Conference	[119]
S55	Haiyu et al.	2020	SVM	GNSS/INS integration	Real	ROC, and AUC	Conference	[198]
S56	S. Semanjski et al.	2020	Support Vector Machine classification (C-SVM), PCA	Detection of GNSS Signal Spoofing	Real and Simulated	Not mentioned	Journal	[238]
S57	S. Semanjski et al.	2020	C-SVM	Detection of GNSS Signal Spoofing	Real and Simulated	Not mentioned	Journal	[199]
S58	F. Dovis et al.	2020	K-means classes, SVM	Multi-path, interference and atmospheric limitations	Real	Not mentioned	Conference	[200]
S59	Y. Jia et al.	2019	RF	Earth monitoring	Simulated	Not mentioned	Conference	[32]
S60	E. I. Adegoke et al.	2019	DT	GNSS navigation	Real	Not mentioned	Conference	[201]
S61	Y. Jia; et al.	2021	RF, SVM, XGBoost, ANN	Earth monitoring	Real	Not mentioned	Journal	[33]
S62	Q. Yan et al.	2020	CNN, SVR	Earth monitoring	Real	TDS-1 measurements in 2017 and 2018 of thin sea ice with thickness less than 1 m.	Conference	[27]
S63	M. Asgarimehr; et al.	2020	NN	Earth monitoring	Real	Not mentioned	Journal	[40]
S65	L. Miotti et al.	2020	ANN	Tropospheric delay	Real	Saastamoinen model	Conference	[110]
S67	Y. Liu et al.	2019	NN	Earth monitoring	Real	Not mentioned	Conference	[41]
S68	T. Mortlock et al.	2021	Time delay neural network (TDNN)	LEO satellite	Real	Ground vehicle Doppler measurements extracted from two Orbcomm LEO satellite signals.	Conference	[146]
S69	L. Mengying et al.	2020	SVM	Ionospheric scintillation	Real	Not mentioned	Conference	[99]
S71	A. Lwin et al.	2020	Bayesian Regularization Neural Network (BRNN)	Earth monitoring	Real	Not mentioned	Conference	[35]
S73	S. J. Cho et al.	2019	RNN, SVM, LSTM	Multipath Detection	Real	Not mentioned	Conference	[184]
S74	J. Wang et al.	2019	Deep belief network(DBN)	Earth monitoring	Real	10-fold Cross Validation	Conference	[202]
S75	X. Chu et al.	2020	HMDL	Earth monitoring	Real	Validation dataset	Journal	[42]
S76	G. Zhang et al.	2021	Fully connected NN (FCNNs), LSTM	GNSS Navigation	Real	Validation dataset	Journal	[70]
S77	D. R. Kartchner et al.	2021	LSTM, CNN	GNSS security	Real	Validation accuracy	Conference	[123]
S79	R. Klus et al.	2021	NN	GNSS denied environments	Real	Not mentioned	Conference	[88]
S80	M. Y. Klimenko et al.	2021	NN	Multipath Detection	Real	Not mentioned	Conference	[203]
S82	Y. Liu et al.	2019	NN	Earth monitoring	Real and Simulated	Conventional wind speed retrieval method and other prevailing ML algorithms.	Journal	[43]
S84	A. Hu et al.	2018	ANN	Earth monitoring	Real	Using Constellation Observing System for Meteorology, Ionosphere, and Climate/FC-3 atmPhs (level 1b) data and compared with SVM.	Journal	[115]

TABLE XI
(Continued.)

ID	Authors	Year	ML Technique	Study Application	Data type	Accuracy/Validation	Type	Ref.
S86	A. R. Kazemi et al.	2020	NN	GNSS Interference	Real	Maximum-Likelihood Power-Distortion (PD-ML) detector and Particle Swarm Optimization-Neural Network (PSO-NN).	Conference	[165]
S87	Y. Yang et al.	2017	BP neural network	tropospheric delay	Real	UNB3m and GPT2 models.	Conference	[114]
S88	Q. Yan et al.	2017	NN	Earth monitoring	Real	Mean error, MAE, Standard deviation, and correlation coefficient	Journal	[25]
S89	J. -. De Boer et al.	2009	NN	GNSS Navigation	Real	Mean absolute error (MAE)	Conference	[132]
S90	J. Reynolds et al.	2020	ANN	Earth monitoring	Real	Validation data set with global root mean square (RMS) difference (RMSD)	Journal	[44]
S91	R. Imam et al.	2020	DT	Ionospheric scintillation, multipath	Real	Validation dataset	Conference	[96]
S92	S. Li et al.	2019	RF	Earth monitoring	Real	10-fold cross-validation, and RMSE	Conference	[47]
S93	Y. Su et al.	2021	RF	Earth monitoring	Real	Validation dataset with cross-validation	Journal	[49]
S94	Z. Ã–ZDEMÄ°R et al.	2019	Logistic Regression	GNSS navigation	Real	Mean square error (MSE)	Conference	[204]
S95	W. Ye et al.	2020	Gaussian Process Regression	GNSS Navigation	Real	Validation dataset	Journal	[205]
S98	Y. Luo et al.	2020	SVM	GNSS Interference	Real	5-fold cross-validation	Conference	[162]
S99	Y. Wang et al.	2020	SVM	GNSS signal detection	Real	Using Spire's GNSS-R measurements with cross-validation	Conference	[14]
S100	Q. Yan et al.	2019	SVM	Earth monitoring	Real	Using Delay-Doppler maps (DDMs) datasets	Journal	[28]
S101	L. Cong et al.	2020	SVM	GNSS/INS integration	Real	Simulation using two 180-s GNSS outages, while the observation window size stays at 60 s.	Journal	[133]
S102	L. He et al.	2020	ELM	Satellite clock	Real	Accuracy percentage	Journal	[206]
S103	J Mendez-Astudill et al.	2021	ML regression, SVR	Earth monitoring	Real	Mean absolute error (MEA), the residual sum of squares (MSE), and the coefficient of determination R2	Journal	[50]
S104	M Kiani et al.	2020	MLP, Bayesian NN, radial basis function (RBF) functions, Gaussian processes, k-nearest neighbor, generalized regression neural network, classification and regression trees, and support vector regression.	Earth monitoring	Real	Traditional statistical method called Theta, Mean Absolute Scaled Error (MASE), Mean of Absolute Errors (MAE), and Root of Mean Squared Errors (RMSE)	Journal	[51]
S105	M Kiani et al.	2020	ML	Earth monitoring	Real	Theta - a traditional statistical method , StD and MAE values	Journal	[52]
S106	Liu et al.	2020	RBF-SVM	Ionospheric scintillation	Real	Compared with threshold method, the linear SVM, the threshold voting, and the logistic regression	Conference	[101]
S107	ES Fogarty et al.	2021	SVM	GNSS/ACCL integration	Real	Validation dataset	Journal	[207]
S108	K Maschera et al.	2021	Logistic Regression, SVM, RF, ANN in form of a Multilayer Perceptron (MLP)	GNSS Navigation	Real	Validation dataset	Conference	[185]
S110	M Łoś et al.	2020	RF	Earth monitoring	Real	5-fold cross-validation	Journal	[54]
S111	Y Jia et al.	2019	XGboost	Earth monitoring	Simulated	Using two GNSS-R ground-based campaigns with different soil conditions and compositions, which corresponds to the soil composition of the Simulated data set.	Journal	[34]
S112	M Kiani et al.	2020	KNN, SVR, MLP, BNN, GRNN, GP, CART	Earth monitoring	Real	RMSE, Mean Absolute Scaled Error (MASE), and Mean of Absolute Errors (MAE)	Journal	[53]
S113	Alessandro Neri et al.	2020	LR, Linear Discriminant Analysis (LDA), SVM, KNN, Classification and Regression Trees (CART), Gaussian Naive Bayes (NB)	GNSS navigation	Real	Not mentioned	Conference	[72]
S114	Y Zhu et al.	2020	DT, RF	Earth monitoring	Real	Validation data: Comparing with the sea ice edge (SIE) data from the Special Sensor Microwave Imager Sounder (SSMIS) data. Two sea ice datasets are used to evaluate the performance of the proposed sea ice monitoring approach. The sea ice edge (SIE) data provided by the Ocean and Sea Ice Satellite Application Facility (OSISAF) are used as the reference data.	Journal	[29]
S115	Y Liu et al.	2020	RBF-SVM	Ionospheric scintillation	Real	Validation datatset with cross-validation	Conference	[102]
S116	H Xu et al.	2020	SVM	NLOS Detection	Real	Mean error	Journal	[208]
S117	V Senyurek et al.	2020	ANN, RF, SVM	Earth monitoring	Real	5-fold, site-independent, and year-based cross-validation methods	Journal	[39]
S118	Y Jia et al.	2020	RF, SVM	Earth monitoring	Real	Using GNSS-R system and ground-truth rod sensor used to make measurements before and after rain in bare and smooth fields (Gruliasco/Agliano)	Journal	[36]
S119	Q Yuan et al.	2019	back-propagation neural network (BPNN), generalized regression neural network (GRNN), RF	Earth monitoring	Real	10-fold cross-validation method, Correlation coefficient (R) and the RMSE	Journal	[48]
S121	A Lwin et al.	2020	SVM	Earth monitoring	Real	Cross-validation with RMSE	Conference	[37]
S122	H Liu et al.	2021	Xgboost	Earth monitoring	Real	Validation dataset	Journal	[55]
S124	J Wang et al.	2020	DBN model: composed of a back propagation (BP) layer and several restricted Boltzmann machine (RBM) layers.	Earth monitoring	Real	Correlation coefficient (R), mean absolute error (MAE), root-mean-square error (RMSE), and 10-fold cross-validation	Journal	[31]
S125	N Liu et al.	2021	CNN-LSTM	GNSS/INS integration	Real	Not mentioned	Journal	[130]
S126	HU Kim et al.	2017	LSTM, ANN	Earth monitoring	Real	Validation dataset with RMSE values	Journal	[56]
S127	S Li et al.	2021	LSSVM	tropospheric delay	Real	RMSE value	Journal	[209]
S130	M Kaselimi et al.	2020	CNN	Ionospheric delay	Real	MAE, RMSE value	Conference	[210]
S131	K Kasantikul et al.	2018	ANN, particle filter	Earth monitoring	Real	RMSE value	Journal	[45]
S132	J Hu et al.	2018	BPNN	GNSS navigation	Real	Dual EKF design used to improve the position accuracy, and provide low-noise training and validation samples	Journal	[179]
S133	Dongchan et al.	2019	DNN	Multipath Detection	Real	Standard deviations (StD with TTFF	Conference	[211]
S134	V Otugo et al.	2019	ANN	Ionospheric scintillation	Real	Validation dataset, root-mean-square deviations (RMSDs), and RMSE	Journal	[92]
S136	X Zou et al.	2019	CNN	GNSS/INS integration	Real	Not mentioned	Journal	[212]
S137	O Eroglu et al.	2019	ANN	Earth monitoring	Real	Validation dataset	Journal	[38]
S138	M Kiani et al.	2020	generalized regression NN	GNSS Navigation	Real	Mean Absolute Percentage Error (sMAPE), Standard Deviation (StD), and Mean Absolute Errors (MAE)	Journal	[166]
S139	E Loli Piccolomini et al.	2019	LSTM	GNSS signal detection	Real	Validation dataw with Mean Squared Error (MSE) value and standard deviation (STD)	Journal	[15]
S140	M Moses et al.	2020	NN	Earth monitoring	Real	By comparing the ARITM predictions with ground-based GNSS TEC, the space-based F3/C ionospheric profiles TEC estimates and the existing GIMs. Also using RMSE and Mean Absolute Error (MAE)	Journal	[213]
S141	MR Veronez et al.	2011	ANN	GNSS Navigation	Real	Compared with the Brazilian Geoid Model (MAPGEO2004)	Journal	[73]
S142	Y Liang et al.	2019	BPNN	Earth monitoring	Real	Cross-validation	Journal	[57]
S143	Y Shi et al.	2021	GA-BP	Earth monitoring	Real	Pearson correlation coefficient R, RMSE, bias, and ubRMSE	Journal	[58]
S144	M Zeybek et al.	2014	Linear Regression	Earth monitoring	Real	Standard deviation (StD), and Mean values	Journal	[11]
S145	L Zhao et al.	2019	Regularized Softmax Regression	GNSS/INS integration	Real	RMSE value	Conference	[214]
S146	M Kaselimi et al.	2020	LSTM	Earth monitoring	Real	Mean absolute error (MAE), and RMSE	Journal	[167]
S147	S Miyazawa et al.	2020	LSTM	GNSS Navigation	Real	Using Real data the model with different parameter settings and based on categorical accuracy	Journal	[74]
S148	Shamshiri et al.	2020	GP regression	Tropospheric delay	Real	RMSE value	Journal	[107]
S149	Surisetty et al.	2021	SVR	Earth monitoring	Real	Compared with in-situ depth points. Also using Bias, Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Normalized bias (MNB) and Difference Median (DM)	Journal	[59]
S150	Mutchakayala et al.	2020	Extreme Kernel–Based Learning Machine (KELM)	Earth monitoring	Real	With the error measurements like MAE, MAPE, and RMSE	Journal	[215]
S151	Wojtusiak et al.	2021	ML	GNSS Navigation	Real	10-fold cross-validation, and validation data	Journal	[75]
S152	Osah et al.	2021	DL	tropospheric delay	Real	Mean Bias (MB), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), coefficient of determination (R2), Nash-Sutcliffe coefficient of Efficiency (NSE), and the fraction of prediction within a Factor of Two (FAC2)	Journal	[216]

TABLE XI
(Continued.)

ID	Authors	Year	ML Technique	Study Application	Data type	Accuracy/Validation	Type	Ref.
S153	Salar et al.	2021	LSTM	Ionospheric TEC content	Real	Root mean square error (RMSE), and mean absolute error (MAE)	Journal	[217]
S155	Xia et al.	2021	SVM	Earth monitoring	Real	RMSE, and relative error (RE)	Journal	[168]
S156	Li et al.	2021	ANN	Earth monitoring	Real	RMSE value	Journal	[46]
S157	Mohammed et al.	2021	ANN	tropospheric delay	Real	Cross-validation, Mean Square Error (MSE) or Sum of Square Error (SSE), and RMSE	Journal	[111]
S158	Okoh et al.	2016	NN	Earth monitoring	Real	Validation dataset with RMSE	Journal	[169]
S159	Rafatnia et al.	2018	Recurrent wavelet neural network (RWNN)	GNSS/INS integration	Real	Mean value and standard deviation: Highly accurate Vitans navigation system is used to provide reference values for evaluatio	Journal	[172]
S160	Sahu et al.	2021	NN	Ionospheric TEC content	Real	Comparison of hourly values of NN TEC with GPS TEC using relative error.	Journal	[170]
S161	Sivavaraprasad et al.	2020	NN	Ionospheric TEC content	Real	RMSE value	Journal	[171]
S162	Ferreira et al.	2017	NN	Ionospheric TEC content	Real	Comparison with Global Ionospheric Maps provided by CODE is performed using RMSE.	Journal	[173]
S163	P. Pavlovčič Prešeren et al.	2013	Wavelet Neural Network (WNN)	GNSS ephemerides distribution and short-time prediction	Real	Minimum value of differences min; Maximum value of differences max; Mean value of differences; Normalized root mean square error (NRMSE): $NRMSE = RMSE/(max - min)$	Journal	[83]
S164	Rahimi et al.	2018	ANN	Earth monitoring	Real	Using six IGS stations. The collected ZTD products were used to compare and validate the accuracy of the estimated ZTD by GNSS analysis MIT (GAMIT)	Journal	[60]
S165	Kemal Tütüncü et al.	2021	ELM	GNSS/leveling geoid determination	Real	Checkpoints serving as a kind of validation	Journal	[239]
S166	R.E. Guinness et al.	2013	SVM, ANN, LR, BN, DT, NB, IBk, LWL and KStar	Situation/Context Awareness	Real	10-fold cross-validation	Journal	[86]
S167	N. Yamaga et al.	2019	NN	Earth monitoring	Real	Not mentioned	Journal	[160]
S168	Q. Yuan et al.	2019	ANN, RF	Earth monitoring	Real	10-fold cross-validation	Journal	[186]
S169	L. Li et al.	2020	GRNN	tropospheric delay	Real	A popular R-ratio is used as the validation method and a threshold value is set to 2.5	Journal	[218]
S170	B. Huang et al.	2021	LSTM	satellite clock	Real	Not mentioned	Journal	[174]
S171	M. Kim et al.	2016	NN	Ionospheric delay	Real	Not mentioned	Journal	[105]
S172	B. Zhang et al.	2021	GRNN	Earth monitoring	Real	10-fold cross-validation (CV)	Journal	[61]
S173	C. Herbert et al.	2021	Regression NN	Earth monitoring	Real	Validation dataset	Journal	[30]
S174	Taro Suzuki et al.	2021	SVM, NN	Multipath Detection	Real	Cross-validation	Journal	[17]
S175	Marco Mendonça et al.	2020	SVM	GNSS/INS integration	Real	Not mentioned	book-chapter	[240]
S179	Selbesoglu et al.	2019	ANN	tropospheric delay	Real	Cross validation	Journal	[112]
S180	Mohamad Orabi et al.	2020	NN	Multipath Detection	Simulated	RMSE value	Conference	[163]
S181	Yu Jiao et al.	2017	Linear SVM, medium Gaussian kernel SVM with a kernel scale of 9.1	Ionospheric scintillation	Real	5-fold cross-validation, and ROC curve	Conference	[90]
S183	Hany Ragabet et al.	2020	MLPNN	GNSS/INS integration	Real	RMSE value	Conference	[219]
S185	J. Rossouw van der Merwe et al.	2020	Logistic regression (LR), K-nearest neighbors (KNN), naive Bayes (NB), DT, SVM	GNSS spoofing attack	Real and Simulated	F1-score	Conference	[124]
S186	Yung-Cheng et al.	2006	NN	GNSS/INS/Odometer integration	Real	RMS value (degree)	Conference	[138]
S187	Hamad Yousif and Ahmed El-Rabbany	2008	NN	GNSS Navigation	Real	Corresponding precise rapid ephemeris before any improvement is applied	Conference	[164]
S188	Jianguo Jack Wang et al.	2007	NN	GNSS/INS integration	Real	Mean RMS	Conference	[139]
S190	T. Désert et al	2015	NN	Ionosphere modelling	Real	EGNOS	Conference	[103]
S191	Li He et al	2016	ANN	GNSS signal detection	Real	Not mentioned	Conference	[20]
S192	W. Vigneau et al.	2006	NN	LEO satellites	simulalted	Mean square error (MSE)	Conference	[147]
S194	Pere Ramos-Bosch et al.	2007	NN	LEO satellites	Real	3D RMS	Journal	[148]
S195	Chengquan Xu et al.	2008	NN	Earth monitoring	Real	Not mentioned	Conference	[175]
S196	Heekwon No et al	2021	Quantile Regression	Multipath Detection	Real	By an integrity assessment of the experimental data.	Conference	[220]
S197	Qiming Zhong et al	2021	Bayesian Filter	GNSS Navigation	Real	Not mentioned	Conference	[221]
S198	Chengjun Guo et al	2021	CNN	GNSS Interference	Simulated	Validation dataset	Conference	[21]
S199	Nesrine HARBAOUI et al	2021	DCNN	GNSS Navigation	Real	Maximum error, and RMSE	Conference	[222]
S200	Sharbel E. Kozhaya et al	2021	TDNN which are a type of Feed Forward Neural Network (FFNN), LSTM a type of Recurrent Neural Network (RNN)	GNSS Navigation	Real	RMSE value	Conference	[149]
S201	Adyasha Mohanty et al	2021	CNN	GNSS/INS integration	Real	RMSE value	Conference	[134]
S202	Kahn-Bao Wu et al	2021	SVM	Oscillator Anomaly detection	Real	Grid-search-based cross-validation	Conference	[223]
S203	Ashwin V. Kanhere et al	2021	DNN	GNSS Navigation	Real and Simulated	Comparison of the absolute localization error to a weighted least squares (WLS) baseline with positioning error	Conference	[76]
S204	Akpojoto Siemuri et al	2021	LR, BR, NN	GNSS Navigation	Real	Validation dataset	Conference	[77]
S207	Gianluca Caparra et al	2021	NN	GNSS Navigation	Real	Percentile, cumulative distribution function (CDF)	Conference	[78]
S208	Nesreen I. Ziedan	2021	Self-Organizing Map (SOM) neural network	GNSS Navigation	Simulated	RMS error	Conference	[79]
S209	Annabel R. Gomez et al	2021	Ridge Regression, LSTM, Classification Neural Network (CNN), Autoencoder Classification Neural Network (ACNN), LSTM Autoencoder Classification Neural Network (LACNN)	Ionospheric scintillation	Real	Validation dataset	Conference	[91]
S210	Lei Liu et al	2021	custom- designed loss function Lc (convLSTM-Lc)	Ionospheric scintillation	Real	Validation dataset	Conference	[224]
S211	Yunxiang Liu et al	2021	spatiotemporal deep learning (STDL) LSTM network	Ionospheric scintillation	Real	ROC curve, and AUC	Conference	[225]
S213	Quoc-Huy Phan et al	2013	SVR	Multipath Detection	Real	Standard deviations	Journal	[226]
S214	Azami Hamed et al	2013	NN (including Levenberg–Marquardt (LM), modified LM, and resilient BP (RBP), scaled conjugate gradient, one-step secant (OSS) and quasi-Newton methods), PCA	GNSS GDOP classification	Real	Classification rate	Journal	[236]
S215	Nadali Zarei	2014	PSO, NPSO, GA and ICA, to train an NN.	GNSS GDOP classification	Real	Classification rates	Journal	[237]
S216	Noureldin et al	2010	NN	GNSS/INS integration	Real	Simulation using 100s GPS outages with RMSE values	Journal	[140]
S217	Yu Jia et al	2017	SVM	Ionospheric scintillation	Real	A 25% hold-out validation is configured to evaluate the performance of the training with ROC curve	Journal	[100]
S218	E. S. Abdolkarimi et al	2018	ELM	GNSS/INS integration	Real	RMSE value	Journal	[131]
S220	Aly M.El-naggar	2013	ANN	Ionospheric scintillation	Real	Not mentioned	Journal	[93]
S221	Yiming Quan et al	2018	CNN	Multipath Detection	Simulated and Real	RMSE value	Journal	[227]
S222	Habarulema et al	2009	NN	Earth monitoring	Real	RMSE value	Journal	[176]

TABLE XI
(Continued.)

ID	Authors	Year	ML Technique	Study Application	Data type	Accuracy/Validation	Type	Ref.
S223	John Bosco Habarulema et al	2009	FFNN	Earth monitoring	Real	RMSE value	Journal	[228]
S224	R. Sharaf et al	2005	RBF-ANN	GNSS/INS integration	Real	MSE value	Journal	[136]
S225	Christos Pikridas et al	2010	ANN	tropospheric delay	Real	RMSE value	Journal	[113]
S226	Pedro Benevides et al	2019	NN	Earth monitoring	Real	RMSE value	Journal	[62]
S227	G. Panice et al	2017	SVM	GNSS security	Real	False Positive Rate (FPR); True Positive Rate (TPR); and Accuracy	Conference	[126]
S228	Rui Sun et al	2020	GBDT	GNSS Navigation	Real	RMSE value	Journal	[80]
S229	Mosavi et al	2017	RBF-NN	GNSS Navigation	Real	Validation dataset, with RMSE	Journal	[81]
S231	Li Jing et al	2016	ensemble learning algorithm (LS-Boost or Bagging)	GNSS/INS integration	Real	Position error	Journal	[187]
S232	Yimin Zhou et al	2017	BPNN	GNSS/INS integration	Real	MSE value	Conference	[127]
S233	Guangcai Wang et al	2019	BPNN	GNSS/INS integration	Real	Mean value of absolute errors (MAE)	Journal	[128]
S234	Fangni Lei et al	2020	ML	Earth monitoring	Real	Correlation coefficient (R) and unbiased root-mean-square-difference (ubRMSD)	Conference	[229]
S235	Zhengxie Zhang et al	2019	SVM	Ionospheric delay	Real	RMSE value	Journal	[108]
S236	Li et al.	2016	Twin SVM	GNSS Interference	Simulated	Not mentioned	Journal	[121]
S237	Wu et al	2017	CNN	GNSS Interference	Real	Accuracy percentage	Journal	[?]
S238	Savas et al	2019	K-means clustering	Multipath Detection	Real and Simulated	Standard deviation	Conference	[177]
S239	Liu et al	2021	convLSTM	Ionospheric irregularities	Real	Not mentioned	Journal	[230]
S241	David et al	2016	Neural Network	Ionospheric Scintillation	Real	Validation dataset	Journal	[231]
S242	Huang et al	2014	RBF-NN	Ionospheric Scintillation	Real	Not mentioned	Journal	[232]
S243	Yilmaz et al	2009	NN	Earth monitoring	Real	Sum-squared error (SSE)	Journal	[188]
S244	Habarulema et al	2007	NN	Earth monitoring	Real	RMSE value	Journal	[233]
S245	Sabzehee et al.	2018	ANN	Earth monitoring	Real	RMSE, and R2 values	Journal	[178]
S246	Leandro et al	2007	NN	Earth monitoring	Real	Not mentioned	Journal	[234]
S247	Wang et al	2017	WNN	satellite clock	Simulated	Not mentioned	Journal	[235]
S248	Lyu et al	2020	SVM	GNSS Navigation	Real	Comparison with the built-in RTK solutions of a dual-frequency u-blox F9P, and multi-frequency Trimble BD982.	Journal	[19]
S249	Shafiee et al	2017	Multi-layer NN	GNSS security	Real and Simulated	K-fold cross-validation	Journal	[125]

regarding the type of data used for evaluating the model, the studies made use of either simulated data, real data or semisimulated data when evaluating the model.

In a multi-GNSS environment, due to the availability of large number of satellite signals and capable hardware, the issue of optimum geometry (i.e., dilution of precision, DOP) can be easily taken care of by use of multiple satellites giving the lowest DOP. However, error situations, such as multipath, can benefit from optimal selection of the satellites. ML plays a major role in this area of research, as this review indicates.

This review provides recommendations as well as guidelines for researchers and practitioners. For researchers, we recommend that they conduct more empirical studies making use of the rarely-used ML algorithms, keeping in mind the GNSS application context, in order to further strengthen the evidence about their performance. For future work, we encourage exploring the possibilities of using unapplied ML techniques to new or already studied GNSS use cases. Additionally, more studies should be performed using the combination of non-ML and ML algorithms to further strengthen the evidence about their performance when utilized in GNSS. Furthermore, it would be important to use the same data. Section III-D shows that there are plenty of different datasets used by different researchers. Therefore, we suggest the development of a common test bench to study the utilization of ML algorithm in GNSS.

Appendix A

See Table X.

Appendix B

See Table XI.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their insightful comments.

REFERENCES

[1] M. Lehtinen, A. Happonen, and J. Ikonen, "Accuracy and time to first fix using consumer-grade GPS receivers," in *Proc. 16th Int. Conf. Softw., Telecommun. Comput. Netw.*, 2008, pp. 334–340, doi: 10.1109/SOFTCOM.2008.4669506.

[2] K. D. Elliott and H. Christopher J, *Understanding GPS/GNSS: Principles and Applications*. Norwood, MA, USA: Artech House, 2017.

[3] Q. Yiming, "A new machine learning based method for multi-GNSS data quality assurance and multipath detection," Ph.D. thesis, Fac. Sci. Eng., Dept. Civil Eng., Univ. Nottingham, Nottingham, U.K., 2017. [Online]. Available: http://eprints.nottingham.ac.uk/39748/

[4] Internet of business, "Google's DeepMind AI is learning to navigate cities without a map," 2022, Accessed: Jul. 4, 2022. [Online]. Available: https://internetofbusiness.com/deepmind-ai-learning-to-navigate-without-map/

[5] M. Joshi, "Google uses deep learning with Street View to update its maps," 2017, Accessed: Jul. 4, 2022. [Online]. Available: https://www.geospatialworld.net/blogs/google-uses-deep-learning-with-street-view-to-update-its-maps/

[6] T. Cozzens, "Apple applies for machine learning GNSS device," 2020, , Accessed: Jul. 4, 2022. [Online]. Available: https://www.gpsworld.com/apple-applies-for-machine-learning-gnss-device/

[7] A. Siemuri, H. Kuusniemi, M. S. Elmusrati, P. Välisuo, and A. Shamsuzzoha, "Machine learning utilization in GNSS—Use cases, challenges and future applications," in *Proc. Int. Conf. Localization (GNSS)*, 2021, pp. 1–6, doi: 10.1109/ICL-GNSS51451.2021.9452295.

[8] B. Kitchenham, "Procedures for performing systematic reviews," Keele Univ., Keele, UK, vol. 33, 2004. [Online]. Available: https://www.inf.ufsc.br/aldo.vw/kitchenham.pdf

[9] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Appl. Soft Comput.*, vol. 27, pp. 504–518, 2015. [Online]. Available: https://www.researchgate.net/publication/228756057_Procedures_for_Performing_Systematic_Reviews

[10] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584911001832

[11] M. Zeybek, I. Sanlıoglu, and A. Genc, "Landslide monitoring with (gnss) measurements and prediction with linear regression model: A case study Taşkent (Konya/Turkey) landslide," 2014.

[12] P. Borhani-Darian and P. Closas, "Deep neural network approach to GNSS signal acquisition," in *Proc. IEEE/ION Position, Location Navigation Symp.*, 2020, pp. 1214–1223, doi: 10.1109/PLANS46316.2020.9110205.

[13] R. Sun, G. Wang, W. Zhang, L.-T. Hsu, and W. Y. Ochieng, "A gradient boosting decision tree based GPS signal reception classification algorithm," *Appl. Soft Comput.*, vol. 86, 2020, Art. no. 105942. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494619307239

[14] Y. Wang, Y. Liu, C. Roesler, and Y. J. Morton, "Detection of coherent GNSS-R measurements using a support vector machine," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2020, pp. 6210–6213, doi: 10.1109/IGARSS39084.2020.9323138.

[15] E. L. Piccolomini, S. Gandolfi, L. Poluzzi, L. Tavasci, P. Cascarano, and A. Pascucci, "Recurrent neural networks applied to GNSS time series for denoising and prediction," in *Proc. 26th Int. Symp. Temporal Representation Reason., ser. Leibniz Int. Proc. Inform. (LIPIcs),*, J. Gamper, S. Pinchinat, and G. Sciavicco, Eds. Dagstuhl, Germany: Schloss Dagstuhl.Leibniz-Zentrum fuer Informatik, vol. 147, 2019, pp. 10:1–10:12. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2019/11368

[16] B. Guermah, H. E. Ghazi, T. Sadiki, and H. Guermah, "A robust GNSS LoS/multipath signal classifier based on the fusion of information and machine learning for intelligent transportation systems," in *Proc. IEEE Int. Conf. Technol. Manage., Operations Decis.*, 2018, pp. 94–100, doi: 10.1109/ITMC.2018.8691272.

[17] T. Suzuki and Y. Amano, "NLoS multipath classification of GNSS signal correlation output using machine learning," *Sensors*, vol. 21, no. 7, 2021, Art. no. 2503. [Online]. Available: https://www.mdpi.com/1424-8220/21/7/2503

[18] Z. Wu, Y. Zhao, Z. Yin, and H. Luo, "Jamming signals classification using convolutional neural network," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol.*, 2017, pp. 062–067, doi: 10.1109/ISSPIT.2017.8388320.

[19] Z. Lyu and Y. Gao, "An SVM based weight scheme for improving kinematic GNSS positioning accuracy with low-cost GNSS receiver in urban environments," *Sensors*, vol. 20, no. 24, 2020, Art. no. 7265. [Online]. Available: https://www.mdpi.com/1424-8220/20/24/7265

[20] L. He, H. Li, W. Li, and M. Lu, "Neural network based C/N0 abnormity detection method for GPS anti-spoofing," in *Proc. Int. Tech. Meeting Inst. Navigation*, 2016, pp. 716–725, doi: 10.33012/2016.13454.

[21] C. Guo and W. Tu, "GNSS interference signal recognition based on deep learning and fusion time-frequency features," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 855–863, doi: 10.33012/2021.17937.

[22] D. Brum et al., "A proposed earthquake warning system based on ionospheric anomalies derived from GNSS measurements and artificial neural networks," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2019, pp. 9295–9298, doi: 10.1109/IGARSS.2019.8900197.

[23] D. Brum et al., "A proposed earthquake warning system based on ionospheric anomalies derived from GNSS measurements and artificial neural networks," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2019, pp. 9295–9298, doi: 10.1109/IGARSS.2019.8900197.

[24] M. Alshaye, F. Alawwad, and I. Elshafiey, "Hurricane tracking using multi-GNSS-R and deep learning," in *Proc. 3rd Int. Conf. Comput. Appl. Inf. Secur.*, 2020, pp. 1–4, doi: 10.1109/ICCAIS48893.2020.9096717.

[25] Q. Yan, W. Huang, and C. Moloney, "Neural networks based sea ice detection and concentration retrieval from GNSS-R delay-Doppler maps," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 8, pp. 3789–3798, Aug. 2017, doi: 10.1109/JSTARS.2017.2689009.

[26] Q. Yan and W. Huang, "Sea ice sensing from GNSS-R data using convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 10, pp. 1510–1514, Oct. 2018, doi: 10.1109/LGRS.2018.2852143.

[27] Q. Yan et al., "Sea ice thickness estimation from TechDemoSat-1 and soil moisture ocean salinity data using machine learning methods," in *Proc. Glob. Oceans: Singapore–U. S. Gulf Coast*, 2020, pp. 1–5, doi: 10.1109/IEEECONF38699.2020.9388974.

[28] Q. Yan and W. Huang, "Detecting sea ice from TechDemoSat-1 data using support vector machines with feature selection," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 5, pp. 1409–1416, May 2019, doi: 10.1109/JSTARS.2019.2907008.

[29] Y. Zhu et al., "Machine learning-aided sea ice monitoring using feature sequences extracted from spaceborne GNSS-reflectometry data," *Remote Sens.*, vol. 12, no. 22, 2020, Art. no. 3751. [Online]. Available: https://www.mdpi.com/2072-4292/12/22/3751

[30] C. Herbert, J. F. Munoz-Martin, D. Llaveria, M. Pablos, and A. Camps, "Sea ice thickness estimation based on regression neural networks using L-band microwave radiometry data from the FSSCat mission," *Remote Sens.*, vol. 13, no. 7, 2021, Art. no. 1366. [Online]. Available: https://www.mdpi.com/2072-4292/13/7/1366

[31] J. Wang et al., "Estimating snow depth by combining satellite data and ground-based observations over Alaska: A deep learning approach," *J. Hydrol.*, vol. 585, 2020, Art. no. 124828. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022169420302882

[32] Y. Jia, Y. Pei, and W. Li, "A machine learning aided method for GNSS-R permittivity retrieval vim analysis," in *Proc. IEEE Int. Conf. Signal, Inf. Data Process.*, 2019, pp. 1–5, doi: 10.1109/ICSIDP47821.2019.9173228.

[33] Y. Jia et al., "Temporal-spatial soil moisture estimation from CYGNSS using machine learning regression with a preclassification approach," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 4879–4893, 2021, doi: 10.1109/JSTARS.2021.3076470.

[34] Y. Jia et al., "GNSS-R soil moisture retrieval based on a XGboost machine learning aided method: Performance and validation," *Remote Sens.*, vol. 11, no. 14, 2019, Art. no. 1655. [Online]. Available: https://www.mdpi.com/2072-4292/11/14/1655

[35] A. Lwin, D. Yang, and X. Hong, "Leveraging Bayesian deep learning for spaceborne GNSS-R retrieval on global soil moisture," in *Proc. Int. Conf. Artif. Intell. Inf. Commun.*, 2020, pp. 352–355, doi: 10.1109/ICAIIC48513.2020.9065053.

[36] Y. Jia, S. Jin, P. Savi, Q. Yan, and W. Li, "Modeling and theoretical analysis of GNSS-R soil moisture retrieval based on the random forest and support vector machine learning approach," *Remote Sens.*, vol. 12, no. 22, 2020, Art. no. 3679. [Online]. Available: https://www.mdpi.com/2072-4292/12/22/3679

[37] A. Lwin, D. Yang, X. Hong, S. C. Shamsabadi, and W. A. Ahmed, "Spaceborne GNSS-R retrieving on global soil moisture approached by support vector machine learning," *Int. Arch. Photogrammetry, Remote Sens. Spatial Inf. Sci.*,, vol. 43B3, pp. 605–610, 2020.

[38] O. Eroglu, M. Kurum, D. Boyd, and A. C. Gurbuz, "High spatio-temporal resolution CYGNSS soil moisture estimates using artificial neural networks," *Remote Sens.*, vol. 11, no. 19, 2019, Art. no. 2272. [Online]. Available: https://www.mdpi.com/2072-4292/11/19/2272

[39] V. Senyurek, F. Lei, D. Boyd, M. Kurum, A. C. Gurbuz, and R. Moorhead, "Machine learning-based CYGNSS soil moisture estimates over ISMN sites in conus," *Remote Sens.*, vol. 12, no. 7, 2020, Art. no. 1168. [Online]. Available: https://www.mdpi.com/2072-4292/12/7/1168

[40] M. Asgarimehr, I. Zhelavskaya, G. Foti, S. Reich, and J. Wickert, "A GNSS-R geophysical model function: Machine learning for wind speed retrievals," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 8, pp. 1333–1337, Aug. 2020, doi: 10.1109/LGRS.2019.2948566.

[41] Y. Liu, J. Wang, I. Collett, and Y. J. Morton, "A machine learning framework for real data GNSS-R wind speed retrieval," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2019, pp. 8707–8710, doi: 10.1109/IGARSS.2019.8899792.

[42] X. Chu et al., "Multimodal deep learning for heterogeneous GNSS-R data fusion and ocean wind speed retrieval," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 5971–5981, 2020, doi: 10.1109/JSTARS.2020.3010879.

[43] Y. Liu, I. Collett, and Y. J. Morton, "Application of neural network to GNSS-R wind speed retrieval," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 12, pp. 9756–9766, Dec. 2019, doi: 10.1109/TGRS.2019.2929002.

[44] J. Reynolds, M. P. Clarizia, and E. Santi, "Wind speed estimation from CYGNSS using artificial neural networks," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 708–716, 2020, doi: 10.1109/JSTARS.2020.2968156.

[45] K. Kasantikul, D. Yang, Q. Wang, and A. Lwin, "A novel wind speed estimation based on the integration of an artificial neural network and a particle filter using BeiDou geo reflectometry," *Sensors*, vol. 18, no. 10, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/10/3350

[46] X. Li et al., "Analysis of coastal wind speed retrieval from CYGNSS mission using artificial neural network," *Remote Sens. Environ.*, vol. 260, 2021, Art. no. 112454. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425721001723

[47] S. Li, Q. Yuan, L. Yue, T. Li, H. Shen, and L. Zhang, "Downscaling GNSS-R based vegetation water content product using random forest model," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2019, pp. 6720–6723, doi: 10.1109/IGARSS.2019.8900472.

[48] Q. Yuan, S. Li, L. Yue, T. Li, H. Shen, and L. Zhang, "Monitoring the variation of vegetation water content with machine learning methods: Point–surface fusion of MODIS products and GNSS-IR observations," *Remote Sens.*, vol. 11, no. 12, 2019, Art. no. 1440. [Online]. Available: https://www.mdpi.com/2072-4292/11/12/1440

[49] Y. Su, K. Weng, C. Lin, and Z. Zheng, "An improved random forest model for the prediction of dam displacement," *IEEE Access*, vol. 9, pp. 9142–9153, 2021, doi: 10.1109/ACCESS.2021.3049578.

[50] J. Mendez-Astudillo and M. Mendez-Astudillo, "A machine learning approach to monitoring the UHI from GNSS data," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–11, 2021, Art. no. 5800911, doi: 10.1109/TGRS.2021.3091949.

[51] M. Kiani, "A precise machine learning aided algorithm for land subsidence or upheave prediction from GNSS time series," Cornell Univ., 2020, *arXiv:2006.03772*.

[52] M. K. Shahvandi, "A specifically designed machine learning algorithm for GNSS position time series prediction and its applications in outlier and anomaly detection and earthquake prediction," Cornell Univ., 2020, *arXiv:2006.09067*.

[53] M. Kiani, "Lateral land movement prediction from GNSS position time series in a machine learning aided algorithm," Cornell Univ., 2020, *arXiv:2006.07891*.

[54] M. Łoś, K. Smolak, G. Guerova, and W. Rohm, "GNSS-based machine learning storm nowcasting," *Remote Sens.*, vol. 12, no. 16, 2020, Art. no. 2536. [Online]. Available: https://www.mdpi.com/2072-4292/12/16/2536

[55] H. Liu, L. Yang, and L. Li, "Analyzing the impact of climate factors on GNSS-derived displacements by combining the extended Helmert transformation and XGboost machine learning algorithm," *Hindawi, J. Sensors*, vol. 2021, 2021, Art. no. 9926442, doi: 10.1155/2021/9926442.

[56] H.-U. Kim and T.-S. Bae, "Preliminary study of deep learning-based precipitation," *J. Korean Soc. Surveying, Geodesy, Photogrammetry Cartography*, vol. 35, no. 5, pp. 423–430, 2017, doi: 10.7848/ksgpc.2017.35.5.423.

[57] Y. Ji Liang, C. Ren, H. Yu Wang, Y. bang Huang, and Z. tian Zheng, "Research on soil moisture inversion method based on GA-BP neural network model," *Int. J. Remote Sens.*, vol. 40, no. 5/6, pp. 2087–2103, 2019.

[58] Y. Shi, C. Ren, Z. Yan, and J. Lai, "High spatial-temporal resolution estimation of ground-based global navigation satellite system interferometric reflectometry (GNSS-IR) soil moisture using the genetic algorithm back propagation (GA-BP) neural network," *Int. J. Geo- Inf.*, vol. 10, no. 9, 2021, Art. no. 623. [Online]. Available: https://www.mdpi.com/2220-9964/10/9/623

[59] V. A. K. Surisetty, C. Venkateswarlu, B. Gireesh, K. Prasad, and R. Sharma, "On improved nearshore bathymetry estimates from satellites using ensemble and machine learning approaches," *Adv. Space Res.*, vol. 68, no. 8, pp. 3342–3364, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117721005202

[60] Z. Rahimi, H. Z. Mohd Shafri, and M. Norman, "A GNSS-based weather forecasting approach using nonlinear auto regressive approach with exogenous input (NARX)," *J. Atmos. Sol.- Terr. Phys.*, vol. 178, pp. 74–84, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364682618302220

[61] B. Zhang and Y. Yao, "Precipitable water vapor fusion based on a generalized regression neural network," *J. Geodesy*, vol. 95, no. 3, 2021, Art. no. 36, doi: 10.1007/s00190-021-01482-z.

[62] P. Benevides, J. Catalao, and G. Nico, "Neural network approach to forecast hourly intense rainfall using GNSS precipitable water vapor and meteorological sensors," *Remote Sens.*, vol. 11, no. 8, 2019, Art. no. 966. [Online]. Available: https://www.mdpi.com/2072-4292/11/8/966

[63] A. Kuratomi, "GNSS position error estimated by machine learning techniques with environmental information input," Masters thesis, , KTH Roy. Inst. Technol., Sch. Ind. Eng. Manage., 2019. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1362035/FULLTEXT01.p

[64] C. Ma, J. Yang, J. Chen, and Y. Tang, "Indoor and outdoor positioning system based on navigation signal simulator and pseudolites," *Adv. Space Res.*, vol. 62, no. 9, pp. 2509–2517, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117718305684

[65] P. Dabove and V. Di Pietra, "Towards high accuracy GNSS real-time positioning with smartphones," *Adv. Space Res.*, vol. 63, no. 1, pp. 94–102, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117718306537

[66] A. Uzun, F. A. Ghani, H. Yenigun, and I. Tekin, "A novel GNSS repeater architecture for indoor positioning systems in ISM band," in *Proc. IEEE Int. Symp. Antennas Propag. North Amer. Radio Sci. Meeting*, 2020, pp. 1631–1632, doi: 10.1109/IEEECONF35879.2020.9329653.

[67] Y. Sun, J. Wang, and J. Chen, "Indoor precise point positioning with pseudolites using estimated time biases iPPP and iPPP-RTK," *GPS Solutions*, vol. 25, no. 2, Jan. 2021, Art. no. 41, doi: 10.1007/s10291-020-01064-0.

[68] D. Bhatt, P. Aggarwal, V. Devabhaktuni, and P. Bhattacharya, "Seamless navigation via Dempster Shafer theory augmented by support vector machines," *Proc. 25th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2012, pp. 98–104. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=10224

[69] H.-U. Kim and T.-S. Bae, "Deep learning-based GNSS network-based real-time kinematic improvement for autonomous ground vehicle navigation," *J. Sensors*, vol. 2019, 2019, Art. no. 3737265, doi: 10.1155/2019/3737265.

[70] G. Zhang, P. Xu, H. Xu, and L.-T. Hsu, "Prediction on the urban GNSS measurement uncertainty based on deep learning networks with long short-term memory," *IEEE Sensors J.*, vol. 21, no. 18, pp. 20563–20577, Sep. 2021, doi: 10.1109/JSEN.2021.3098006.

[71] Z. Zhou, Y. Li, C. Fu, and C. Rizos, "Least-squares support vector machine-based Kalman filtering for GNSS navigation with dynamic model real-time correction," *IET Radar, Sonar Navigation*, vol. 11, no. 3, pp. 528–538, 2017. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-rsn.2016.0422

[72] A. Neri, A. Ruggeri, A. Vennarini, and A. Coluccia, "Machine learning for GNSS performance analysis and environment characterization in rail domain," in *Proc. 33rd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2020, pp. 3561–3566, doi: 10.33012/2020.17673.

[73] M. R. Veronez, S. Florêncio de Souza, M. T. Matsuoka, A. Reinhardt, and R. Macedônio da Silva, "Regional mapping of the geoid using GNSS (GPS) measurements and an artificial neural network," *Remote Sens.*, vol. 3, no. 4, pp. 668–683, 2011. [Online]. Available: https://www.mdpi.com/2072-4292/3/4/668

[74] S. Miyazawa, X. Song, R. Jiang, Z. Fan, R. Shibasaki, and T. Sato, "City-scale human mobility prediction model by integrating GNSS trajectories and SNS data using long short-term memory," *ISPRS Ann. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 5, no. 4, pp. 87–94, 2020, doi: 10.5194/isprs-annals-V-4-2020-87-2020.

[75] J. Wojtusiak and R. M. Nia, "Location prediction using GPS trackers: Can machine learning help locate the missing people with dementia?," *Internet Things*, vol. 13, 2021, Art. no. 100035. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660518300787

[76] A. V. Kanhere, S. Gupta, A. Shetty, and G. Gao, "Improving GNSS positioning using neural network-based corrections," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 3068–3080, doi: 10.33012/2021.17999.

[77] A. Siemuri, K. Selvan, H. Kuusniemi, P. Välisuo, and M. S. Elmusrati, "Improving precision GNSS positioning and navigation accuracy on smartphones using machine learning," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 3081–3093, doi: 10.33012/2021.18004.

[78] G. Caparra, P. Zoccarato, and F. Melman, "Machine learning correction for improved PVT accuracy," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 3392–3401, doi: 10.33012/2021.17974.

[79] N. I. Ziedan, "Optimized position estimation in multipath environments using machine learning," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 3437–3451, doi: 10.33012/2021.17880.

[80] R. Sun et al., "Improving GPS code phase positioning accuracy in urban environments using machine learning," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 7065–7078, Apr. 2021, doi: 10.1109/JIOT.2020.3037074.

[81] M. Mosavi and A. Rashidinia, "Improving accuracy of DGPS correction prediction in position domain using radial basis function neural network trained by PSO algorithm," *Iranian J. Elect. Electron. Eng.*, vol. 13, no. 3, pp. 219–227, 2017. [Online]. Available: http://www.iust.ac.ir/ijeee/article-1-1070-en.pdf

[82] J. Chang et al., "Combining genetic algorithms and machine learning for exploring the navigation satellite constellation design tradespace," in *Proc. 33rd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2020, pp. 1209–1228, doi: 10.33012/2020.17616.

[83] P. Prešeren and B. Stopar, "Wavelet neural network employment for continuous GNSS orbit function construction: Application for the assisted-GNSS principle," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2526–2536, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494612005182

[84] H. S. Maghdid, I. A. Lami, K. Z. Ghafoor, and J. Lloret, "Seamless outdoors-indoors localization solutions on smartphones: Implementation and challenges," *ACM Comput. Surv.*, vol. 48, no. 4, 2016, Art. no. 53, doi: 10.1145/2871166.

[85] Y. Xia et al., "Recurrent neural network based scenario recognition with multi-constellation GNSS measurements on a smartphone," *Measurement*, vol. 153, 2020, Art. no. 107420. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224119312874

[86] R. E. Guinness, "Beyond where to how: A machine learning approach for sensing mobility contexts using smartphone sensors," *Sensors*, vol. 15, no. 5, pp. 9962–9985, 2015. [Online]. Available: https://www.mdpi.com/1424-8220/15/5/9962

[87] Q. Liu, Z. Huang, and J. Wang, "Indoor non-line-of-sight and multipath detection using deep learning approach," *GPS Solutions*, vol. 23, no. 3, pp. 1–14, 2019, doi: 10.1007/s10291-019-0869-4.

[88] R. Klus, J. Talvitie, and M. Valkama, "Neural network fingerprinting and GNSS data fusion for improved localization in 5G," in *Proc. Int. Conf. Localization GNSS*, 2021, pp. 1–6, doi: 10.1109/ICL-GNSS51451.2021.9452245.

[89] Y. Liu, Y. J. Morton, and Y. Jiao, "Application of machine learning to the characterization of GPS l1 ionospheric amplitude scintillation," in *Proc. IEEE/ION Position, Location Navigation Symp.*, 2018, pp. 1159–1166, doi: 10.1109/PLANS.2018.8373500.

[90] Y. Jiao, J. Hall, and Y. J. Morton, "Automatic GPS phase scintillation detector using a machine learning algorithm," in *Proc. Int. Tech. Meeting Inst. Navigation*, 2017, pp. 1160–1172, doi: 10.33012/2017.14903.

[91] A. R. Gomez and X. Pi, "Applying machine learning to predict Alaskan ionospheric irregularities," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 3848–3858, doi: 10.33012/2021.18032.

[92] V. Otugo et al., "Estimation of ionospheric critical plasma frequencies from GNSS-TEC measurements using artificial neural networks," *Advancing Earth Space Sci., Space Weather*, vol. 17, pp. 1329–1340, 2019, doi: 10.1029/2019SW002257.

[93] A. M. El-naggar, "Artificial neural network as a model for ionospheric TEC map to serve the single frequency receiver," *Alexandria Eng. J.*, vol. 52, no. 3, pp. 425–432, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016813000537

[94] A. Favenza, A. Farasin, N. Linty, and F. Dovis, "A machine learning approach to GNSS scintillation detection: Automatic soft inspection of the events," in *Proc. 30th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2017, pp. 4103–4111. [Online]. Available: http://www.ion.org/publications/abstract.cfm?jp=p&articleID=15351

[95] N. Linty, A. Farasin, A. Favenza, and F. Dovis, "Detection of GNSS ionospheric scintillations based on machine learning decision tree," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 1, pp. 303–317, Feb. 2019, doi: 10.1109/TAES.2018.2850385.

[96] R. Imam and F. Dovis, "Distinguishing ionospheric scintillation from multipath in GNSS signals using bagged decision trees algorithm," in *Proc. IEEE Int. Conf. Wireless Space Extreme Environ.*, 2020, pp. 83–88, doi: 10.1109/WiSEE44079.2020.9262699.

[97] Y. Jiao, J. Hall, and Y. J. Morton, "Performance evaluations of an equatorial GPS amplitude scintillation detector using a machine learning algorithm," in *Proc. 29th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2016, pp. 195–199. [Online]. Available: http://www.ion.org/publications/abstract.cfm?jp=p&articleID=14554

[98] Y. Jiao, J. J. Hall, and Y. T. Morton, "Performance evaluation of an automatic GPS ionospheric phase scintillation detector using a machine-learning algorithm," *Navigation: J. Inst. Navigation*, vol. 64, no. 3, pp. 391–402, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/navi.188

[99] L. Mengying, Z. Xuefen, L. Yimei, and Y. Fan, "Analysis of ionospheric scintillation detection based on machine learning," in *Proc. Int. Conf. Sens., Meas. Data Anal. Era Artif. Intell.*, 2020, pp. 357–361, doi: 10.1109/ICSMD50554.2020.9261642.

[100] Y. Jiao, J. J. Hall, and Y. T. Morton, "Automatic equatorial GPS amplitude scintillation detection using a machine learning algorithm," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 1, pp. 405–418, Feb. 2017, doi: 10.1109/TAES.2017.2650758.

[101] Y. Liu and Y. J. Morton, "Automatic detection of ionospheric scintillation-like GNSS satellite oscillator anomaly using a machine-learning algorithm," *Navigation: J. Inst. Navigation*, vol. 67, no. 3, pp. 651–662, 2020, doi: 10.1002/navi.385.

[102] Y. Liu and Y. J. Morton, " Machine learning based automatic detection of ionospheric scintillation-like GNSS oscillator anomaly using dual frequency signals," in *Proc. 51st Annu. Precise Time Time Interval Syst. Appl. Meeting*, 2020, pp. 366–373, doi: 10.33012/2020.17311.

[103] T. Desert, T. Authié, and S. Trilles, "Modelling of the ionosphere by neural network for equatorial SBAS," in *Proc. 28th Int. Tech. Meeting Satell. Div. Inst. Navigation*, Sep. 2015, pp. 3542–3549. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01218339/document

[104] R. O. Perez, "Using tensorflow-based neural network to estimate GNSS single frequency ionospheric delay (IONONet)," *Adv. Space Res.*, vol. 63, no. 5, pp. 1607–1618, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117718308652

[105] M. Kim and J. Kim, "Extending ionospheric correction coverage area by using a neural network method," *Int. J. Aeronautical Space Sci.*, vol. 17, pp. 64–72, 2016, doi: 10.5139/IJASS.2016.17.1.64.

[106] L. Mallika I, D. V. Ratnam, S. Raman, and G. Sivavaraprasad, "Machine learning algorithm to forecast ionospheric time delays using global navigation satellite system observations," *Acta Astronautica*, vol. 173, pp. 221–231, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576520302630

[107] R. Shamshiri, M. Motagh, H. Nahavandchi, M. Haghshenas Haghighi, and M. Hoseini, "Improving tropospheric corrections on large-scale Sentinel-1 interferograms using a machine learning approach for integration with GNSS-derived zenith total delay (ZTD)," *Remote Sens. Environ.*, vol. 239, 2020, Art. no. 111608. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425719306285

[108] Z. Zhang, S. Pan, C. Gao, T. Zhao, and W. Gao, "Support vector machine for regional ionospheric delay modeling," *Sensors*, vol. 19, no. 13, 2019, Art. no. 2947. [Online]. Available: https://www.mdpi.com/1424-8220/19/13/2947

[109] M. O. Selbesoglu, "Prediction of tropospheric wet delay by an artificial neural network model based on meteorological and GNSS data," *Eng. Sci. Technol., Int. J.*, vol. 23, no. 5, pp. 967–972, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098619316003

[110] L. Miotti et al., "Tropospheric delays derived from ground meteorological parameters: Comparison between machine learning and empirical model approaches," in *Proc. Eur. Navigation Conf.*, 2020, pp. 1–10, doi: 10.23919/ENC48637.2020.9317442.

[111] J. Mohammed, "Artificial neural network for predicting global sub-daily tropospheric wet delay," *J. Atmospheric Sol.- Terr. Phys.*, vol. 217, 2021, Art. no. 105612. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364682621000730

[112] M. O. Selbesoglu, "Spatial interpolation of GNSS troposphere wet delay by a newly designed artificial neural network model," *Appl. Sci.*, vol. 9, no. 21, 2019, Art. no. 4688. [Online]. Available: https://www.mdpi.com/2076-3417/9/21/4688

[113] C. Pikridas, S. Katsougiannopoulos, and I. M. Ifadis, "Predicting zenith tropospheric delay using the artificial neural network technique. application to selected EPN stations," *J. Nat. Cancer Inst.*, vol. 88, no. 24, pp. 1803–1805, 2010. [Online]. Available: https://www.researchgate.net/publication/258330194_Predicting_Zenith_Tropospheric_Delay_using_the_Artificial_Neural_Network_technique_Application_to_selected_EPN_stations

[114] Y. Yang, T. Xu, and L. Ren, "A new regional tropospheric delay correction model based on BP neural network," in *Proc. Forum Cooperative Positioning Serv.*, 2017, pp. 96–100, doi: 10.1109/CPGPS.2017.8075104.

[115] A. Hu et al., "Improvement of reflection detection success rate of GNSS RO measurements using artificial neural network," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 760–769, Feb. 2018, doi: 10.1109/TGRS.2017.2754512.

[116] M. L. Psiaki and T. E. Humphreys, "GNSS spoofing and detection," *Proc. IEEE*, vol. 104, no. 6, pp. 1258–1270, Jun. 2016, doi: 10.1109/JPROC.2016.2526658.

[117] M. R. Manesh, J. Kenney, W. C. Hu, V. K. Devabhaktuni, and N. Kaabouch, "Detection of GPS spoofing attacks on unmanned aerial systems," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf.*, 2019, pp. 1–6, doi: 10.1109/CCNC.2019.8651804.

[118] S. Tohidi and M. R. Mosavi, "Effective detection of GNSS spoofing attack using a multi-layer perceptron neural network classifier trained by PSO," in *Proc. 25th Int. Comput. Conf., Comput. Soc. Iran*, 2020, pp. 1–5, doi: 10.1109/CSICC49403.2020.9050078.

[119] R. Calvo-Palomino, A. Bhattacharya, G. Bovet, and D. Giustiniano, "Short: LSTM-based GNSS spoofing detection using low-cost spectrum sensors," in *Proc. IEEE 21st Int. Symp. 'A World Wireless, Mobile Multimedia Netw.'*, 2020, pp. 273–276, doi: 10.1109/WoWMoM49955.2020.00055.

[120] S. Semanjski, A. Muls, I. Semanjski, and W. De Wilde, "Use and validation of supervised machine learning approach for detection of GNSS signal spoofing," in *Proc. Int. Conf. Localization GNSS*, 2019, pp. 1–6, doi: 10.1109/ICL-GNSS.2019.8752775.

[121] W. Li, Z. Huang, R. Lang, H. Qin, K. Zhou, and Y. Cao, "A real-time interference monitoring technique for GNSS based on a twin support vector machine method," *Sensors*, vol. 16, no. 3, 2016, Art. no. 329. [Online]. Available: https://www.mdpi.com/1424-8220/16/3/329

[122] R. M. Ferre, A. de la Fuente, and E. S. Lohan, "Jammer classification in GNSS bands via machine learning algorithms," *Sensors*, vol. 19, no. 22, 2019, Art. no. 4841. [Online]. Available: https://www.mdpi.com/1424-8220/19/22/4841

[123] D. R. Kartchner, R. Palmer, and S. K. Jayaweera, "Satellite navigation anti-spoofing using deep learning on a receiver network," in *Proc. IEEE Cogn. Commun. Aerosp. Appl. Workshop*, 2021, pp. 1–5, doi: 10.1109/CCAAW50069.2021.9527295.

[124] J. R. V. D. Merwe et al., "Blind spoofing detection for multi-antenna snapshot receivers using machine-learning techniques," in *Proc. 33th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2020, pp. 3294–3312, doi: 10.33012/2020.17564.

[125] E. Shafiee, M. Mosavi, and M. Moazedi, "Detection of spoofing attack using machine learning based on multi-layer neural network in single-frequency GPS receivers," *J. Navigation*, vo. 71, no. 1, pp. 169–188, 2017, doi: 10.1017/S0373463317000558.

[126] G. Paniceet al., "A SVM-based detection approach for GPS spoofing attacks to UAV," in *Proc. 23rd Int. Conf. Automat. Comput.*, 2017, pp. 1–11, doi: 10.23919/IConAC.2017.8081999.

[127] Y. Zhou, J. Wan, Z. Li, and Z. Song, "GPS/INS integrated navigation with BP neural network and Kalman filter," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2017, pp. 2515–2520, doi: 10.1109/ROBIO.2017.8324798.

[128] G. Wang, X. Xu, Y. Yao, and J. Tong, "A novel BPNN-based method to overcome the GPS outages for INS/GPS system," *IEEE Access*, vol. 7, pp. 82134–82143, 2019, doi: 10.1109/ACCESS.2019.2922212.

[129] Z. Zou, T. Huang, L. Ye, and K. Song, "CNN based adaptive Kalman filter in high-dynamic condition for low-cost navigation system on highspeed UAV," in *Proc. 5th Asia-Pacific Conf. Intell. Robot Syst.*, 2020, pp. 103–108, doi: 10.1109/ACIRS49895.2020.9162601.

[130] N. Liu, Z. Hui, Z. Su, L. Qiao, and Y. Dong, "Integrated navigation on vehicle based on low-cost SINS/GNSS using deep learning," *Wireless Pers. Commun.*,, vol. 126, pp. 2043–2064, 2022, doi: 10.1007/s11277-021-08758-9.

[131] E. Adbolkarimi, G. Abaei, and M. Mosavi, "A wavelet-extreme learning machine for low-cost INS/GPS navigation system in high-speed applications," *GPS Solutions*, vol. 22, no. 15, pp. 1–13, 2018, doi: 10.1007/s10291-017-0682-x.

[132] J.-R. De Boer, V. Calmettes, J.-Y. Tourneret, and B. Lesot, "Outage mitigation for GNSS/MEMS navigation using neural networks," in *Proc. 17th Eur. Signal Process. Conf.*, 2009, pp. 2156–2160.

[133] L. Cong, S. Yue, H. Qin, B. Li, and J. Yao, "Implementation of a MEMS-based GNSS/INS integrated scheme using supported vector machine for land vehicle navigation," *IEEE Sensors J.*, vol. 20, no. 23, pp. 14423–14435, Dec. 2020, doi: 10.1109/JSEN.2020.3007892.

[134] A. Mohanty and G. Gao, "A particle filtering framework for tight GNSS-camera fusion using convolutional neural networks," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 2646–2655, doi: 10.33012/2021.17940.

[135] H. fa Dai, H. wei Bian, R. ying Wang, and H. Ma, "An INS/GNSS integrated navigation in GNSS denied environment using recurrent neural network," *Defence Technol.*, vol. 16, no. 2, pp. 334–340, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214914719303058

[136] R. Sharaf, A. Noureldin, A. Osman, and N. El-Sheimy, "Online INS/GPS integration with a radial basis function neural network," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 20, no. 3, pp. 8–14, Mar. 2005, doi: 10.1109/MAES.2005.1412121.

[137] Z. Liu, N. Wu, and F. Wang, "Application of neural network-assisted GNSS/SINS calibration system in UUV," in *Proc. IEEE 3rd Adv. Inf. Manage., Communicates, Electron. Automat. Control Conf.*, 2019, pp. 1253–1257, doi: 10.1109/IMCEC46724.2019.8984176.

[138] Y.-C. Lin, Y.-W. Huang, and K.-W. Chiang, "A neural-KF hybrid sensor fusion scheme for INS/GPS/odometer integrated land vehicular navigation system," in *Proc. 19th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2006, pp. 2174–2181. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=7085

[139] J. J. Wang, W. Ding, and J. Wang, "Improving adaptive Kalman filter in GPS/SDINS integration with neural network," in *Proc. 20th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2007, pp. 571–578. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=7555

[140] A. Noureldin, A. El-Shafie, and M. Bayoumi, "GPS/INS integration utilizing dynamic neural networks for vehicular navigation," *Inf. Fusion*, vol. 12, no. 1, pp. 48–57, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253510000175

[141] T. Soininen, P. Syrjarinne, S. Ali-Loytty, and C. Schmid, "Data-driven approach to satellite selection in multi-constellation GNSS receivers," in *Proc. 8th Int. Conf. Localization GNSS*, 2018, pp. 1–6, doi: 10.1109/ICL-GNSS.2018.8440912.

[142] T. Soininen, P. Syrjarinne, S. Ali-Loytty, and C. Schmid, "Data-driven approach to satellite selection in multi-constellation GNSS receivers," in *Proc. 8th Int. Conf. Localization GNSS*, 2018, pp. 1–6, doi: 10.1109/ICL-GNSS.2018.8440912.

[143] P. Huang, C. Rizos, and C. Roberts, "Satellite selection with an end-to-end deep learning network," *GPS Solutions*, vol. 22, no. 4, 2018, Art. no. 108.

[144] D. Simon and H. El-Sherief, "Navigation satellite selection using neural networks," *Neurocomputing*, vol. 7, no. 3, pp. 247–258, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/092523129400024 M

[145] D. Gebre-Egziabher and S. Gleason, *GNSS Applications and Methods*. Norwood, MA, USA: Artech House, 2009, p. 335.

[146] T. Mortlock and Z. M. Kassas, "Assessing machine learning for LEO satellite orbit determination in simultaneous tracking and navigation," in *Proc. IEEE Aerosp. Conf.*, 2021, pp. 1–8, doi: 10.1109/AERO50100.2021.9438144.

[147] W. Vigneau et al., "Neural networks algorithms prototyping to mitigate GNSS multipath for LEO positioning applications," in *Proc. 19th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2006, pp. 1752–1762. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=7068

[148] P. Ramos-Bosch, M. Hernández-Pajares, J. Juan, and J. Sanz, "Real time GPS positioning of LEO satellites mitigating pseudorange multipath through neural networks," *J. Inst. Navigation*, vol. 54, pp. 309–315, 2007, doi: 10.1002/j.2161-4296.2007.tb00411.x.

[149] S. E. Kozhaya, J. A. Haidar-Ahmad, A. A. Abdallah, Z. M. Kassas, and S. S. Saab, "Comparison of neural network architectures for simultaneous tracking and navigation with LEO satellites," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 2507–2520, doi: 10.33012/2021.18110.

[150] E. Munin, A. Blais, and N. Couellan, "Convolutional neural network for multipath detection in GNSS receivers," in *Proc. Int. Conf. Artif. Intell. Data Anal. Air Transp.*, 2020, pp. 1–10, doi: 10.1109/AIDA-AT48540.2020.9049188.

[151] A. Louis, "Neural network based evil waveforms detection," in *Proc. RFI Workshop - Coexisting Radio Freq. Interference*, 2019, pp. 1–4, doi: 10.23919/RFI48793.2019.9111769.

[152] L. M. I., D. V. Ratnam, S. Raman, and G. Sivavaraprasad, "Machine learning algorithm to forecast ionospheric time delays using global navigation satellite system observations," *Acta Astronautica*, vol. 173, pp. 221–231, 2020, doi: 10.1016/j.actaastro.2020.04.048.

[153] R. O. Perez, "Using TensorFlow-based neural network to estimate GNSS single frequency ionospheric delay (IONONet)," *Adv. Space Res.*, vol. 63, no. 5, pp. 1607–1618, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0273117718308652

[154] Y. Liu, Y. J. Morton, and Y. Jiao, "Application of machine learning to the characterization of GPS L1 ionospheric amplitude scintillation," in *Proc. IEEE/ION Position*, 2018, pp. 1159–1166, doi: 10.1109/PLANS.2018.8373500.

[155] N. Linty, A. Farasin, A. Favenza, and F. Dovis, "Detection of GNSS ionospheric scintillations based on machine learning decision tree," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 1, pp. 303–317, Feb. 2019, doi: 10.1109/TAES.2018.2850385.

[156] L.-T. Hsu, "GNSS multipath detection using a machine learning approach," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–6, doi: 10.1109/ITSC.2017.8317700.

[157] Y. Quan, L. Lau, G. W. Roberts, X. Meng, and C. Zhang, "Convolutional neural network based multipath detection method for static and kinematic GPS high precision positioning," *Remote Sens.*, vol. 10, no. 12, 2018, Art. no. 2052. [Online]. Available: https://www.mdpi.com/2072-4292/10/12/2052

[158] G. Gogliettino, M. Renna, F. Pisoni, D. Di Grazia, and D. Pau, "A machine learning approach to GNSS functional safety," in *Proc. 32nd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2019, pp. 1738–1752. [Online]. Available: http://www.ion.org/publications/abstract.cfm?jp=p&articleID=17001

[159] N. Yamaga and Y. Mitsul, "Machine learning approach to characterize the postseismic deformation of the 2011 Tohoku-Oki earthquake based on recurrent neural network," *Geophys. Res. Letters, Advancing Earth Space Sci.*, vol. 46, pp. 11886–11892, 2019, doi: 10.1029/2019GL084578.

[160] H. Li, P. Borhani-Darian, P. Wu, and P. Closas, "Deep learning of GNSS signal correlation," in *Proc. 33rd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2020, pp. 2836–2847, doi: 10.33012/2020.17598.

[161] Y. Luo, X. Zhu, M. Lin, and F. Yang, "Detection method of solar radio bursts based on support vector machine model," in *Proc. Int. Conf. Sens., Meas. Data Anal. Era Artif. Intell.*, 2020, pp. 362–366, doi: 10.1109/ICSMD50554.2020.9261712.

[162] M. Orabi, J. Khalife, A. A. Abdallah, Z. M. Kassas, and S. S. Saab, "A machine learning approach for GPS code phase estimation in multipath environments," in *Proc. IEEE/ION Position, Location Navigation Symp.*, 2020, pp. 1224–1229. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=17503

[163] H. Yousif and A. El-Rabbany, "GPS orbital prediction using artificial neural networks," in *Proc. Nat. Tech. Meeting Inst. Navigation*, 2008, pp. 773–780. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=7736

[164] A. R. Kazemi, S. Tohidi, and M. R. Mosavi, "Enhancing classification performance between different GNSS interferences using neural networks trained by TAC-PSO algorithm," in *Proc. 10th Int. Symp. Telecommun.*, 2020, pp. 150–154, doi: 10.1109/IST50524.2020.9345914.

[165] M. Kiani, "On the suitability of generalized regression neural networks for GNSS position time series prediction for geodetic applications in geodesy and geophysics," 2020, *arXiv:2005.11106*.

[166] M. Kaselimi, A. Voulodimos, N. Doulamis, A. Doulamis, and D. Delikaraoglou, "A causal long short-term memory sequence to sequence model for TEC prediction using GNSS observations," *Remote Sens.*, vol. 12, no. 9, 2020, Art. no. 1354. [Online]. Available: https://www.mdpi.com/2072-4292/12/9/1354

[167] G. Xiaet al., "Ionospheric TEC forecast model based on support vector machine with GPU acceleration in the China region," *Adv. Space Res.*, vol. 68, no. 3, pp. 1377–1389, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117721002441

[168] D. Okohet al., "A regional GNSS-VTEC model over Nigeria using neural networks: A novel approach," *Geodesy Geodynamics*, vol. 7, no. 1, pp. 19–31, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1674984716300052

[169] S. Sahu, R. Trivedi, R. Choudhary, A. Jain, and S. Jain, "Prediction of total electron content (TEC) using neural network over anomaly crest region Bhopal," *Adv. Space Res.*, vol. 68, no. 7, pp. 2919–2929, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117721004476

[170] G. Sivavaraprasad, V. Deepika, D. SreenivasaRao, M. R. Kumar, and M. Sridhar, "Performance evaluation of neural network TEC forecasting models over equatorial low-latitude Indian GNSS station," *Geodesy Geodynamics*, vol. 11, no. 3, pp. 192–201, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1674984719300242

[171] S. Rafatnia, H. Nourmohammadi, J. Keighobadi, and M. Badamchizadeh, "In-move aligned SINS/GNSS system using recurrent wavelet neural network (RWNN)-based integration scheme," *Mechatronics*, vol. 54, pp. 155–165, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957415818301272

[172] A. A. Ferreira, R. A. Borges, C. Paparini, L. Ciraolo, and S. M. Radicella, "Short-term estimation of GNSS TEC using a neural network model in Brazil," *Adv. Space Res.*, vol. 60, no. 8, pp. 1765–1776, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117717304088

[173] B. Huang et al., "Clock bias prediction algorithm for navigation satellites based on a supervised learning long short-term memory neural network," *GPS Solutions*, vol. 25, 2021, Art. no. 80, doi: 10.1007/s10291-021-01115-0.

[174] C. Xu, Z. Li, X. Hua, and Q. Fan, "Regional TEC model using improved neural network and its application in single frequency precise point positioning," in *Proc. 21st Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2008, pp. 2434–2439. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=8145

[175] J. B. Habarulema, L.-A. McKinnell, P. J. Cilliers, and B. D. Opperman, "Application of neural networks to South African GPS TEC modelling," *Adv. Space Res.*, vol. 43, no. 11, pp. 1711–1720, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117709001288

[176] C. Savas and F. Dovis, "Multipath detection based on K-means clustering," in *Proc. 32nd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2019, pp. 3801–3811, doi: 10.33012/2019.17028.

[177] F. Sabzehee, S. Farzaneh, M. A. Sharifi, and M. Akhoonzadeh, "TEC regional modeling and prediction using ANN method and single frequency receiver over Iran," *Ann. Geophys.*, vol. 61, no. 1, 2018, Art. no. GM103, doi: 10.4401/ag-7297.

[178] J. Hu, Z. Wu, X. Qin, H. Geng, and Z. Gao, "An extended Kalman filter and back propagation neural network algorithm positioning method based on anti-lock brake sensor and global navigation satellite system information," *Sensors*, vol. 18, no. 9, 2018, Art. no. 2753. [Online]. Available: https://www.mdpi.com/1424-8220/18/9/2753

[179] M. Socharoentum and H. Karimi, "A machine learning approach to detect non-line-of-sight GNSS signals in Nav2Nav," in *Proc. 23rd ITS World Congr.*, 2016, pp. 10–14.

[180] E. Munin, A. Blais, and N. Couellan, "Convolutional neural network for multipath detection in GNSS receivers," in *Proc. Int. Conf. Artif. Intell. Data Anal. Air Transp.*, 2020, pp. 1–10, doi: 10.1109/AIDA-AT48540.2020.9049188.

[181] P. Borhani-Darian, H. Li, P. Wu, and P. Closas, "Deep neural network approach to detect GNSS spoofing attacks," in *Proc. 33rd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2020, pp. 3241–3252, doi: 10.33012/2020.17537.

[182] Y. Liu and Y. J. Morton, "Improved automatic detection of GPS satellite oscillator anomaly using a machine learning algorithm," in *Proc. 33rd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2020, pp. 3567–3580, doi: 10.33012/2020.17674.

[183] S. J. Cho, B. Seong Kim, T. S. Kim, and S.-H. Kong, "Enhancing GNSS performance and detection of road crossing in urban area using deep learning," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2019, pp. 2115–2120, doi: 10.1109/ITSC.2019.8917224.

[184] K. Maschera, S. Lallera, and M. Wieser, "Development of smart shin guards for soccer performance analysis based on MEMS accelerometers, machine learning, and GNSS," in *Proc. ICL-GNSS 2021 WiP Proc.*, Jun. 03, 2021. [Online]. Available: http://ceur-ws.org/Vol-2880/paper5.pdf

[185] Q. Yuan et al., "Monitoring the variation of vegetation water content with machine learning methods: Point–surface fusion of modis products and GNSS-IR observations," *Remote Sens.*, vol. 11, no. 12, 2019, Art. no. 1440. [Online]. Available: https://www.mdpi.com/2072-4292/11/12/1440

[186] J. Li, N. Song, G. Yang, M. Li, and Q. Cai, "Improving positioning accuracy of vehicular navigation system during GPS outages utilizing ensemble learning algorithm," *Inf. Fusion*, vol. 35, pp. 1–10, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253516300641

[187] A. Yilmaz, K. Akdogan, and M. Gurun, "Regional TEC mapping using neural networks," *Radio Sci., Adv. Earth Space Sci.*, vol. 44, no. 3, Jun. 2009, Art. no. RS3007,, doi: 10.1029/2008RS004049.

[188] W. Machado and E. J. Fonseca, "VTEC prediction at Brazilian region using artificial neural networks," in *Proc. 24th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2011, pp. 2552–2560. [Online]. Available: http://www.ion.org/publications/abstract.cfm?jp=p&articleID=9808

[189] Q. Yang, Y. Zhang, B. Lian, and C. Tang, "Airborne GPS interference cancellation algorithm based on deep learning," in *Proc. 30th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2017, pp. 1695–1700. [Online]. Available: http://www.ion.org/publications/abstract.cfm?jp=p&articleID=15176

[190] L.-T. Hsu, "GNSS multipath detection using a machine learning approach," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–6, doi: 10.1109/ITSC.2017.8317700.

[191] T. Suzuki, Y. Nakano, and Y. Amano, "NLOS multipath detection by using machine learning in urban environments," in *Proc. 30th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2017, pp. 3958–3967. [Online]. Available: http://www.ion.org/publications/abstract.cfm?jp=p&articleID=15291

[192] Z. Zhang, F. Wu, Y. Liu, Y. Zuo, and Y. Ji, "A neural network aided integrated navigation algorithm based on vehicle motion mode information," in *Proc. IEEE Int. Conf. Signal Process., Commun. Comput.*, 2019, pp. 1–5, doi: 10.1109/ICSPCC46631.2019.8960761.

[193] A. LOUIS, "Neural network based evil waveforms detection," in *Proc. RFI Workshop - Coexisting Radio Freq. Interference*, 2019, pp. 1–4, doi: 10.23919/RFI48793.2019.9111769.

[194] K. Lamb et al., "Prediction of GNSS phase scintillations: A machine learning approach," 2019, *arXiv:1910.01570*.

[195] E. Munin, A. Blais, and N. Couellan, "GNSS multipath detection using embedded deep CNN on intel neural compute stick," in *Proc. 33rd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2020, pp. 2018–2029, doi: 10.33012/2020.17654.

[196] T. Suzuki, Y. Nakano, and Y. Amano, "NLOS multipath detection by using machine learning in urban environments," in *Proc. 30th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2017, pp. 3958–3967, doi: 10.33012/2017.15291.

[197] H. Lan, Y. B. Sarvrood, A. Moussa, and N. El-Sheimy, "Zero velocity detection for un-tethered vehicular navigation systems using support vector machine," in *Proc. 33rd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2020, pp. 1990–2002, doi: 10.33012/2020.17652.

[198] S. Semanjski, I. Semanjski, W. D. Wilde, and S. Gautama, "Use of supervised machine learning for GNSS signal spoofing detection with validation on real-world meaconing and spoofing data—Part II," *Sensors*, vol. 20, 2020, Art. no. 1171, doi: 10.3390/s20071806.

[199] F. Dovis, R. Imam, W. Qin, C. Savas, and H. Visser, "Opportunistic use of GNSS signals to characterize the environment by means of machine learning based processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 9190–9194, doi: 10.1109/ICASSP40776.2020.9052924.

[200] E. I. Adegokeet al., "Evaluating machine learning amp; antenna placement for enhanced GNSS accuracy for CAVS," in *Proc. IEEE Intell. Veh. Symp.*, 2019, pp. 1007–1012, doi: 10.1109/IVS.2019.8813775.

[201] J. Wang, Q. Yuan, T. Li, H. Shen, and L. Zhang, "Estimating snow-depth by fusing satellite and station observations: A deep learning approach," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2019, pp. 4109–4112, doi: 10.1109/IGARSS.2019.8900518.

[202] M. Y. Klimenko and A. V. Veitsel, "Evaluation of neural network-based multipath mitigation approach for the GNSS receivers," in *Proc. Syst. Signal Synchronization, Generating Process. Telecommun.*, 2021, pp. 1–5, doi: 10.1109/SYNCHROINFO51390.2021.9488410.

[203] Z. Özdemir and B. Tuğrul, "Geofencing on the real-time GPS tracking system and improving GPS accuracy with moving average, Kalman filter and logistic regression analysis," in *Proc. 3rd Int. Symp. Multidisciplinary Stud. Innov. Technol.*, 2019, pp. 1–6, doi: 10.1109/ISMSIT.2019.8932766.

[204] W. Ye, B. Wang, Y. Liu, B. Gu, and H. Chen, "Deep Gaussian process regression for performance improvement of POS during GPS outages," *IEEE Access*, vol. 8, pp. 117483–117492, 2020, doi: 10.1109/ACCESS.2020.3004706.

[205] L. He, H. Zhou, S. Zhu, and P. Zeng, "An improved QZSS satellite clock offsets prediction based on the extreme learning machine method," *IEEE Access*, vol. 8, pp. 156557–156568, 2020, doi: 10.1109/ACCESS.2020.3019941.

[206] E. S. Fogarty, D. L. Swain, G. M. Cronin, L. E. Moraes, D. W. Bailey, and M. Trotter, "Developing a simulated online model that integrates GNSS, accelerometer and weather data to detect parturition events in grazing sheep: A machine learning approach," *Animals*, vol. 11, no. 2, 2021, Art. no. 303. [Online]. Available: https://www.mdpi.com/2076-2615/11/2/303

[207] H. Xu, A. Angrisano, S. Gaglione, and L. Hsu, "Machine learning based LOS/NLOS classifier and robust estimator for GNSS shadow matching," *Satell. Navigation*, vol. 1, no. 1, 2020, Art. no. 15. [Online]. Available: https://link.springer.com/content/pdf/10.1186/s43020-020-00016-w.pdf

[208] S. Li, T. Xu, N. Jiang, H. Yang, S. Wang, and Z. Zhang, "Regional zenith tropospheric delay modeling based on least squares support vector machine using GNSS and ERA5 data," *Remote Sens.*, vol. 13, no. 5, 2021, Art. no. 1004. [Online]. Available: https://www.mdpi.com/2072-4292/13/5/1004

[209] M. Kaselimi, N. Doulamis, A. Doulamis, and D. Delikaraoglou, "A sequence-to-sequence temporal convolutional neural network for ionosphere prediction using GNSS observations," *Int. Arch. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 43, pp. 813–820, 2020, [Online]. Available: 10.5194/isprs-archives-XLIII-B3-2020-813-2020

[210] D. Min, M. Kim, J. Lee, and J. Lee, "Deep neural network based multipath mitigation method for carrier based differential GNSS systems," in *Proc. ION Pacific PNT Meeting*, 2019, pp. 451–466, doi: 10.33012/2019.16856.

[211] X. Zou, B. Lian, and P. Wu, "Fault identification ability of a robust deeply integrated GNSS/INS system assisted by convolutional neural networks," *Sensors*, vol. 19, no. 12, 2019, Art. no. 2734. [Online]. Available: https://www.mdpi.com/1424-8220/19/12/2734

[212] M. Moses, J. D. Dodo, L. M. Ojigi, and K. Lawal, "Regional TEC modelling over Africa using deep structured supervised neural network," *Geodesy Geodynamics*, vol. 11, pp. 367–375, 2020, doi: 10.1016/j.geog.2020.05.004.

[213] L. Zhao and H. Quan, "Using regularized softmax regression in the GNSS/INS integrated navigation system with nonholonomic constraints," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 538, no. 1, 2019, Art. no. 012058. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/538/1/012058/meta

[214] S. P. Mutchakayala, V. M. Mandalapu, J. K. Dabbakuti, and S. S. Vedula, "Machine learning methodology for TEC prediction using global positioning system signal measurements," *Mater. Today: Proc.*, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221478532037629X

[215] S. Osah, A. A. Acheampong, C. Fosu, and I. Dadzie, "Deep learning model for predicting daily IGS zenith tropospheric delays in West Africa using TensorFlow and Keras," *Adv. Space Res.*, vol. 68, no. 3, pp. 1243–1262, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117721003380

[216] F. W. Salar, J. D. Chowdary, C. R. Reddy, M. V. Rao, and S. K. Panda, "Withdrawn: Implementation of deep learning algorithms for predicting ionospheric total electron content," *Mater. Today: Proc.*, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785321006295

[217] L. Li, Y. Xu, L. Yan, S. Wang, G. Liu, and F. Liu, "A regional NWP tropospheric delay inversion method based on a general regression neural network model," *Sensors*, vol. 20, no. 11, 2020, Art. no. 3167. [Online]. Available: https://www.mdpi.com/1424-8220/20/11/3167

[218] H. Ragab, S. K. Abdelaziz, M. Elhabiby, S. Givigi, and A. Noureldin, "Machine learning-based visual odometry uncertainty estimation for low-cost integrated land vehicle navigation," in *Proc. 33rd Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2020, pp. 2569–2578, doi: 10.33012/2020.17740.

[219] H. No and C. Milner, "Machine learning based overbound modeling of multipath error for safety critical urban environment," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 180–194. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=17874

[220] Q. Zhong and P. D. Groves, "Multi-epoch 3D-mapping-aided positioning using Bayesian filtering techniques," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 195–225, doi: 10.33012/2021.17894.

[221] N. Harbaoui, N. A. Tmazirte, K. Makkawi, and M. E. B. E. Najjar, "Navigation context adaptive fault detection and exclusion strategy based on deep learning and information theory: Application to a GNSS/IMU integration," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 1809–1827, doi: 10.33012/2021.17970.

[222] K.-B. Wu, Y. Liu, and Y. J. Morton, "Automatic detection of Galileo satellite oscillator anomaly by using a machine learning algorithm," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 2925–2939, doi: 10.33012/2021.17992.

[223] L. Liu, Y. J. Morton, and Y. Liu, "Machine learning prediction of highlatitude ionospheric irregularities from GNSS-derived ROTI maps," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 3870–3877, doi: 10.33012/2021.18046.

[224] Y. Liu, Z. Yang, Y. T. J. Morton, and R. Li, "Spatiotemporal deep learning network for high-latitude ionopsheric phase scintillation forecasting," in *Proc. 34th Int. Tech. Meeting Satell. Div. Inst. Navigation*, 2021, pp. 3920–3931, doi: 10.33012/2021.18061.

[225] Q.-H. Phan, S.-L. Tan, I. McLoughlin, and D.-L. Vu, "A unified framework for GPS code and carrier-phase multipath mitigation using support vector regression," *Adv. Artif. Neural Syst.*, vol. 2013, 2013, Art. no. 240564, doi: 10.1155/2013/240564.

[226] J. B. Habarulema, L.-A. McKinnell, and B. D. Opperman, "Towards a GPS-based TEC prediction model for Southern Africa with feed forward networks," *Adv. Space Res.*, vol. 44, no. 1, pp. 82–92, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117709001690

[227] F. Lei, V. Senyurek, M. Kurum, A. Gurbuz, R. Moorhead, and D. Boyd, "Machine-learning based retrieval of soil moisture at high spatio-temporal scales using CYGNSS and SMAP observations," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2020, pp. 4470–4473, doi: 10.1109/IGARSS39084.2020.9323106.

[228] L. Liu et al., "Machine learning prediction of storm-time high-latitude ionospheric irregularities from GNSS-derived ROTI maps," *Geophysical Res. Lett., Advancing Earth Space Sci.*, vol. 48, no. 20, 2021, Art. no. e2021GL095561, doi: 10.1029/2021GL095561.

[229] S. Taabu, F. D'ujanga, and T. Ssenyonga, "Prediction of ionospheric scintillation using neural network over east African region during ascending phase of sunspot cycle 24," *Adv. Space Res.*, vol. 57, no. 7, pp. 1570–1584, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117716000375

[230] Z. Huang and H. Yuan, "Ionospheric single-station TEC short-term forecast using RBF neural network," *Radio Sci., Advancing Earth Space Sci.*, vol. 49, no. 4, pp. 283–292, 2014, doi: 10.1002/2013RS005247.

[231] J. B. Habarulema, L.-A. McKinnell, and P. J. Cilliers, "Prediction of global positioning system total electron content using neural networks over South Africa," *J. Atmos. Sol.- Terr. Phys.*, vol. 69, no. 15, pp. 1842–1850, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364682607002489

[232] R. Leandro and M. Santos, "A neural network approach for regional vertical total electron content modelling," *Studia Geophysica et Geodaetica*, vol. 51, no. 2, pp. 279–292, 2007, doi: 10.1007/s11200-007-0015-6.

[233] Y. Wang, Z. Lu, Y. Qu, L. Li, and N. Wang, "Improving prediction performance of GPS satellite clock bias based on wavelet neural network," *GPS Solutions*, vol. 21, no. 2, pp. 523–534, 2017, doi: 10.1007/s10291-016-0543-z.

[234] H. Azami and S. Sanei, "GPS GDOP classification via improved neural network trainings and principal component analysis," *Int. J. Electron.*, vol. 101, no. 9, pp. 1300–1313, 2014.

[235] N. Zarei, "Artificial intelligence approaches for GPS GDOP classification," *Int. J. Comput. Appl.*, vol. 96, no. 16, pp. 16–21, 2014. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?url=10.1.1.680.6283&rep=rep1&type=pdf

[236] S. Semanjski, I. Semanjski, W. De Wilde, and A. Muls, "Use of supervised machine learning for GNSS signal spoofing detection with validation on real-world meaconing and spoofing data—Part I," *Sensors*, vol. 20, no. 4, 2020, Art. no. 1171. [Online]. Available: https://www.mdpi.com/1424-8220/20/4/1171

[237] K. Tütüncü, M. A. Şahman, and E. Tuşat, "A hybrid binary grey wolf optimizer for selection and reduction of reference points with extreme learning machine approach on local GNSS/leveling geoid determination," *Appl. Soft Comput.*, vol. 108, 2021, Art. no. 107444. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494621003677

[238] M. Mendonça and M. C. Santos, *Assessment of a GNSS/INS/Wi-Fi Tight-Integration Method Using Support Vector Machine and Extended Kalman Filter*. Berlin, Germany: Springer, 2020, pp. 1–7, doi: 10.1007/1345_2020_120.

[239] S. Bibi and I. Stamelos, "Selecting the appropriate machine learning techniques for the prediction of software development costs," in *Artificial Intelligence Applications and Innovations*, I. Maglogiannis, K. Karpouzis, and M. Bramer, Eds., Boston, MA, USA: Springer, 2006, pp. 533–540.

[240] D. Zhang and J. Tsai, "Machine learning and software engineering," *Softw. Qual. Control*, vol. 11, pp. 87–119, 2003.

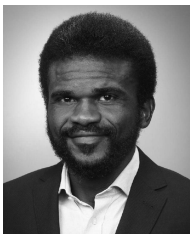
[241] S. G. MacDonell and M. J. Shepperd, "Combining techniques to optimize effort predictions in software project management," *J. Syst. Softw.*, vol. 66, no. 2, pp. 91–98, May 2003, doi: 10.1016/S0164-1212(02)00067-5.

[242] M. Jørgensen, "Forecasting of software development work effort: Evidence on expert judgement and formal models," *Int. J. Forecasting*, vol. 23, pp. 449–462, 2007.

[243] S. Bibi, I. Stamelos, and L. Angelis, "Combining probabilistic models for explanatory productivity estimation," *Inf. Softw. Technol.*, vol. 50, no. 7, pp. 656–669, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584907000730

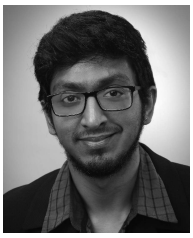
[244] M. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," *IEEE Trans. Softw. Eng.*, vol. 27, no. 11, pp. 1014–1022, Nov. 2001, doi: 10.1109/32.965341.

[245] G. Chan, "How to compare machine learning algorithms," 2020. [Online]. Available: https://towardsdatascience.com/how-to-compare-machine-learning-algorithms-ccc266c4777



Akpojoto Siemuri (Student Member, IEEE) received the B.S. degree in electrical and computer engineering from the Federal University of Technology, Minna, Nigeria, in 2010, the M.S. degree in wireless industrial automation, and a minor study in industrial management in 2019 from the University of Vaasa, Vaasa, Finland, where he is currently working toward the Ph.D. degree in automation technology.

From 2018 to 2019, he was a Research Assistant for the Smart Energy Systems Research Platform (SESP) Project with the University of Vaasa, where he is currently a Project Researcher for the Digital Economy Research Platform. His research interest includes machine learning, GNSS technologies, smart devices, embedded systems, communication systems, and game theory.



Kannan Selvan received the B.S. degree in electronics and communication engineering from Anna University, Chennai, India, in 2012, and the M.S degree in communications and systems engineering in 2020 from the University of Vaasa, Vaasa, Finland, where he is currently working toward the Ph.D. degree in automation technology.

From 2018 to 2020, he was a Research Assistant for the Digital Economy Research Platform with the University of Vaasa, where he is currently a Project Researcher for the Digital Economy Research Platform. His research interest includes GNSS technologies, satellite-data analysis, machine learning, satellite communication, and smart devices.

Heidi Kuusniemi(Member, IEEE) received the M.Sc. (Tech.) degree (with distinction) in 2002 and the D.Sc. (Tech.) degree from the Tampere University of Technology, Finland, in 2002 and 2005, respectively, both in information technology.

She is currently a Professor in Computer Science and the Director of Digital Economy with the University of Vaasa, Vaasa, Finland. She is also a part-time Research Professor in Satellite Navigation with the Finnish Geospatial Research Institute of the National Land Survey, Espoo, Finland. Her research interests include GNSS reliability and resilience, estimation and data fusion, mobile precision positioning, indoor localization, and PNT in new space.

Dr. Kuusniemi was a Member of the Council for Natural Sciences and Engineering at the Academy of Finland in 2019-2021 and is a Past Member of the Scientific Advisory Committee for GNSS (GSAC) at ESA.

Petri Valisuo received the M.Sc. (Tech.) degree in computer science from the Tampere University of Technology, Tampere, Finland, in 1996, and the D.Sc. (Tech.) degree in automation technology from the University of Vaasa, Vaasa, Finland, in 2011.

He worked for 10 years in the telecommunication industry before his research career in the University of Vaasa, where he is currently an Associate Professor (tenure track), Sustainable Automation, with the School of Technology and Innovation Management. He has authored and coauthored 27 peer-reviewed and more than 10 other scientific publications. His research interests include machine learning, IoT, positioning methods, and other technologies relevant to industrial automation.

Mohammed S. Elmusrati (Senior Member, IEEE) received the B.Sc. (with honors) and M.Sc. (with high honors) degrees in telecommunication engineering from the Electrical and Electronic Engineering Department, Benghazi (old name: Garyounis) University, Benghazi, Libya, in 1991 and 1995, respectively, and the Licentiate of Science in technology (with distinction) and the Doctor of Science in Technology (D.Sc.) degrees in automation and control engineering from Aalto University, Espoo, Finland, in 2002 and 2004, respectively.

He is currently a Full Professor and the Head of the Digitalization Unit, School of Technology and Innovations, University of Vaasa, Vaasa, Finland. He has published more than 145 peer reviewed papers, books, and reports. His research interests include wireless communications, artificial intelligence, machine learning, biotechnology, Big Data analysis, stochastic systems, and game theory.

Dr. Elmusrati is an active member in different scientific societies such as Member of the Society of Industrial and Applied Mathematics (SIAM) and Finnish Automation Society.

Publication V

Seamless navigation for indoor-outdoor positioning using GNSS-aided UWB/WiFi/IMU system

Akpojoto Siemuri[1], Mahmoud Elsanhoury[1], Kannan Selvan[1], Petri Välisuo[1], Heidi Kuusniemi[1,2], Mohammed S. Elmusrati[1]
[1] *School of Technology and Innovations, University of Vaasa, Finland*
[2] *Finnish Geospatial Research Institute, National Land Survey, Finland*

BIOGRAPHY

Akpojoto Siemuri received his B.Sc.(tech) degree in electrical and computer engineering from the Federal University of Technology Minna, Nigeria in 2010, an M.Sc.(tech) degree in wireless industrial automation with a minor study in industrial management from the University of Vaasa, Finland in 2019. He is currently pursuing a Ph.D. degree in automation technology at the University of Vaasa. From 2018 to 2019, he was a Research Assistant in the Smart Energy Systems Research Platform (SESP) Project at the University of Vaasa, Finland. He is currently a Project Researcher at the University of Vaasa. His research interest includes machine learning, GNSS technologies, Factor graphs, smart devices, embedded systems, communication systems, and game theory.

Mahmoud Elsanhoury is currently pursuing a doctoral (Ph.D.) degree in computer science and telecommunications engineering at the University of Vaasa, Finland. He received his M.Sc. (tech) degree in telecommunications engineering from Vaasa University in 2018, and his B.Sc. degree in telecommunications engineering from Alexandria University, Egypt in 2013. His current research interests cover ubiquitous indoor positioning systems, ultra-wideband (UWB) indoor localization, low-earth orbit (LEO) satellites for positioning, multisensor fusion technologies, Kalman filters, uncertain stochastic processes, and machine learning algorithms.

Kannan Selvan received his B.Sc.(tech) degree in Electronics and Communication Engineering from Anna University, India in 2012, the M.Sc.(tech) degree in Communications and Systems Engineering from the University of Vaasa, Finland in 2020. From 2018 to 2020, he was a Research Assistant in the Digital Economy Research Platform at the University of Vaasa, Finland. He is currently a Project Researcher and doctoral student at the University of Vaasa. His research interest includes LEO-PNT, GNSS technologies, satellite data analysis, machine learning, satellite communication, smart devices, and embedded Systems.

Petri Välisuo is currently working as an Associate Professor (tenure track), in sustainable automation, at the School of Technology and Innovations, University of Vaasa, Finland. He received an M.Sc.(tech) degree in computer science from the Tampere University of Technology, Finland, and a D.Sc.(Tech) degree in automation technology from the University of Vaasa, in years 1996 and 2011 respectively. He has authored and co-authored 27 peer-reviewed and more than 10 other scientific publications. His research interests cover machine learning, IoT, positioning methods, and other technologies relevant to industrial automation. He has been working for 10 years in the telecommunication industry before his research career at the University of Vaasa.

Dr. Heidi Kuusniemi is a professor of computer science and director of Digital Economy at the University of Vaasa in Finland. She is also a part-time research professor in satellite navigation at the Finnish Geospatial Research Institute. She has an M.Sc. (Tech.) degree (with distinction) from 2002 and a D.Sc. (Tech.) degree from 2005 in information technology, respectively, from Tampere University of Technology, Finland. She served as a member of the Council of Natural Sciences and Technology at the Academy of Finland 2019-2021 and was a member of the scientific advisory committee for GNSS (GSAC) at ESA. Her technical expertise and interests include GNSS reliability and resilience, estimation and data fusion, mobile precision positioning, indoor localization, and PNT in new space.

Prof. Mohammed S. Elmusrati received his B.Sc. (with honors) and M.Sc. (with high honors) degrees in electrical and electronic engineering, from the University of Benghazi, Libya, in 1991 and 1995, respectively, and the Licentiate of Science in Technology (with distinction) and the Doctor of Science in Technology (D.Sc.) degrees in automation and control engineering from Aalto University Finland, in 2002 and 2004, respectively. Currently, he is a Full Professor and Head of the Digitalization Unit at the School of Technology and Innovations – University of Vaasa, Finland. His research interest includes wireless communications, artificial intelligence, machine learning, biotechnology, big data analysis, stochastic systems, and game theory. Elmusrati has published more than 130 papers, books, and book chapters. Prof. Elmusrati is an active member of different scientific societies such as a Senior Member at IEEE, a Member of the Society of Industrial and Applied Mathematics (SIAM), and a Member of the Finnish Automation Society.

ABSTRACT

The need for seamless indoor-outdoor navigation is growing in different application fields, especially for factory scenarios, military, or first response emergency services. Multi-sensor fusion technology has become very prominent for seamless navigation systems owing to its complementary capabilities to Global Navigation Satellite System (GNSS) positioning. Machine learning (ML) and artificial intelligence (AI) solutions have also been widely adopted in literature in the last few years combining them with localization techniques for error mitigation and maximizing overall accuracy and system integrity. The research work of this paper is aimed at the following: firstly, to design an indoor-outdoor (IO) detection strategy that helps to correctly identify the transition between outdoor and indoor with reduced latency; secondly, to construct two separate integration schemes. GNSS integrated with indoor positioning technology (GNSS/UWB/IMU and GNSS/WiFi/IMU) can account for the GNSS signal distortion in indoor and urban environments. Based on this context, a loosely coupled architecture is implemented. The GNSS receiver helps the IMU to update with absolute positions from GNSS in the outdoor scenario. This positioning strategy takes advantage of both GNSS and UWB/IMU/WiFi combinations to realize a seamless indoor-outdoor positioning for personnel and indoor robots moving between the outdoor and indoor environments. The correct detection of the IO transition is essential in seamless IO positioning. This enables making the right decision on the navigation mode. Our implementation of indoor-outdoor (IO) detection strategies makes use of ML to find the IO signal transition pattern. The proposed system would be tested in a scenario over 1.1 km including outdoor, and indoor phases.

Keywords - Seamless positioning; Indoor-outdoor (IO) detection; Machine Learning; RINEX data.

I. INTRODUCTION

Global Navigation Satellite System (GNSS) is still a popular technique for outdoor positioning and navigation. However, it is limited and has well-known vulnerabilities when exposed to constrained environments such as urban canyons and indoor environments due to phenomena such as multi-path, Non-Line-of-Sight (NLOS) reception, and signal blockage. Some of these errors and vulnerabilities can be taken care of using techniques that help monitor the GNSS measurement quality, such as Fault Detection and Exclusion (FDE). In addition, certain GNSS system errors can even be directly removed by applying more accurate corrections, such as Precise Point Positioning (PPP). This is applicable to errors due to urban environments, however, there still remains the issue of indoor environments where it is impossible to use GNSS standalone.

There are a number of techniques that are being used to address indoor positioning, from the well-known Inertial Measurement Unit (IMU) to other technologies such as Wi-Fi positioning [1], Ultra-wideband (UWB) positioning [2], [3], Bluetooth low energy (BLE) or Bluetooth positioning [4], camera-based positioning [5], etc. These technologies are usually combined together so as to make the system more robust for different indoor contexts or scenarios.

There has been an increase in the amount of research work done based on the various developments of indoor and outdoor navigation technologies as mentioned above. This has increased the focus on seamless indoor-outdoor navigation. The main issues in the field of seamless indoor-outdoor navigation are not limited to only the choice of positioning techniques and algorithms, but also the detection of the transition between indoor-outdoor. In this work, we will develop a high-precision seamless navigation system for light indoor to outdoor positioning using ML-based indoor-outdoor (IO) detection strategies. Many works address the issue of detecting indoor/outdoor environments. In [6] seamless pedestrian positioning and navigation were developed using landmarks detection ML models. An ML-based IO signal transition pattern detection was implemented to determine the signal pattern transition between indoors/outdoors. [7].

The difference between open outdoors and semi-outdoors, light indoors, and deep outdoors is the number of satellites in view. In open outdoor environments, we have at any moment at least four satellites available, which means the localization of the user can be done using GNSS, while in semi-outdoors, light indoors, and deep indoors environments, the number of visible satellites begins to reduce as you progress to deep indoors. Then there is not enough satellite for GNSS localization. The difference between light indoors and deep indoors is the visibility of navigation satellites. In light indoor environments, users may receive navigation signals from several satellites and therefore can be localized using peer-to-peer cooperative positioning, however, in deep indoor environments, no navigation satellites are available. In that case, cooperative positioning fails.

Within the campus of the University of Vaasa, lies Technobothnia the reputable industrial venue having a modern laboratory serving a minimum of five universities and other corporations in the Vaasa region. It has several laboratories for all kinds of technical sciences for example, industrial robots, smart operations, mobile robots, chemistry labs, heavy-duty 3D printing machines, telecommunications equipment, etc. Therefore, visitor traffic is high, and this brings about the need for a seamless outdoor/indoor positioning system that will be beneficial to both human operators and robot assets inside the laboratory. This building has been used as our test indoor environment.

Indoor-outdoor navigation systems can be used to fulfill the following tasks depending on the requirements:

1. location determination;

2. building a route to the desired point, including between buildings within the territory;
3. real-time turn-by-turn navigation;
4. collecting geodata for analytics;
5. sending out push notifications with tips and other information.

In our work, we proposed to use the C/NO value of the GNSS signal and the number of available GNSS satellites to detect the transition between indoor/outdoor environments. By analyzing the C/NO value of the GNSS signal and making use of the available GNSS satellites in view, we can determine the user's environment. We have investigated some machine learning algorithms for classification, including Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), Naive Bayesian (NB), and Neural Network (NN).

The results of the experiment showed that the proposed algorithm was capable of detecting open outdoors, and light-indoor environments with up to 100% accuracy. The hardware and signals required for this are easily accessible and used in our daily lives.

The rest of the article is organized as follows: Section II describes the materials, methods, and strategies used in this research and the essential elements (software and hardware) to build the IO detection model. Section III defines the steps taken during the implementation of the IO detection model. Section IV discusses the output results and provides technical interpretations, evaluations, and comments on the applied performance metrics. In the conclusion section V, the overall achievements are highlighted, and potential future work is presented.

II. MATERIALS, METHODOLOGIES, AND MODELS

In our work, the proposed algorithm is categorized into two main stages: Data Input, ML-based IO detection model training and testing. An overview of the indoor/outdoor detection process is illustrated in Figure 1.

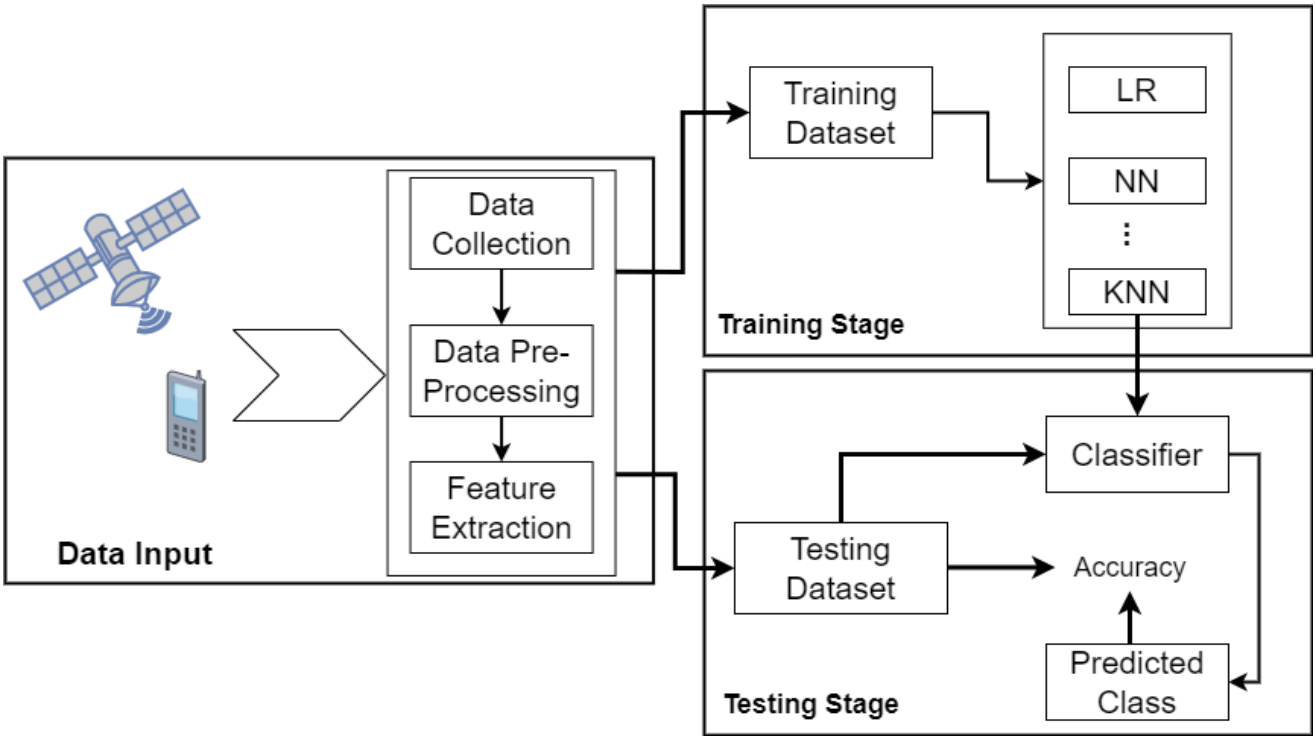


Figure 1: Framework for indoor/outdoor detection. The three processes in this framework include Data Input, Training, and Testing. In the Data Input process, GNSS observation data is recorded, and features are extracted. The data are classified in the Training phase. The classifier is applied to detect the user's current environment in the Testing phase.

1. Data Input

Three sub-processes are included in the data input stage: Data Collection, Pre-Processing, and Feature Extraction.

a). Data Collection

Regarding the format of the collected data, receivers use their own propriety (binary) formats, but programs can be used to convert the data into a standard format called Receiver Independent Exchange Format (RINEX) which can be regarded as the most common GNSS Data file format. The RINEX data can be divided into two, namely RINEX Data header and RINEX Data block.

We used an Android smartphone with an application capable of capturing raw RINEX observables. The smartphone used was the Samsung Galaxy Note20 Ultra. The software used was the GEO++ RINEX Logger.

In this process, the Geo++ RINEX Logger uses the most recent Android API services to log the device's raw GNSS measurement data into a RINEX file. The GNSS data are collected for the test area. The outdoor-indoor positioning is performed in Technobothnia. The UWB (Decawave), Wi-Fi, and IMU (Xsense) data are collected in the Technobothnia building using their respective sensors mounted on a robot (Omron robot).

b). Data pre-processing

The data is then analyzed and pre-processed for use in the IO positioning model. A Python script was developed and used to implement the processing strategy. The recorded GNSS RINEX observation data is processed and parsed using a Python script. The processed data is analyzed before use in the ML-based IO detection model. The features to be extracted are included in the parsed RINEX data.

c). Feature Extraction

In each processed RINEX file, we can extract several features to describe the character of the environment. These features include pseudorange in meters, carrier phase in meters, Doppler, C/N0 in DbHz, and the number of available satellites per epoch.

Pseudorange: Pseudorange is the measured range between the phase centers of the GNSS satellite and receiver antennas, plus the offset between the transmitter and receiver clocks.

Carrier phase: The carrier phase measurement is the measurement of the beat frequency between the satellite signal's received carrier and a reference frequency generated by the receiver. It is the difference between the receiver oscillator phase and the signal received plus the number of cycles at the initial start of tracking.

Doppler shift: In order to be able to receive the signal, the receiver does an estimation of the Doppler shift of each received signal. The time derivative of a signal's carrier phase gives the Doppler shift of the signal. Therefore, the Doppler shift is mainly determined using the relative velocity of the satellite's and receiver's antennas, plus a common offset proportional to the receiver's clock frequency error.

Number of available satellites: The number of available satellites depends on the environment. In the open outdoor environment, no matter whether on water or on a highway, we can localize ourselves using GNSS [8]. In the indoor environment, the number of available satellites is greatly reduced as you go deep indoors. This has a degrading effect on the localization of the user using GNSS, and it becomes impossible in the deep indoors.

2. ML base indoor-outdoor (IO) detection strategy

The correct detection of the IO transition is essential in seamless IO positioning. This enables making the right decision on the navigation mode. The implementation of indoor-outdoor (IO) detection strategies makes use of ML to recognize certain landmarks as well as to find the IO signal transition pattern.

a). Training

In the training stage, we investigate some machine learning algorithms to classify the training data. These ML algorithms include Random Forest (RF), Support Vector Machines (SVM) like support vector classifier (SVC) and Linear SVC, Logistic Regression (LR), Naive Bayesian (NB), and Neural Networks. The classifiers are applied for Indoors/outdoor (IO) recognition. The training data are the raw RINEX data from the GNSS sensors on the smartphone. The best classifier will be selected for indoor/outdoor detection.

b). Testing

The testing phase is used to categorize new samples using the classifiers created in the training phase by the machine learning algorithms. The results show the different performances of the implemented ML algorithms. The performance measures for multi-class classification proposed by researchers have been applied. A confusion matrix was proposed to evaluate the performance of a classification system for the 2-class samples.

To evaluate the IO detection performance, we have made use of the following performance measures:

The testing indoor environment for this experiment is shown in Figure 2. The IO transition is made from outside the building to the section highlighted with green lines for the robot's trajectory inside the building.

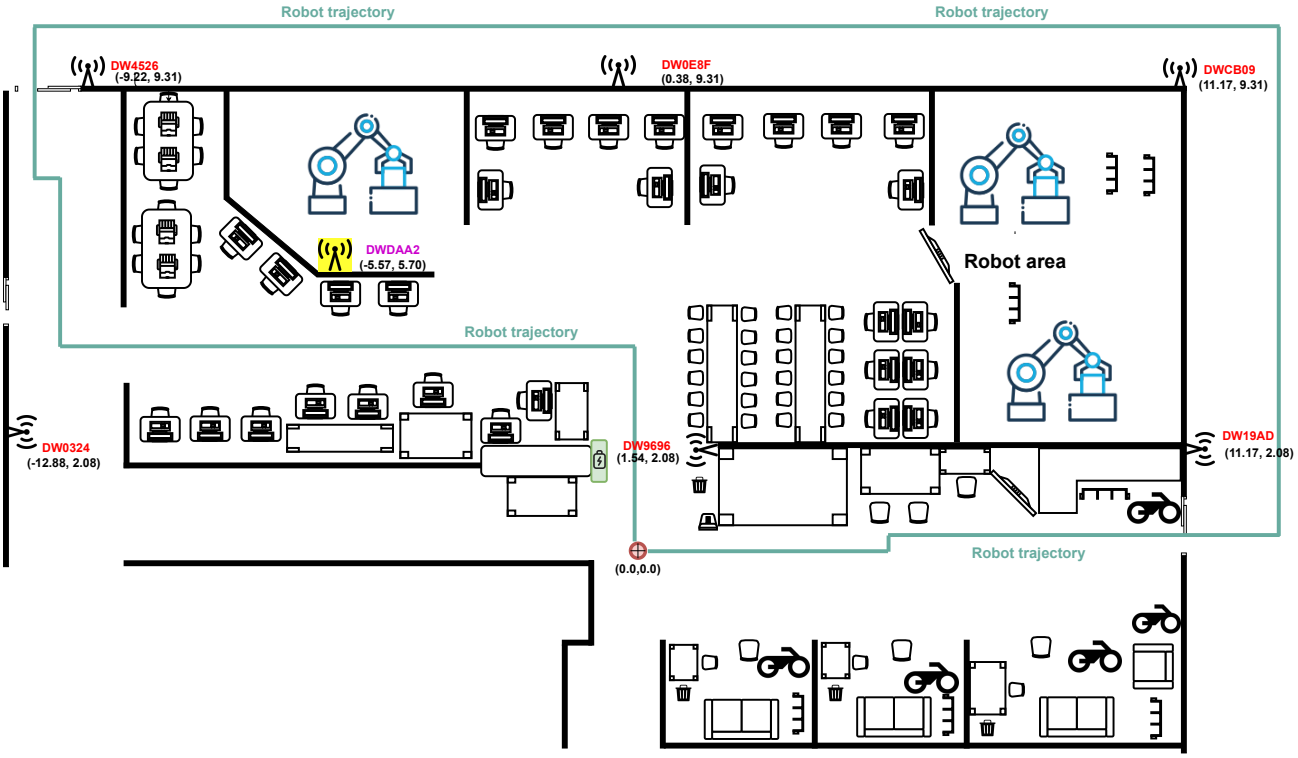


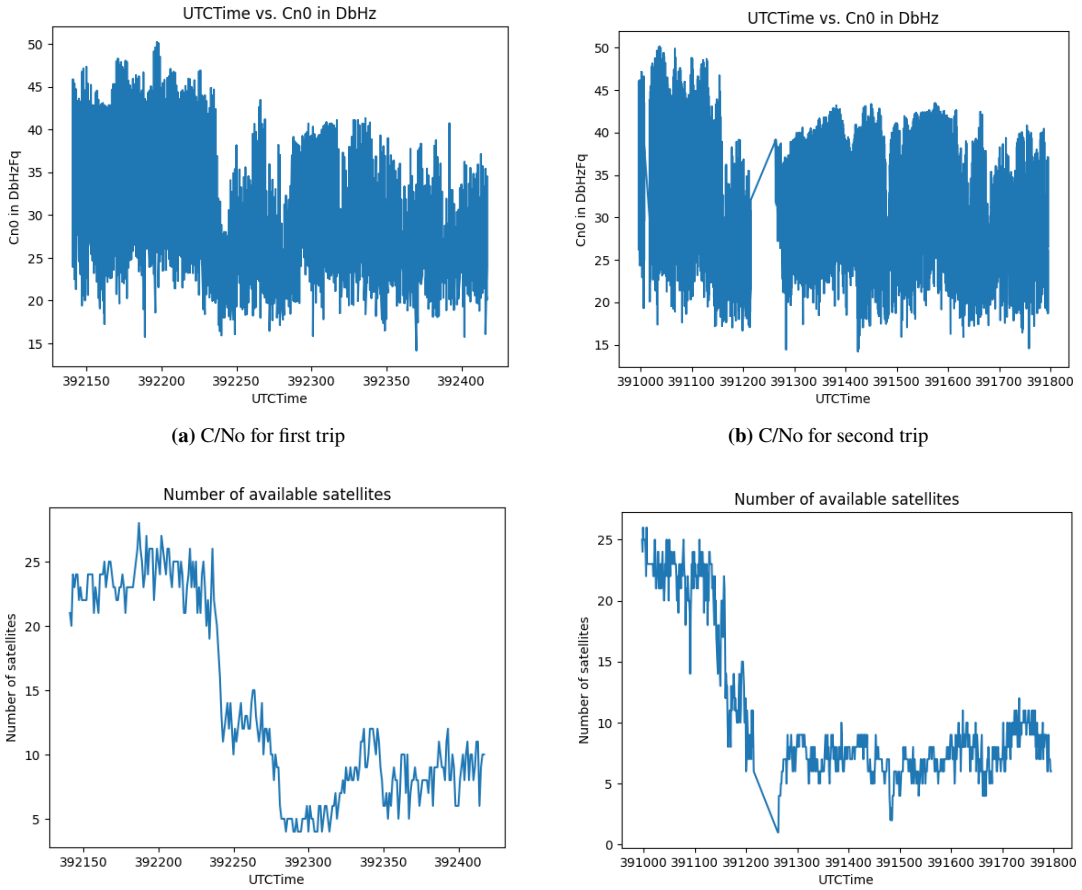
Figure 2: Floor plan of Technobothnia laboratory with the planned robot trajectory.

III. EXPERIMENTS

To test the IO detection algorithm, we make use of separate data (the test data) not shown to the ML-based IO detection models during training. The trained ML classifiers are used to predict the detected environment.

From Figure 3 (c) and (d), we can see the sharp drop in the number of satellites after some UTC time epoch. This drop was noticed during the transition from outdoor to indoor (light-indoor). The number of satellites kept fluctuating during the time when moving around the indoor environment. This is also seen in Figure 3 (a) and (b) for the C/NO values.

The ML IO detection model is trained to detect the transition pattern, and this can be used in the selection of the appropriate positioning technology to apply. With this, the system can switch from the GNSS/IMU positioning method suitable for outdoor positioning to UWB/IMU for indoor positioning, for example.



(a) C/No for first trip

(b) C/No for second trip

(c) Maximum available satellites for first trip

(d) Maximum available satellites for second trip

Figure 3: Comparison of C/NO and number of satellites for two test trips on same navigation path

IV. RESULTS ANALYSIS

Accurate detection of the IO transition would lead to the right decision-making in the navigation mode to be employed in seamless positioning. In our studies, we made use of the ML model for indoor-outdoor (IO) detection using an IO signal transition pattern and the available number of satellites. This approach is more reliable than the use of the signal strength of certain sensors embedded in the location system such as a light sensor, Bluetooth, Wi-Fi, magnetometer, or GNSS signal strength. This is because the solutions based on power-hungry sensors, such as Wi-Fi, could be an issue, especially for mobile devices due to limited battery capacity.

We implemented and tested some machine learning algorithms for classification, including Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), Naive Bayesian (NB), and Neural Network (NN). The ML models were used to train and classify the detected environment into indoor or outdoor.

From Figure 4, we find that all the algorithms have accuracies better than 70%. Logistic regression (LR), SVC, RF, and DT all perform the best, with an average accuracy of 100%. KNN had an accuracy of 99.35%. The NN performed worst with an accuracy of 70.4%. The confusion matrix for the Näive Bayes algorithm is shown in Table 1.

Table 1: Confusion matrix for NB classifier.

Environment	Open Outdoors	Light Indoors
Open Outdoors	75.84%	24.16%
Light Indoors	4.88%	95.12%

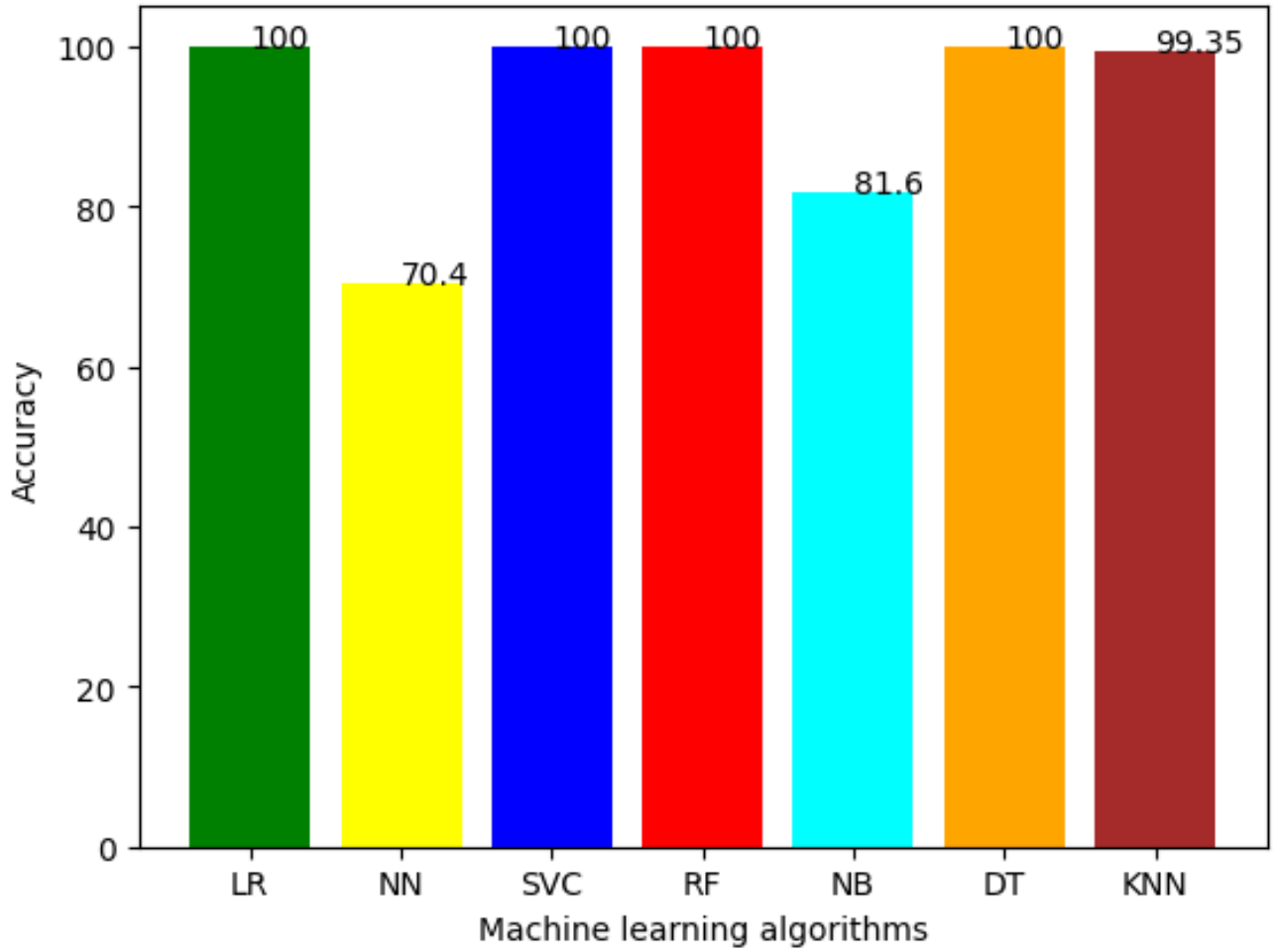


Figure 4: Classification accuracy using different classifiers implemented by different machine learning algorithms.

Table 1 shows that we can detect open outdoor environments correctly. There is a 24.16% possibility of identifying light indoors as open outdoor environments, and a 4.88% possibility of identifying open outdoors as light indoors environments. The LR, SVC, RF, and DT algorithms could distinguish between the open outdoors and light indoors environments more accurately by 100%.

This IO detection method can be useful in the integration of GNSS with UWB/Wi-Fi/IMU systems to achieve seamless navigation for indoor-outdoor positioning. The integration of GNSS with UWB/Wi-Fi/IMU systems can be done using different integration schemes. The positioning method can be selected between, for example, a GNSS/IMU positioning method suitable for outdoor positioning and UWB/IMU for indoor positioning.

Integration schemes: GNSS measurements are easily disturbed by external influences such as multipath, but integrating GNSS with IMU is beneficial as the advantages and disadvantages of GNSS and IMU complement each other to enable accurate measurements in challenging areas. There are three types of integration schemes, namely loosely coupled, tightly coupled, and ultra-tight coupled integration [9].

V. CONCLUSION

Seamless indoor-outdoor positioning remains a challenging aspect for Positioning, Navigation, and Timing (PNT). It is even more complex with variations in the nature of the outdoor (open skies, urban canyons, etc.) and indoor (semi-indoor, deep indoor, etc.) environment. The past decades have seen a variety of navigation techniques being proposed to provide accurate positioning in different environments. Much ongoing research still concentrates on the separate environments as either outdoor

or indoor navigation, thereby, lacking to address the difficulties of seamless positioning.

This research seeks to investigate and implement a seamless IO positioning by correct detection of the IO transition and integration of GNSS and indoor technologies (UWB, Wi-Fi, etc.) to provide accurate positioning when transitioning from indoor to outdoor environments. Our implementation of indoor-outdoor (IO) detection strategies makes use of ML to recognize the IO signal transition pattern. We evaluated the IO detection performance and presented the results.

In the future, we propose to continue the process by using the developed IO detection method to perform different integration schemes to combine GNSS with UWB, Wi-Fi, and IMU in one system. We will use the time latency of the transition detection as an important criterion. When the condition is favorable for GNSS (open outdoor), the output positions of the GNSS receiver are used as inputs to the Extended Kalman Filter for GNSS/IMU integration, and when GNSS is degraded (indoors) the system will switch to UWB/Wi-Fi/IMU integration.

REFERENCES

[1] S. Woo, S. Jeong, E. Mok, L. Xia, C. Choi, and M. Pyeon, "J. heo application of wifi-based indoor positioning system for labor tracking at construction sites a case study in guangzhou mtr, 20," 2011.

[2] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. A. Al-Ammar, and H. S. Al-Khalifa, "Ultra wideband indoor positioning technologies: Analysis and recent advances," *Sensors*, vol. 16, no. 5, p. 707, 2016.

[3] G. Shi and Y. Ming, "Survey of indoor positioning systems based on ultra-wideband (uwb) technology," in *Wireless Communications, Networking and Applications: Proceedings of WCNA 2014*. Springer, 2016, pp. 1269–1278.

[4] R. Faragher and R. Harle, "Location fingerprinting with bluetooth low energy beacons," *IEEE journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2418–2428, 2015.

[5] K. Li, C. Wang, S. Huang, G. Liang, X. Wu, and Y. Liao, "Self-positioning for uav indoor navigation based on 3d laser scanner, uwb and ins," in *2016 IEEE International Conference on Information and Automation (ICIA)*. IEEE, 2016, pp. 498–503.

[6] A. Basiri, P. Amirian, A. Winstanley, S. Marsh, T. Moore, and G. Gales, "Seamless pedestrian positioning and navigation using landmarks," *The Journal of Navigation*, vol. 69, no. 1, pp. 24–40, 2016.

[7] I. Saffar, M. L. A. Morel, K. D. Singh, and C. Viho, "Machine learning with partially labeled data for indoor outdoor detection," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2019, pp. 1–8.

[8] K. Elliott D. and H. Christopher J., *Understanding GPS/GNSS: Principles and Applications*. Artech House, 2017.

[9] Cahyadi, Mokhamad Nur and Rwabudandi, Irene, "Integration of GNSS-IMU for increasing the observation accuracy in condensed areas (infrastructure and forest canopies)," *E3S Web Conf.*, vol. 94, p. 03015, 2019. [Online]. Available: https://doi.org/10.1051/e3sconf/20199403015

Publication VI

Optimal robust positioning using Factor Graph

Akpojoto Siemuri[1], Elham Ahmadi[1], Mahmoud Elsanhoury[1], Kannan Selvan[1], Petri Välisuo[1], Heidi Kuusniemi[1,2], Mohammed S. Elmusrati[1]

[1] *School of Technology and Innovations, University of Vaasa, Finland*

[2] *Finnish Geospatial Research Institute, National Land Survey, Finland*

BIOGRAPHY

Akpojoto Siemuri received his B.Sc.(tech) degree in electrical and computer engineering from the Federal University of Technology Minna, Nigeria in 2010, an M.Sc.(tech) degree in wireless industrial automation at the University of Vaasa, Finland in 2019. He is currently a Project Researcher and pursuing a Ph.D. degree in automation technology at the University of Vaasa. His research interest includes machine learning, GNSS technologies, smart devices, embedded systems, and game theory.

Elham Ahmadi received her M.Sc degree in electrical and electronics engineering from the Shiraz University of Technology (SUTCH), Iran, in 2017, and Ph.D. degree in automation and system engineering from the Federal University of Santa Catarina (UFSC), Brazil in 2023. She is currently a Postdoctoral researcher at the University of Vaasa. Her research interests include mathematical optimization, estimation and data fusion, control theory, and machine learning with applications to autonomous systems.

Mahmoud Elsanhoury is currently pursuing a doctoral (Ph.D.) degree in computer science and telecommunications engineering at the University of Vaasa, Finland. He received his M.Sc. (tech) degree in telecommunications engineering from Vaasa University in 2018, and his B.Sc. degree in telecommunications engineering from Alexandria University, Egypt in 2013. His current research interests cover ubiquitous indoor positioning systems, ultra-wideband (UWB) indoor localization, low-earth orbit (LEO) satellites, multisensor fusion technologies, Kalman filters, uncertain stochastic processes, and machine learning algorithms.

Kannan Selvan received his B.Sc.(tech) degree in Electronics and Communication Engineering from Anna University, India in 2012, the M.Sc.(tech) degree in Communications and Systems Engineering from the University of Vaasa, Finland in 2020. He is currently a Project Researcher at the University of Vaasa. His research interest includes GNSS technologies, satellite-data analysis, machine learning, satellite communication, smart devices, and embedded Systems.

Petri Välisuo is currently working as an Associate Professor (tenure track), in sustainable automation, at the School of Technology and Innovations, University of Vaasa, Finland. He received M.Sc.(tech) degree in computer science from the Tampere University of Technology, Finland, and a D.Sc.(Tech) degree in automation technology from the University of Vaasa, in years 1996 and 2011 respectively. He has authored and co-authored 40 peer-reviewed publications. His research interests cover machine learning, IoT, positioning methods, and other technologies relevant to industrial automation. He has been working for 10 years in the telecommunication industry before his research career at the University of Vaasa.

Heidi Kuusniemi is a professor of computer science and director of Digital Economy at the University of Vaasa in Finland. She is also a part-time research professor in satellite navigation at the Finnish Geospatial Research Institute. She has an M.Sc. (Tech.) degree (with distinction) from 2002 and a D.Sc. (Tech.) degree from 2005 in information technology, respectively, from Tampere University of Technology, Finland. She is the President of the Nordic Institute of Navigation. Her technical expertise and interests include GNSS reliability and resilience, estimation and data fusion, mobile precision positioning, indoor localization, and PNT in new space.

Mohammed S. Elmusrati received his B.Sc. (with honors) and M.Sc. (with high honors) degrees in electrical and electronic engineering, from the University of Benghazi, Libya, in 1991 and 1995, respectively, and the Licentiate of Science in Technology (with distinction) and the Doctor of Science in Technology (D.Sc.) degrees in automation and control engineering from Aalto University Finland, in 2002 and 2004, respectively. Currently, he is a Full Professor and Head of the Cyber-Physical Systems Research Group at the School of Technology and Innovations, University of Vaasa, Finland. His research interest includes wireless communications, artificial intelligence, machine learning, biotechnology, big data analysis, stochastic systems, and game theory. Elmusrati has published over 130 papers, books, and book chapters. He is an active member of different scientific societies - a Senior Member at IEEE, a Member of the Society of Industrial and Applied Mathematics (SIAM), and a Member of the Finnish Automation Society.

ABSTRACT

Smartphone GNSS positioning accuracy is often limited by the use of low-cost antennas and chips, resulting in significant impact of signal degradations due to non-line-of-sight (NLOS) and multipath effects. Traditional techniques, such as extended Kalman filtering (EKF), often struggle to maintain accuracy, especially in urban settings. However, factor graph optimization (FGO) has emerged as a promising solution, demonstrating robust navigation capabilities in urban environments by leveraging multi-epoch GNSS measurements to estimate user positions concurrently. Therefore, we will implement an optimal robust positioning solution using FGO framework in this study. The integration of the Inertial Measurement Unit (IMU) and Global Navigation Satellite System (GNSS) measurements is achieved through an approach that employs the factor graph model. Unlike traditional filtering algorithms, the factor graph framework leverages all preceding measurements for state estimation. This study conducts experiments utilizing the Google Smartphone Decimeter Challenge 2023 dataset as its basis. This research aims to develop an optimal and resilient positioning algorithm by integrating GNSS and IMU through a loosely coupled approach leveraging FGO formulation with directly combining position data from GNSS receivers with the IMU measurements. This method of integration is easily implementable in both hardware and software. We compare this to the use of Kalman Filter/RTS for the same integration process. The Kalman Filter/RTS results gave an accuracy of 2.603 m while the accuracy of FGO implementation was 1.661 m. This research achieved an approximately 56.71% increase in accuracy with the implemented FGO approach.

Keywords - Factor graph optimization; GNSS positioning; IMU measurements; PNT.

I. INTRODUCTION

Global Navigation Satellite System (GNSS) is the most commonly used method for Positioning, Navigation and Timing (PNT) in outdoor environments. However, it is limited and has well-known vulnerabilities when exposed to constrained environments such as urban canyons and indoor environments due to phenomena such as multipath, Non-Line-of-Sight (NLOS) reception as well as signal blockage. Some of these errors and vulnerabilities can be addressed using techniques that help monitor the GNSS measurement quality, such as Fault Detection and Exclusion (FDE). Some GNSS system errors can be directly removed by applying more accurate corrections, such as in Precise Point Positioning (PPP).

GNSS measurements are often disrupted by external factors such as multipath, but integrating GNSS with IMU is advantageous because the strengths and weaknesses of each technology complement one another, facilitating accurate solutions even in challenging environments. Several techniques are being used to improve position estimation robustness when integrating GNSS with Inertial Measurement Units (IMU). Combining these technologies is usually accomplished using extended Kalman filtering (EKF), however, some other studies have applied factor graph optimization (FGO) for the positioning estimation [1], [2], [3]. However, in [4], FGO was used to improve the position estimation without integration with IMU.

This study aims to create an optimal and robust positioning algorithm for integrating GNSS and IMU using an approach based on FGO framework. We utilized loosely coupled integration with a FGO-based integration methods to combine GNSS and IMU into a single system. We used the acceleration and gyroscope data from the IMU for inertial navigation calculations to determine position data, which was then integrated with GNSS information.

II. RELATED RESEARCH

Precise point positioning (PPP) using carrier phase measurements has been extensively researched in the GNSS field. In [5], Watson implements and compares PPP using carrier phase measurements with graph optimization against a Kalman filter-based approach. The integration of RTK position estimation with IMU and other sensors has been investigated in [6, 7], demonstrating improved positioning accuracy over least-squares-based methods. GNSS position estimation is highly influenced by the operating environment of the GNSS receiver, such as open sky or urban canyons. In [4], Suzuki considers the data collection area during position estimation to account for these environmental effects. In [2], the paper proposes an ARIMA auxiliary model to predict GNSS measurements during interruptions, filling data gaps. Additionally, INS and GNSS measurements are loosely coupled using a factor graph model, which utilizes all previous measurements for state estimation. This method provides higher precision and continuous navigation results, even with GNSS data interruptions, compared to traditional KF/EKF GNSS/INS integration.

Based on the analysis of existing research, the main differences between the factor graph algorithm and the EKF algorithm are as follows:

1. The factor graph algorithm connects all historical information through the INS factor, leveraging the time correlation of measurements to resist outliers. This enhances the positioning accuracy and robustness of the system. In contrast, the EKF algorithm only considers the current measurement and the previous estimated value, lacking high robustness.
2. The factor graph can integrate non-synchronized sensor information, which is crucial for multi-source system expansion. Conversely, the EKF requires synchronized information; otherwise, the system needs reconfiguration.

3. In terms of real-time navigation, the EKF algorithm outperforms the factor graph algorithm because global optimization in factor graphs is more time-consuming. Consequently, improving the real-time performance of factor graphs is an emerging research focus.

III. MATERIALS, METHODOLOGIES, AND MODELS

This work is part of the Google Smartphone Decimeter Challenge (GSDC) 2023-2024. The goal of this competition is to determine the limit of smartphone GNSS positioning accuracy: that could be down to the decimeter or even centimeter level. In our work, we will make use of the post-processed kinematic (PPK) precise positioning techniques to process the GNSS datasets in an attempt to achieve sub-meter-level accuracy. GNSS/IMU-based localization is then applied to get the position estimation.

The approach we adopted for integrating GNSS/IMU within FGO is outlined as follows:

1. Data collection, and analysis

The data used is from the Google Smartphone Decimeter Challenge 2023 [8]. The data consist of a pool of raw GNSS measurements from Android smartphones along with high accuracy ground truth: 196 different drives over tens of different routes, with raw GNSS measurements including carrier phase and IMU data. The goal of this challenge is to determine the limit of smartphone GNSS positioning accuracy - performance down to the decimeter or even centimeter level.

2. GNSS data processing using PPK technique

The GNSS data is processed using the post-process kinematic (PPK) technique. In the PPK technique, carrier phase data is used to compute the user position. As PPK requires differential corrections, raw observation data for the appropriate dates and times were downloaded from a nearby CORS station on the NOAA National Geodetic Survey website using a base observation. The satellite broadcast navigation data (BRDM files) for each data set was also downloaded from the International GNSS Service website, including the clock navigation data, and satellite precise orbits. A python script is used to implement RTKLIBS programs [9] to post-process the GNSS data using the PPK technique.

3. Factor graph GNSS/IMU-based localization

The Inertial Measurement Unit (IMU) is frequently used in robotics, vehicles, and, of course, also in mobile phones. An IMU is usually made up of an accelerometer, gyroscope, and also a magnetometer. A tactical grade IMU is fully calibrated, whereas low-cost ones typically come with factory calibrations stored in the IMU registers. The calibration only covers the scaling factors of each axis. The accelerometer measures the specific force that can be derived into the acceleration of a movement, the gyroscope measures the angular speed of the sensor, and the magnetometer measures the Earth's local magnetic field direction and magnitude. The integration of GNSS/IMU using factor graph optimization has recently attracted attention as an alternative to the extended Kalman filter (EKF) for GNSS-IMU integration [1].

The recently introduced formulation of Factor Graph Optimization (FGO) by Indelman et al. [10] has introduced a fresh perspective on multisensor integration, as highlighted by Chen & Gao [11], and Pfeifer & Protzel [12]. This approach is structured as a probabilistic graphical model, featuring nodes representing various system states ($x_i \in \mathbf{X}$) and factors ($f_i \in \mathbf{F}$) corresponding to measurements. The flow chart of the method proposed in this paper is shown in Figure 1

FGO provides a flexible framework for integrating various sensor data measurements ($z_i \in \mathbf{Z}$) through the factorization of the probability distribution, representing the joint probability distribution of all states and measurements as a product of factors, each relating a small subset of variables [10],

$$f(\mathbf{X}) = \prod_i f_i(\mathbf{X}_i), \tag{1}$$

where $\mathbf{X}_i$ be the set of all variables x_i that are connected to the factor f_i by an edge. In terms of non-linear least squares optimization, each factor encapsulates an error function aimed at minimization. The optimal estimate $\mathbf{X}^*$ is determined by minimizing the overall error across the entire factor graph,

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \sum_i \|e_i(\mathbf{X}_i, \mathbf{Z}_i)\|^2_{\Sigma_i}, \tag{2}$$

where Σ is the known covariance of the measurement vector.

In FGO for loosely coupled GNSS/IMU integration, we assume that measurements follow a Gaussian distribution. Considering two types of factors used in the loosely coupled GNSS/IMU integration with FGO, the optimal state set $\mathbf{X}^*$ is obtained by

solving the following optimization problem:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \sum_i \|e_i^{\text{GNSS}}\|^2_{\Sigma_i^{\text{GNSS}}} + \sum_i \|e_i^{\text{IMU}}\|^2_{\Sigma_i^{\text{IMU}}}, \tag{3}$$

where $\|e\|_\Sigma = e \cdot \Sigma^{-1} \cdot e$ is the Mahalanobis norm of e weighted by covariance Σ, $e_i = z_i - h(x_i)$ is the difference between the received measurement and the measurement expected for the true state, and Σ^{GNSS} and Σ^{IMU} are the covariance matrix for the GNSS and IMU factors, respectively. To solve optimization problem (3), a batch solver like Gauss-Newton [13], Levenberg-Marquardt [14, 15], or an incremental one like provided by iSAM2 [16] can be used.

4. Levenberg–Marquardt (LM) algorithm

The Levenberg-Marquardt algorithm combines gradient descent and Gauss-Newton numerical minimization algorithms. Gradient descent reduces the sum of squared errors by adjusting coefficients in the direction of steepest descent. Meanwhile, Gauss-Newton reduces these errors by approximating the least squares function as locally quadratic in coefficients and finding its minimum. The Levenberg-Marquardt algorithm behaves akin to gradient descent when coefficients are far from optimal, and shifts towards Gauss-Newton when coefficients are close to their optimal values [17].

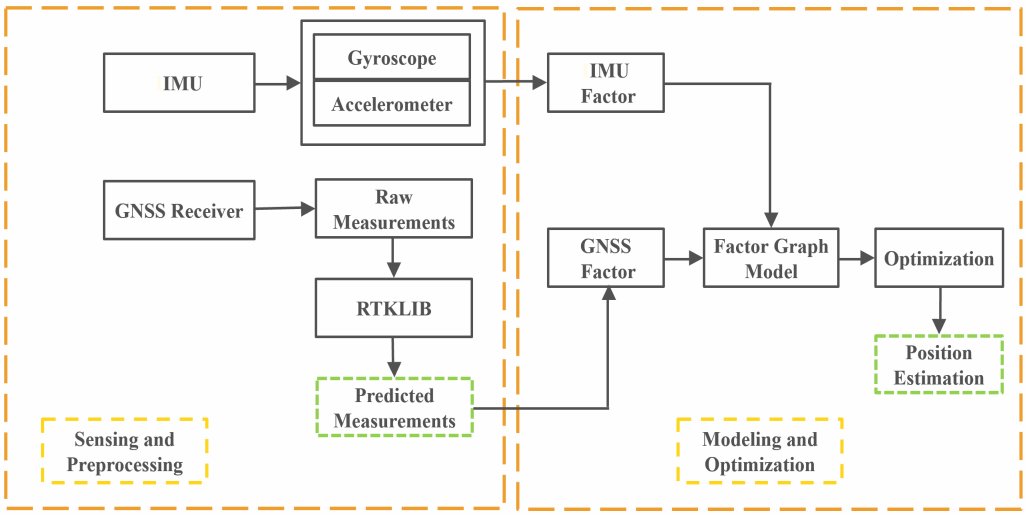


Figure 1: Flowchart of the proposed FGO-based GNSS/IMU integration.

Unlike the traditional EKF-based integration, the FGO captures the posterior probability of states over time, integrating both past measurements and system updates to optimize the entire state set. Historical information plays a crucial role in FGO, where all measurements and states are encoded into a factor graph, and the sensor fusion problem is iteratively tackled through optimization using non-linear optimization method. Consequently, errors arising from linearization steps are minimized. Additionally, FGO adeptly handles delayed measurements by seamlessly incorporating them as supplementary factors into the FGO upon reception. It should be noted that we performed all experiments with the GTSAM framework [18], in which we used the Levenberg-Marquardt algorithm to solve the problem (3). In addition, we used the IMU preintegration scheme as presented in [19].

IV. RESULTS

Accurate position estimation is important in several GNSS applications. Recent studies have focused on the comparison of using EKF-based and FGO-based methods for GNSS/IMU integration, with the focus on the potential of the latter. This research shows that the latter approach is more reliable as it helps to reduce the chance of a decrease in position accuracy due to missing GNSS data points and its ability to capture the posterior probability of states over time, integrating both past measurements and system updates to optimize the entire state set. Filling up the missing GNSS data can also enhance combined navigation, ensuring uninterrupted position estimation even during GNSS device degradation or outages.

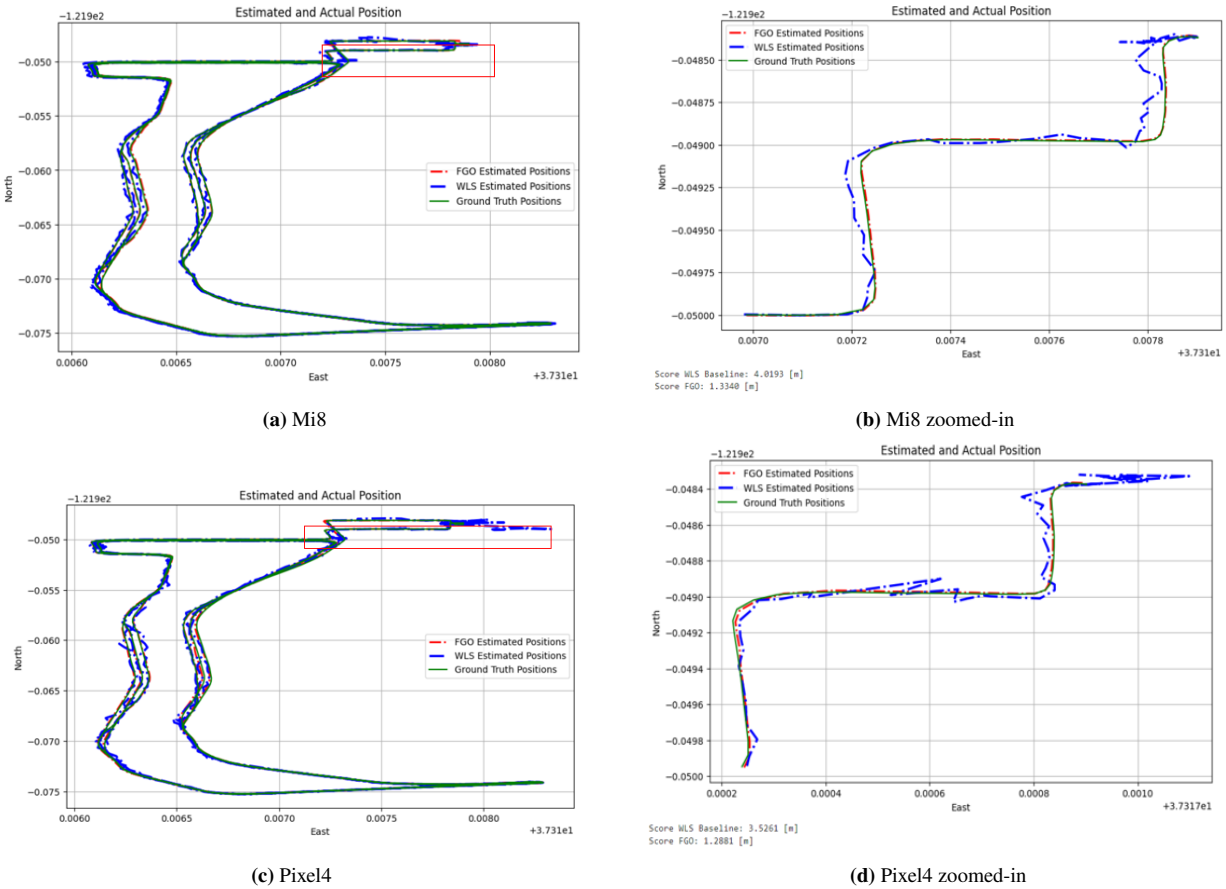
When comparing the WLS position estimation with the implemented FGO-based GNSS/IMU integration using the Vincenty distance from the ground truth in the training dataset, we observed a significant improvement in the accuracy score. In table

1, we present the accuracy score comparison between the WLS baseline score and the FGO-based GNSS/IMU integration for some selected phones and routes. It was however noticed that at some drive routes, the Samsung phones performed very poorly compared to other phones. The best position estimation accuracy when comparing phones were the Pixel phones and the Pixel6pro to be specific. However, when estimating the percentage increase in accuracy, the Mi8 phone has a percentage increase in accuracy of approximately 65.46%.

Table 1: Position estimation accuracy

Phone	WLS Baseline [m]	FGO [m]
Pixel4	2.94	1.47
Pixel4xl	3.64	1.36
Pixel5	5.03	1.48
Pixel6pro	3.85	0.99
Pixel7pro	5.39	1.46
Samsung	4.43	1.96
Mi8	5.99	2.07

An interesting route 2020-12-10-22-52-us-ca-sjc-c in the train dataset had four phones on the same drive route to process. The figure 2 shows the plot of Mi8 and Pixel4 for ground-truth, WLS and FGO plots.



(a) Mi8

(b) Mi8 zoomed-in

(c) Pixel4

(d) Pixel4 zoomed-in

Figure 2: Comparison of positioning error between the selected phones baseline and proposed FGO-based method for particular driving path.

When the FGO-based method of integration was compared with extended Kalman filtering (KF-RTS) for the same integration process [20], the KF-RTS-based results gave an accuracy of 2.291 m while the accuracy of FGO-based implementation in this research was 1.977 m when evaluated on www.kaggle.com for the GSDC competition. There was an 15.88% increase in the accuracy.

V. CONCLUSION

The Global Navigation Satellite System (GNSS) is the preferred technology for outdoor positioning and navigation. However, it faces significant limitations and vulnerabilities in constrained environments such as urban canyons and indoor spaces, where phenomena like multipath, Non-Line-of-Sight (NLOS) reception, and signal blockage are prevalent. Techniques such as Fault Detection and Exclusion (FDE) and Precise Point Positioning (PPP) can mitigate some of these errors, but urban environments remain among the most challenging for accurate GNSS performance.

Integrating GNSS data with Inertial Measurement Unit (IMU) technology can significantly improve position estimation performance. While the Extended Kalman Filter (EKF) is a common approach for this integration, factor graph optimization (FGO) methods have also shown promise. Moreover, FGO has been independently utilized to enhance position estimation. This study aimed to develop an optimal and robust positioning algorithm for GNSS/IMU integration using a loosely coupled approach. This approach directly combines position data from GNSS receivers with IMU measurements, facilitating easy implementation in both hardware and software.

The contributions of this paper include the implementation of a factor graph optimization (FGO)-based method for integrating GNSS position estimation with IMU measurements using the LC method. By adopting this approach, we achieved improved position estimation accuracy compared to our previous studies that utilized the Kalman Filter/RTS for GNSS/IMU integration. The Kalman Filter/RTS results gave an accuracy of 2.603 m while the accuracy of FGO implementation was 1.661 m. This research achieved an approximately 56.71% increase in accuracy with the implemented FGO approach.

In our future study, we plan to do the following, development of robust FGO framework for tightly-coupled (TC) GNSS/INS integration, fusing new sensors to the same framework, reinforcement learning based robust FGO where we will perform optimized robust positioning using Factor Graph (FG) in combination with threat detection and apply adaptive positioning strategies using Reinforcement Learning (RL).

REFERENCES

[1] W. Wen, T. Pfeifer, X. Bai, and L.-T. Hsu, "Factor graph optimization for gnss/ins integration: A comparison with the extended kalman filter," *NAVIGATION: Journal of the Institute of Navigation*, vol. 68, no. 2, pp. 315–331, 2021. [Online]. Available: https://navi.ion.org/content/68/2/315

[2] Q. Li, L. Zhang, and X. Wang, "Loosely coupled gnss/ins integration based on factor graph and aided by arima model," *IEEE Sensors Journal*, vol. 21, no. 21, pp. 24 379–24 387, 2021.

[3] W. Wen and L.-T. Hsu, "Towards Robust GNSS Positioning and Real-time Kinematic Using Factor Graph Optimization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 5884–5890, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/document/9562037

[4] T. Suzuki, "First place award winner of the smartphone decimeter challenge: Global optimization of position and velocity by factor graph optimization," in *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+2021)*, September 2021, pp. 2974–2985.

[5] R. M. Watson and J. N. Gross, "Evaluation of kinematic precise point positioning convergence with an incremental graph optimizer," in *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2018, p. 589–596.

[6] D. Chen and G. X. Gao, "Probabilistic graphical fusion of lidar, gps, and 3d building maps for urban uav navigation," in *Navigation, Journal of the Institute of Navigation*, vol. 66, 2019, p. 151–168.

[7] W. Li, X. Cui, and M. Lu, "A robust graph optimization realization of tightly coupled gnss/ins integrated navigation system for urban vehicles," in *Tsinghua Science and Technology*, vol. 23, no. 6, 2018, p. 724–732.

[8] G. M. Fu, M. Khider, and F. van Diggelen, "Android raw gnss measurement datasets for precise positioning," in *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+2020)*, September 2020, pp. 1925–1937.

[9] T. Everett. (2021) Batch processing rtklib solutions with rnx2rtkp and python. [Online]. Available: https://rtklibexplorer.wordpress.com/2022/01/05/batch-processing-rtklib-solutions-with-rnx2rtkp-and-python/

[10] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Factor graph based incremental smoothing in inertial navigation systems," in *2012 15th International Conference on Information Fusion*. IEEE, 2012, pp. 2154–2161.

[11] D. Chen and G. X. Gao, "Probabilistic graphical fusion of lidar, gps, and 3d building maps for urban uav navigation," *Navigation*, vol. 66, no. 1, pp. 151–168, 2019.

[12] T. Pfeifer and P. Protzel, "Expectation-maximization for adaptive mixture models in graph optimization," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 3151–3157.

[13] Y. Wang, "Gauss–newton method," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 4, pp. 415–420, 2012.

[14] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[15] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[16] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[17] H. P. Gavin, "The levenberg-marquardt algorithm for nonlinear least squares curve-fitting problems," *Department of Civil and Environmental Engineering Duke University August*, vol. 3, 2019.

[18] F. Dellaert and G. Contributors. (2022, May) borglab/gtsam. [Online]. Available: https://github.com/borglab/gtsam)

[19] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, *IMU preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation*. Georgia Tech Borg Lab, 2015.

[20] A. Siemuri, M. Elsanhoury, P. Välisuo, H. Kuusniemi, and M. S. Elmusrati, "Application of machine learning to gnss/imu integration for high precision positioning on smartphones," *Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022), Denver, Colorado, September 2022*, pp. 2256–2264, 2022. [Online]. Available: https://doi.org/10.33012/2022.18375

Publication VII

Machine Learning for LEO and MEO Satellite Orbit Prediction

Kannan Selvan[1], Akpojoto Siemuri[1], Fabricio S. Prol[1,2], Petri Välisuo[1], Heidi Kuusniemi[1,2]
[1] *School of Technology and Innovations, University of Vaasa, Finland*
[2] *Finnish Geospatial Research Institute, National Land Survey, Finland*

BIOGRAPHY

Kannan Selvan received his B.Sc.(tech) degree in Electronics and Communication Engineering from Anna University, India in 2012, the M.Sc.(tech) degree in Communications and Systems Engineering from the University of Vaasa, Finland in 2020. He is currently pursuing a Ph.D. degree in automation technology at the University of Vaasa. From 2018 to 2020, he was a Research Assistant in the Digital Economy Research Platform at the University of Vaasa, Finland. He is currently a Project Researcher at the University of Vaasa. His research interest includes GNSS technologies, LEO-PNT, satellite-data analysis, machine learning, satellite communication, smart devices, and embedded Systems.

Akpojoto Siemuri received his B.Sc.(tech) degree in electrical and computer engineering from the Federal University of Technology Minna, Nigeria in 2010, an M.Sc.(tech) degree in wireless industrial automation with a minor study in industrial management from the University of Vaasa, Finland in 2019. He is currently pursuing a Ph.D. degree in automation technology at the University of Vaasa. From 2018 to 2019, he was a Research Assistant in the Smart Energy Systems Research Platform (SESP) Project at the University of Vaasa, Finland. He is currently a Project Researcher at the University of Vaasa. His research interest includes machine learning, GNSS technologies, smart devices, embedded systems, communication systems, and game theory.

Fabricio dos Santos Prol is a Senior Researcher at the Finnish Geospatial Research Institute (FGI) in the National Land Survey of Finland (NLS) - Finland. He is also a Docent Fellow in the University of Vaasa. The focus of his current research lies in GNSS positioning, LEO-PNT systems, ionospheric modeling, and data assimilation.

Petri Välisuo is currently working as an Associate Professor (tenure track), in sustainable automation, at the School of Technology and Innovations, University of Vaasa, Finland. He received M.Sc.(tech) degree in computer science from the Tampere University of Technology, Finland, and a D.Sc.(Tech) degree in automation technology from the University of Vaasa, in years 1996 and 2011 respectively. He has authored and co-authored 27 peer-reviewed and more than 10 other scientific publications. His research interests cover machine learning, IoT, positioning methods, and other technologies relevant to industrial automation. He has been working for 10 years in the telecommunication industry before his research career at the University of Vaasa.

Heidi Kuusniemi is a professor of computer science and director of Digital Economy at the University of Vaasa in Finland. She is also a part-time research professor in satellite navigation at the Finnish Geospatial Research Institute. She has an M.Sc. (Tech.) degree (with distinction) from 2002 and a D.Sc. (Tech.) degree from 2005 in information technology, respectively, from Tampere University of Technology, Finland. She is the President of the Nordic Institute of Navigation. Her technical expertise and interests include GNSS reliability and resilience, estimation and data fusion, mobile precision positioning, indoor localization, and PNT in new space.

ABSTRACT

Accurate orbit prediction plays a significant role in many space geodesy applications, including space situational awareness, orbital maneuvers, and satellite navigation in real-time scenarios. The traditional orbit prediction approach includes analytical and numerical algorithms to accurately propagate the satellites' state and associated uncertainties. However, these methods are based on limited dynamic models and may have limitations in accurately capturing the complex dynamics of satellite motion. With the rapid growth in computing power and the development of advanced machine learning (ML) and deep learning (DL) algorithms, there has been growing interest in leveraging ML algorithms to enhance orbit prediction accuracy. In this study, we investigate the potential of using different ML algorithms for the orbit prediction of low Earth orbit (LEO) satellites. We also extend our analysis to medium Earth orbit (MEO) satellites. LEO Swarm-A satellite's precise orbit products and MEO GNSS satellites' final ephemeris products are used. The datasets are pre-processed and used in training various ML models. The ML models are then used to estimate the position and velocity of the LEO and MEO satellites. The accuracy of both LEO Swarm-A satellite and MEO GNSS satellites' using various ML models are compared and discussed. Overall, the standalone ML-based method holds significant promise for improving orbit prediction accuracy and reliability for SWARM-A satellite and GNSS satellites, ultimately enhancing the performance and efficiency of space-based applications and services.

Keywords - Orbit prediction; Low-Earth-Orbit; Medium-Earth Orbit; Global Navigation Satellite Systems; Machine Learning; Ephemeris

I. INTRODUCTION

Highly accurate orbit prediction is indispensable for a wide range of applications in space geodesy [1]. With the advancements in ground-based and space-based technology, near-real-time and real-time services have been developed requiring high-accuracy orbit prediction. Mission design, orbit determination, payload data analysis, satellite navigation and positioning, and other applications such as real-time navigation, atmospheric monitoring, and precise point positioning require a high-precision satellite orbit prediction [2], [3], [4], [5], [6].

The traditional method for satellite orbit prediction relies on physics-based models, which include solving the differential equations of motion using either analytical or numerical methods. The numerical method involves integrating the equations of motion to predict the satellite orbit [7]. The main advantage of using the numerical method is that the orbit prediction is highly accurate compared to other methods, being suitable for complex scenarios but computationally demanding. On the other hand, the analytical method provides the position and velocity of the satellite at any point in time [8]. It is a commonly used method for orbit prediction and is computationally efficient. However, the disadvantage of the analytical method is the challenge to derive higher-order solutions, leading to less accuracy in predicting the satellite orbit compared to the numerical methods. In addition, a combination method known as the semi-analytical method involves complex perturbation effects, which are then transferred by the analytical method [9]. Although faster computational results can be obtained compared to the numerical method, a trade-off between accuracy and agility should be analyzed.

Various analytical and numerical method propagators are used to propagate the state of the satellites and its associated uncertainties [10]. Multiple sources of perturbing forces such as Earth's non-uniform gravitational field, solar radiation pressure, and attraction of the sun and moon are taken into account to various extents [11]. The simplified general perturbations (SGP4) algorithm is an analytical propagator which uses the two-line element (TLE) files to generate the ephemeris. TLE includes a set of mean orbital elements that defines position and velocity of the satellite at a reference epoch without needing to calculate each intervening state [12]. Higher order terms are omitted in SGP4 due to the intricacy of the equations of motion of orbital mechanics. Therefore, with analytical methods purely based on limited dynamic models, it is not suitable for all positioning, navigation and timing (PNT) applications requiring high accuracy [11]. On the other hand, high-precision orbit propagators (HPOP) are numerical propagators incorporating detailed force models, taking into account the higher-order terms that are omitted by SGP4. Though the numerical integration technique yield orbit predictions accurate to within meters, it requires the computation of the object's state vector for each time step increment. Thus, numerical methods require large quantities of accurate data and computation capabilities, which are often not existing for many satellites. [13] Therefore, it is challenging to establish an accurate physics-based model using analytical and numerical methods and predict the satellite orbit with high accuracy.

Considering the rapid growth in computing power, advanced machine learning (ML) algorithms can be used for predicting the LEO and MEO satellites' orbit. ML offers new modeling strategies for orbit prediction methods. ML models learn the underlying pattern based on a large amount of data and predict future events [14]. This data-driven approach enables ML models to capture dynamic and unstable environments, handle nonlinear relationships, and improve accuracy by identifying correlations that are difficult to model using the conventional techniques. The ML approach has been beneficial in a wide range of applications [15, 16] and especially in the aerospace field [17, 18, 19, 20].

In this paper, we aim to apply ML techniques for the orbital accuracy of LEO satellites. A simple yet efficient method for modeling the relationship between input variables are provided by utilizing ML model. Based on the training of various ML

models using historical satellite orbit data, we develop a predictive model capable of estimating the positions and velocities of the LEO satellite, thereby reducing the root-mean-square error (RMSE) of orbit predictions. Therefore, the primary objective is to assess the feasibility and efficacy of utilizing different ML techniques for orbit predictions of LEO satellites and improving the prediction accuracy quantified by RMSE. To evaluate the robustness and versatility of our proposed method using different ML models, we also extend our analysis to MEO satellites and highlight any potential differences between LEO and MEO satellite predictions. We implemented various ML algorithms such as KNeighbors Regressor (KNR), Random Forests Regressor (RFR), linear regression (LR), and polynomial regression (PR), and aimed to demonstrate the effectiveness of different ML models and compare their performances.

The remaining parts of this article are organized as follows. In Section II, the related works are presented with respect to the orbit prediction carried out using various approaches. Section III presents the strategy used in this research work to assess the ML models for orbit prediction of LEO and MEO satellites. In Section IV, detailed information on the data preprocessing and preparation is provided. Section V discusses the ML models implemented to improve the orbit prediction accuracy, while section VI presents the results and evaluation of the ML models for both the LEO and MEO satellites. In Section VII, conclusions are summarized, and future research are suggested.

II. RELATED WORKS

From the existing studies, different approaches have been implemented to improve the orbit prediction accuracy. In [13], the authors proposed a batch least-squares method, in which differential corrections are applied to objects in the two-line element (TLE) catalog. Using a high-precision numerical propagator, the orbit is fitted to the state vectors obtained from consecutive TLEs. The predicted range error increases at a typical rate of 100 m per day. In [5], the authors proposed a method to improve the computational efficiency of numerical propagators for orbit propagation of Molniya satellite while maintaining the required accuracy levels. It aimed towards optimal balance between accuracy and computation by adjusting the number of geopotential spherical harmonics retained during integration, thereby ensuring that the numerical propagator achieves the required accuracy with minimum computational cost. In [21], the authors utilized an analytical orbit model to analyze the precision of orbit prediction for MEO and LEO satellites. The orbit prediction was validated using a high precision laser ephemeris and obtained an accuracy of several hundred meters in a 1-day orbit prediction and not more than 10 km in 7 days. In addition to the traditional methods, [22] introduced a novel approach in orbital modeling by utilizing data-driven techniques to forecast the orbital parameters. Using historical orbital data, the novel predictor system upon training constructs new TLEs near the desired prediction time. The initial results using a simple predictor system demonstrate comparable or better accuracy than traditional SGP4, particularly when predicting two weeks beyond the available data.

In [23], the authors discuss the Earth rotation parameters (ERPs) predicted by different services to analyze their impact on ultra-rapid ephemeris prediction of GNSS. A real-time correction method is proposed to reduce the impact of the ERPs on GNSS ultra-rapid orbit prediction products. Experimental results demonstrated significant accuracy improvements when using the proposed correction method. In [24], the authors introduced an error analysis approach based on historical data and the periodic characteristics of the orbit prediction error. The error fitting is carried out by introducing the Poisson series. In [25], two methods aimed to overcome the challenges caused by the memory and computing time required for orbit prediction procedures by providing an analytical representation of numerical orbits.

Based on ML approaches, the three most common types that have been studied include supervised learning, reinforcement learning, and unsupervised learning [15]. In [26], the authors discuss challenges in achieving accurate orbit prediction due to limited information about space object's trajectories and operating environments. It highlights the shortcomings of physics-based models and the lack of accuracy in TLE-based prediction techniques. The approach for orbit prediction makes use of ML techniques, particularly curve fitting and Long Short-Term Memory (LSTM) models trained using TLE data. In [27], a novel hybrid model was implemented combining the SGP4 with two ML estimators, the autoencoder and random forest. This approach aims to reduce errors in SGP4 propagators caused by incomplete perturbation forces and low-order of series expansions. The applied ML estimators model enabled the correction of the time-series nature of error patterns, resulting in more accurate orbit predictions. The hybrid model, tested on three satellite objects with corresponding TLE data, achieved a 20-30% improvement in orbit prediction accuracy over a 30-day period. In [28], orbit prediction on GNSS constellations in the medium Earth orbit (MEO) using different ML and DL algorithms was analyzed for improving the accuracy of the ultra-rapid products from IGS. Utilizing Long Short-Term Memory and Gated Recurrent Unit, an average improvement of 40% in 3D RMS was obtained within the 24-hour prediction interval of the ultra-rapid products.

In [29], the Support Vector Machine (SVM) was used to improve the satellite orbit prediction accuracy making use of two publicly available data to validate the proposed ML approach, namely the TLE catalog and the International Laser Ranging Service (ILRS) catalog. In most cases, the results showed that the designed dataset structure and SVM model can improve the orbit prediction accuracy with good performance. Further, in [30], three machine learning approaches are investigated to improve the orbit prediction accuracy in a simulation environment, namely SVM, artificial neural network (ANN), and Gaussian

processes (GP). ANN showed better approximation capability compared to SVM and GP.

In addition to the aforementioned studies, numerous other research focused on enhancing the precision of orbital propagation models through the application of ML techniques [31, 32, 33]. These investigations have demonstrated that ML methods can predict accurately while maintaining the orbit propagation model using historical data with limited resources. Nevertheless, each of the techniques have its own strengths and limitations based on factors such as data types, sizes, and dataset behavior.

ML has also been extensively used for the orbit determination and orbit prediction of LEO satellites. LEO satellites have the potential to deliver several benefits over MEO satellites in terms of PNT, as well as location-enabled communications [34]. In [11], a hybrid analytical-ML framework was developed for improved LEO satellite orbit prediction. It includes three stages as follows: 1) initial estimation of LEO satellite's states with SGP4-propagated TLE data and subsequently estimating with extended Kalman filter (EKF) during satellite visibility; 2) utilization of a nonlinear autoregressive with exogenous inputs (NARX) neural network for orbit propagation when the satellite is not in view; and 3) estimation of the terrestrial receiver position using ML-predicted satellite ephemeris and carrier phase measurements. Experimental results demonstrated a significant reduction in ML-predicted ephemeris error by nearly 90% compared to SGP4 propagation. In [35], ML is used for predicting LEO satellites in a simultaneous tracking and navigation (STAN) framework. In this work, a time delay neural network (TDNN) is developed, which is shown to improve the LEO satellite tracking performance over an extended Kalman filter (EKF). The EKF-STAN achieved 3D position RMSE of 71 m and 26 m for the two LEO Orbcomm satellites, while the proposed EKF-TDNN-STAN framework achieved 3D position RMSE of 6 m and 26 m, respectively. In [36], the paper employed HPOP in conjunction with decoded Orbcomm satellite ephemeris messages to train a neural-network (NN) model capable of estimating the satellite's position with meter-level precision in a short time period.

These existing studies implemented various approaches to improve the orbit prediction accuracy. Traditional methods have been employed to predict the satellite orbits with varying degrees of accuracy. Subsequently, ML approach has shown capability compared to the dynamic models. Furthermore, existing studies also implemented a hybrid approach which combines ML and traditional methods, demonstrating significant improvements in the orbit prediction accuracy. However, there remains value in further exploring standalone ML-based method for orbit prediction. In this paper, we utilize various ML models and assess the accuracy of each of the models in predicting the LEO and MEO GNSS satellites. The decision to employ a standalone ML-based method is purely based on the need to provide a simplified and accessible alternative to traditional or hybrid methods while maintaining predicted accuracy. By focusing solely on the ML-based method, this approach aims to alleviate the intricacies associated with traditional and hybrid methodologies, thus facilitating simplified implementation and interpretation. Therefore, while various approaches have demonstrated notable success, the use of ML-based method provide an additional means of addressing orbit prediction challenges with simplicity, efficiency and interpretability. The models will be trained using the precise ephemeris data obtained from the European Space Agency's Swarm Data Access [37] for LEO satellites and from the office of Geomatics of the National Geospatial-Intelligence Agency (NGA) [38] for MEO GPS satellites.

III. STRATEGY

Orbit prediction involves the estimation of future state of motion of an object based on the orbit determined in past observations. Applications of orbit prediction include satellite navigation, orbital maneuvers, and space situational awareness. A set of approximate equations of motion is used to describe the motion of the object. The degree of approximation varies depending on the intended use of orbital information. However, a better orbit prediction can be made with highly accurate dynamic models [39].

In this work, the primary orbit prediction strategy involves implementing ML-based algorithms to estimate the LEO SWARM-A satellite and MEO GPS satellites' orbit. The accuracy of ML models are evaluated in terms of RMSE. By using different ML models such as KNR, RFR, LR and PR models, we aim to demonstrate that a standalone ML-based approach can yield high accuracy in orbit prediction.

The steps involved in orbit prediction are as follows:

1. Data preprocessing and preparation;
2. Proposed ML-based prediction method;
3. Performance comparison of the ML models;
4. Evaluation of the implemented ML models.

IV. DATA PREPROCESSING AND PREPARATION

With respect to LEO, the data used to train the model are the SWARM-A satellite precise ephemeris data in the SP3 format. SWARM-A is one of the three satellites in the ESA's SWARM constellation, operating in a circular near-polar orbit at an altitude

of 460 km [40]. The ephemeris data includes time of observation, position, velocity, and satellite clock parameters. In this study, we used the time series of dynamic position and velocity of the SWARM-A satellite in the International Terrestrial Reference Frame (ITRF) related to the center of mass. The sampling period is 10 seconds, i.e., the positions and velocities are observed for every 10 seconds. The ephemeris data from 02/10/2023 to 31/01/2024 are collected from the ESA portal [37].

In MEO, the data used to train the model is the GNSS satellites' final ephemeris data provided by the International GNSS service (IGS) in the SP3 format. The GNSS satellites' here refers only to the 32 GPS satellites orbiting in different orbital planes. The ephemeris data includes time of observation, position, velocity, and satellite clock parameters. Time series of dynamic position and velocity of GPS satellites were used in the analysis. The sampling period is 5 minutes, i.e., the position and velocity of all 32 GPS satellites are measured every 5 minutes. Data from GPS week 2086 to 2251 were collected from the NGA portal for the analysis [38].

The initial step in the data processing pipeline involves defining the number of days of data to be used by the ML models. We consider different number of days of data such as 1, 3, 10, 30 and 50 days for the analysis of the ML models. In case of MEO, the algorithm processes only one single GPS satellite at a time, so we include GPS PRN number in the processing pipeline. The next step involves reading the time, position and velocity data for SWARM-A satellite or a single GPS satellite. In order to ensure that the data are in a consistent and appropriate coordinate system, the function not only retrieves the necessary satellite tracking data but also converts the time-series position and velocity data from ITRF to the Geocentric Celestial Reference System (GCRS). Following the coordinate transformation, Lagrange interpolation is used to improve the temporal resolution of the data. A polynomial is generated by using the interpolation technique that passes over a group of measured data points and takes certain values randomly, providing a denser and more continuous dataset. For SWARM-A satellite, considering the 10-second sampling interval, 24 hours consist of 8639 actual data points of position and velocity. Upon interpolation process, there are a total of 86390 points as shown in figure 1 for step size of 1 s. In other words, there is an actual data point or interpolated data point for every one second. Similarly, for step size 3, there is an actual data point or interpolated point for every three seconds. The step size defines the prediction time of the satellites. In this analysis, for SWARM-A satellite, different prediction time intervals such as 1, 3, 5, 7, 9 and 10 seconds are also considered in addition to the number of days of data used by the model. The red-colored dots are the actual data points obtained from SP3 files at a certain sampling time (seconds), while black-colored dots indicate the interpolated points, providing denser data points upon interpolation process as seen in the figure 1.

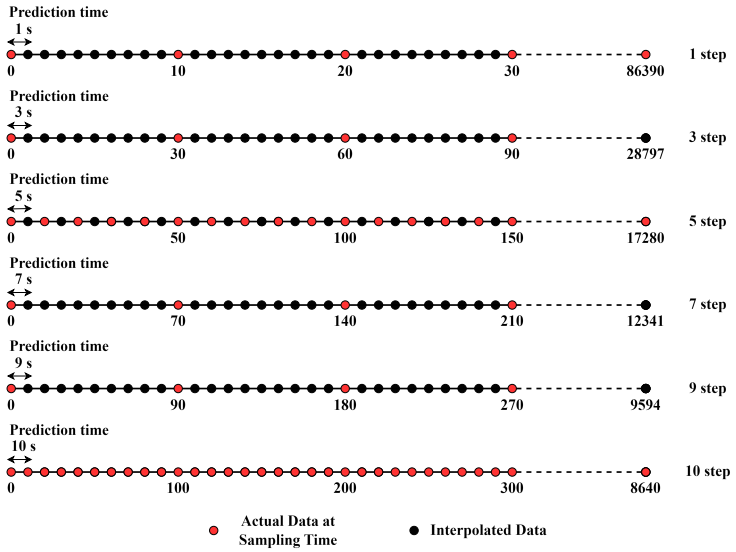


Figure 1: Interpolating points between the observed data points for LEO SWARM-A satellite data using Lagrange interpolation method.

For each GPS satellite, considering a 5-min sampling interval, there are a total of 288 data points in 24 hours. Upon interpolation process, a single satellite selected with 288 data points in 24 hours will have a total of 86100 points for step size of 1 as shown in figure 2. There are 299 interpolated points between every 5-min sampling interval when the step size is 1. Similarly, there are 29 interpolated points between every 5-min sampling interval for a step size of 10. In this analysis, for GPS satellites, the prediction time interval of 10, 20, 30, 40, 50 and 60 seconds were considered in addition to the number of days of data used. Figure 2 shows how the interpolation process works, displaying the actual data for every 5-minutes (300 seconds) with red color and interpolated points with black color.

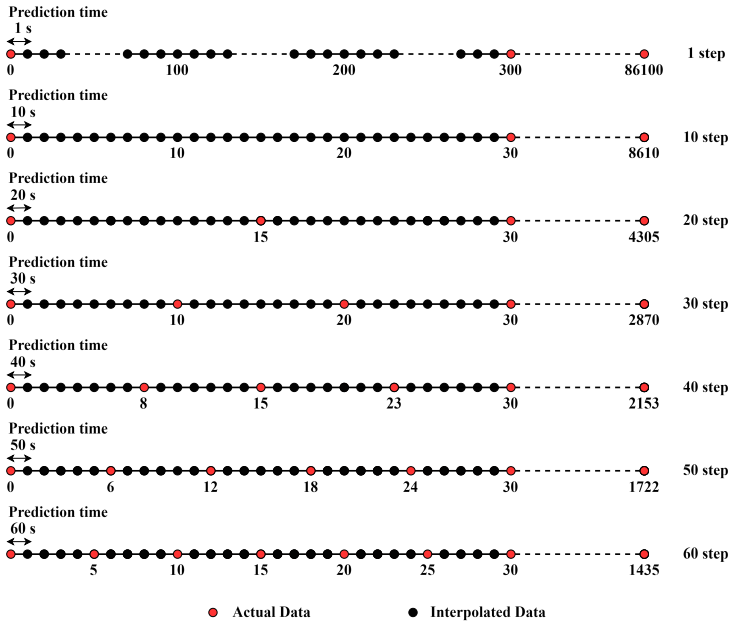


Figure 2: Interpolating points between the observed data points for a single GPS satellite using Lagrange interpolation method.

After the completion of the interpolation process, the dataset now contains the observed data points and interpolated points. Based on this dataset, our developed algorithm is processed to generate the input features and labels to train the ML models. The input features are the historical positions Px, Py, Pz and velocities Vx, Vy, Vz of the SWARM-A and GPS satellites.

For i^{th} timestamp, the position and velocity are given as follows: $Px_i,\ Py_i,\ Pz_i$ and $Vx_i,\ Vy_i,\ Vz_i$

Similarly, the position and velocity for the next timestamp, i.e., the next data point is given by: $Px_{i+1},\ Py_{i+1},\ Pz_{i+1}$ and $Vx_{i+1},\ Vy_{i+1},\ Vz_{i+1}$

The labels are the true position offsets and velocity offsets computed based on the positions and velocities value. True position offset, y_p is given by the difference between the current position data (Px_i, Py_i, Pz_i) and next position data $(Px_{i+1}, Py_{i+1}, Pz_{i+1})$, given as follows:

$$y_p = C_p - N_p \tag{1}$$

where y_p is the true position offsets, C_p is the current position and N_p is the true next position.

Similarly, the true velocity offset, y_v is given by the difference between the current velocity data (Vx_i, Vy_i, Vz_i) and next velocity data point $(Vx_{i+1}, Vy_{i+1}, Vz_{i+1})$, given as follows:

$$y_v = C_v - N_v \tag{2}$$

where y_v is the true velocity offsets, C_v is the current velocity and N_v is the true next velocity.

The true position and velocity offsets calculated as shown in the equations 1 and 2 using the positions and velocities value will be used as the reference data to validate the different ML models. In general, the true offsets are represented using the following equation.

$$y = C - N \tag{3}$$

where y is the true offset which can be either a positive or negative value, C is the current position or velocity, N is the true next position or velocity.

Equation 3 can be rearranged as follows:

$$N = C - y \tag{4}$$

where N is the true next position or velocity, C is the current position or velocity and y is the true position or velocity offset.

Following the generation of the input features and labels, the dataset is now divided into training (70%) and test (30%) subsets. Training the ML model with training data enables the model to learn the underlying pattern from the historical positions and velocities obtained. The predictive performance of the model is then evaluated using the test data. This comprehensive data preparation approach ensures that the ML model is trained using a high-quality dataset.

V. PROPOSED METHOD

1. ML-based Prediction Method

ML models are utilized to estimate the satellite orbits using precise ephemeris data. We implemented different ML models, namely the KNR model, RFR model, LR model and PR model. The models are trained using the input features generated as discussed in the section IV. The trained model then predicts the position and velocity offsets based on the test data fed into the trained model. Similar to equation 3, the predicted position and velocity offsets, in general, are given using the following equation.

$$\hat{y} = C - \hat{N} \tag{5}$$

where $\hat{y}$ is the predicted offset, C is the current position or velocity, $\hat{N}$ is the estimated position or velocity.

From the predicted offsets for the position or velocity, the estimated position or velocity is obtained by rearranging the equation 5:

$$\hat{N} = C - \hat{y} \tag{6}$$

where $\hat{N}$ is the estimated position or velocity, C is the current position or velocity and $\hat{y}$ is the predicted position or velocity offset from the ML model.

The loss function used for evaluating the performance of different ML models is the root-mean-square error (RMSE). RMSE is a common metric used in ML and statistics to measure the accuracy of a predictive model due to its ability to quantify the average magnitude of the prediction error. By minimizing the RMSE, we aim to ensure that the predicted positions and velocities correspond with actual positions and velocities, thereby improving the accuracy of the orbit predictions.

The RMSE is given by the following equation:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(N_i - \hat{N}_i)^2} \tag{7}$$

where n is the number of observations, N_i represents the true values and $\hat{N}_i$ is the predicted values.

The various ML models implemented are as follows:

*a). **KNR Model***

The KNR model is a versatile and widely used non-parametric algorithm that predicts the dependent variable, given by y based on the average of the nearest neighbors in the input space [41]. The model captures the non-linear relationships by considering the structure of the data. The general formula for the KNR model is given by:

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i \tag{8}$$

where the target variables are the predicted position and velocity offsets, given by $\hat{y}$. k represents the number of nearest neighbors considered in the regression and y_i are the target variable value or the outputs of the nearest neighbors. From the predicted position and velocity offsets based on the KNR model, using equation 6, the positions and velocities can be estimated.

b). ***RFR Model***

The RFR model is a commonly used model due to its ability to work well with a large and diverse dataset. An ensemble learning method that combines multiple decision trees is used to make predictions. Based on the prediction of the individual trees, the model captures the non-linear relationships in the data. [42] The general formula for the RFR model is given by:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(x) \tag{9}$$

where the target variables are the predicted position and velocity offsets, given by $\hat{y}$. T represents the number of decision trees considered in the regression, and $f_t(x)$ are the prediction or outputs of the t^{th} decision tree for the input x. From the predicted position and velocity offsets based on the RFR model, using equation 6, the positions and velocities can be estimated.

c). ***LR Model***

The LR model is one of the most widely used modeling technique due to its simplicity and interpretability [43]. It describes the relationship between a dependent variable, y, and one or more independent variables, X. The general formula for LR model is given by:

$$y = \beta X + \epsilon \tag{10}$$

The input features are the historical positions and velocities of the satellite, given by X and the target variables are the predicted position and velocity offsets, given by y. β represents the coefficients and ϵ is the error term. From the predicted position and velocity offsets based on the LR model, using equation 6, the positions and velocities can be estimated.

d). ***PR Model***

The PR model is a regression algorithm that models the relationship between a dependent variable, y, and one or more independent variables, X as n^{th} degree polynomial [44]. The PR model is implemented as the data is non-linear in nature. The general formula for PR model is given by:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \epsilon \tag{11}$$

The input features are the historical positions and velocities of the satellite, and the target variables are the predicted position and velocity offsets given by y. In the PR model, the input features are transformed into polynomial features of required degree (2, 3, ... n) and then modeled using a linear model. The model then fits a polynomial function to the data, allowing for non-linear relationships to be captured. $\beta_0, \beta_1, \beta_2, ...\beta_n$ represents the coefficients of the polynomial term, and ϵ is the error term. From the predicted position and velocity offsets based on the PR model, using equation 6, the positions and velocities can be estimated.

VI. RESULTS

A comprehensive evaluation of different ML models for estimating the orbits of LEO SWARM-A were conducted and extended the proposed method to MEO GPS satellites. We assess the efficacy of ML models in predicting the satellite orbits, compare the performance of each model, identify their strengths and limitations, and enhance the orbit prediction accuracy. Therefore, this section presents the performance comparison of ML models, evaluation of each ML models implemented, and discussion to interpret the findings and highlight the implications.

1. Performance Comparison of ML Models

As stated in section IV, we define the number of days of data to be used by the ML models such as 1, 3, 10, 30 and 50 days. For clarity and focus, we present and discuss the performance comparison of the ML models that showed the best performance

when using a certain number of days of data. In this case, the ML models obtained best and reliable results when using 3-days of data. Therefore, we highlight the performance comparison of ML models using 3-days of data. The performance of the models were assessed using the RMSE as the primary evaluation metric.

Figure 3 shows that PR model demonstrated significant performance overall compared to KNR, RFR and LR model for both position RMSE (m) and velocity RMSE (m/s) of LEO SWARM-A satellite. In position RMSE (left), PR model obtained the lowest RMSE for different prediction intervals, followed by LR model. Since the KNR and RFR models obtained high position RMSE, the LR model position RMSE is not visible in the plot, which is as close as to PR model results. In case of velocity RMSE (right), PR model obtained lowest RMSE compared to the KNR, RFR and LR models. Additionally, the position and velocity RMSE analysis shows that RFR model collectively performed better when compared with KNR model. Overall, PR model exhibited best RMSE, indicating superior performance in estimating the position and velocity of the LEO SWARM-A satellite.

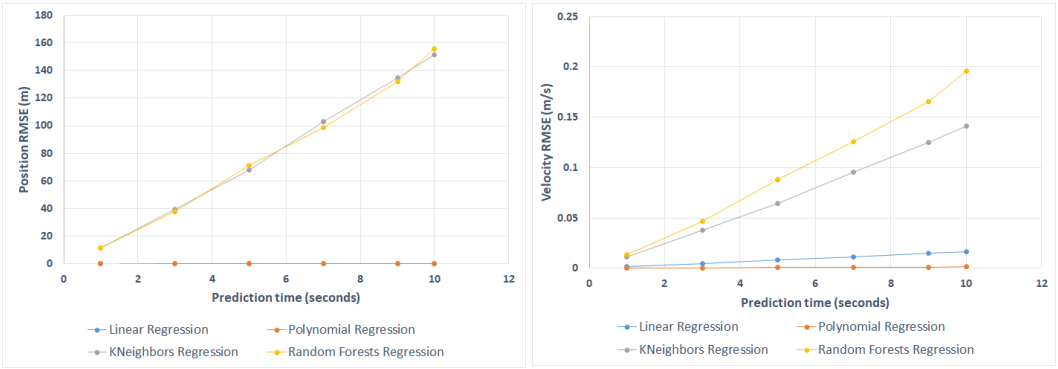


Figure 3: LEO SWARM-A satellite position RMSE (left) and velocity RMSE (right) of implemented ML Models for different prediction time intervals.

Similarly, different ML models were evaluated for estimating the position and velocity of MEO GPS satellites based on the prediction of position and velocity offsets. Figure 4 shows that PR model demonstrated significant performance overall compared to KNR, RFR and LR model for both position RMSE (m) and velocity RMSE (m/s) of GPS satellite PRN-14. Overall, PR model exhibited lowest RMSE, indicating superior performance in estimating the position and velocity of the MEO GPS satellite PRN-14.

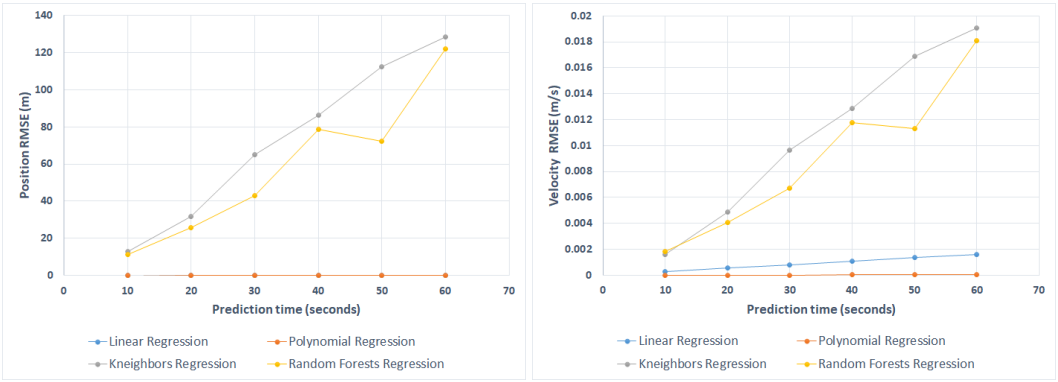


Figure 4: MEO GPS satellite position RMSE (left) and velocity RMSE (right) of implemented ML models for different prediction time intervals.

2. Evaluation of ML Models

In the evaluation of ML models, we present the results of various ML models used in estimating the position and velocity of the LEO SWARM-A satellite. Though we extended our analysis to MEO GPS satellites to evaluate the robustness and versatility of our proposed method, only the best RMSE results obtained by different ML models for MEO GPS satellite is presented as

table in the discussion of results section VI.3. Therefore, this section presents the results of various ML models implemented for LEO SWARM-A satellite.

a). KNeighbors Regressor (KNR) Model

KNR model is a regression technique that works by averaging the target values of K-nearest neighbors in the feature space to make prediction. In the KNR model, the total number of neighbors denoted by K, as stated in section V.1 a), can significantly influence the performance of the model. KNR model with K = 2 is considered to be the optimal parameter, yielding to position RMSE of 11.652 m and velocity RMSE of 0.01173 m/s when prediction time is one second and uses three days of data. When K = 1, the model offers better accuracy, but it overfits, has high variance and does not generalize to new datasets. When K is higher, the model was unable to learn the patterns in the data, which resulted in a highly biased model and yielded higher RMSE. Upon cross validation, using K = 2 yielded the best score, indicating that the model predicts the offsets with better accuracy. Therefore, using K = 2 provided an optimal balance between the bias and variance.

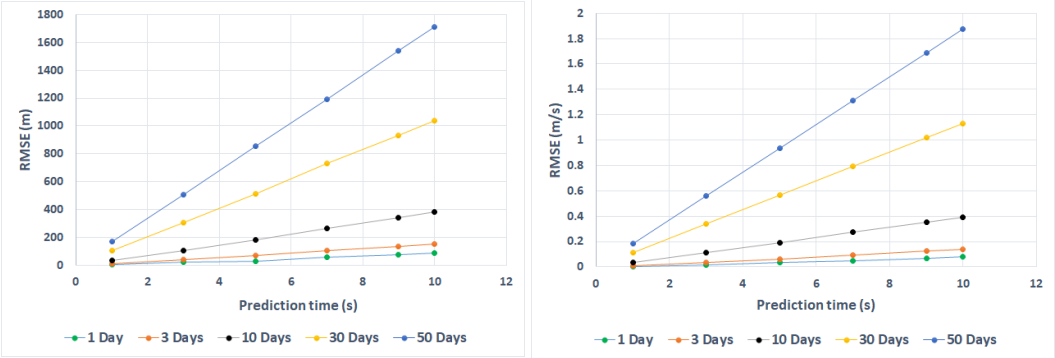


Figure 5: LEO position RMSE (left) and velocity RMSE (right) using KNR Model for different prediction time intervals and according to different days of data used.

Figure 5 shows the position and velocity RMSE obtained by KNR model for different prediction time intervals and according to the number of days of data used. Though best position and velocity RMSE results were achieved using one day of data, the KNR model is considered to be reliable when using three days of data, obtaining a better position and velocity RMSE. RMSE increases when the prediction time increase and amount of data used by the model increase. In general, deep learning models perform better when large amount of data is fed to the model. However, position and velocity RMSE for KNR model increased when more number of days of data were used, as seen in figure 5. Overall, the evaluation of KNR model shows that highest RMSE values are obtained for predicting the position and velocity offsets.

b). Random Forests Regressor (RFR) Model

RFR model is a supervised learning algorithm which operates by building a multitude of decision trees and outputs a mean prediction of the individual trees. In the RFR model, total number of decision trees denoted by T as stated in V.1 b), can significantly influence the accuracy, bias, variance and overall performance of the model. In general, more number of decision trees ($n\_estimators$) will lead to improved accuracy, as the forest averages the prediction of the individual trees. However, at a certain point, with more number of trees, the improvement becomes negligible and computational costs increases. Thus, the RFR model with number of trees, T = 100 is considered to be the optimal number of estimators, yielding to position RMSE of 11.45 m and velocity RMSE of 0.0134 m/s when prediction time is one second and uses three days of data. With few number of trees, i.e., when T < 100, the model was unable to learn the patterns in the data, which lead to high variance and causes the model to overfit. Upon cross validation, when T is higher, i.e., T > 100, the model yielded the best scores. However, the improvement becomes negligible and increased the computational costs. Therefore, T = 100 is considered to be the optimal parameter in predicting the position and velocity offsets via cross validation, providing a balance between the accuracy and computational costs. In addition, feature scaling and hyperparameter tuning are done during the cross-validation to ensure an effective process for optimizing the model performance.

Figure 6 shows the position and velocity RMSE obtained by RFR model for different prediction time intervals and according to the number of days of data used as stated in section IV. Though best position and velocity RMSE results were achieved using one day of data, the RFR model is considered to be more reliable when using three days of data, obtaining a better position and velocity RMSE. Similar to the KNR model, position and velocity RMSE increases when the prediction time and days of data used increase, as seen in figure 6. Overall, the evaluation of RFR model shows that higher RMSE values are achieved in predicting the position and velocity offsets.

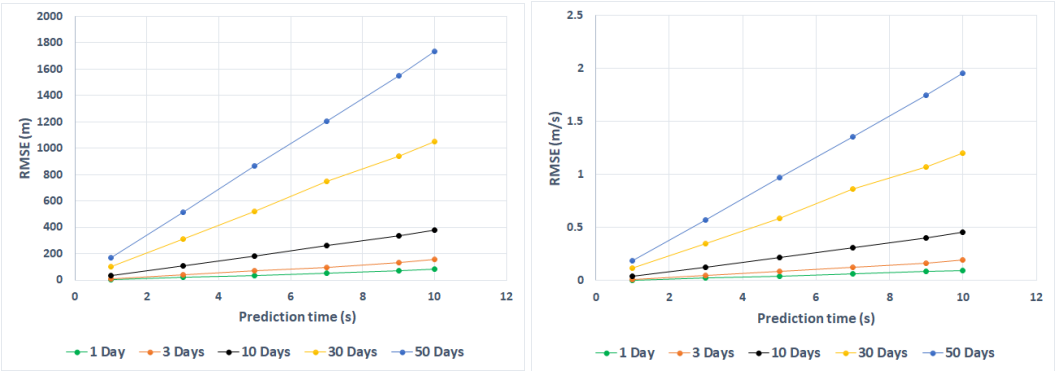


Figure 6: LEO position RMSE (left) and velocity RMSE (right) using RFR Model for different prediction time intervals and according to different days of data used.

c). Linear Regression (LR) Model

LR model is one of the simplest and most interpretable ML model implemented for estimating the position and velocity of the satellites. The simplicity of the LR model lies in its straightforward approach. LR model achieved position RMSE of 0.00077 m and velocity RMSE of 0.00155 m/s for prediction time of one second when using one day of data. Though best position and velocity RMSE results were achieved using one day of data, the LR model is considered to be reliable and robust when using three days of data, obtaining a better position and velocity RMSE. Figure 7 shows the position and velocity RMSE obtained by LR model for different prediction time intervals and according to the number of days of data used. Similar to the KNR and RFR models, position and velocity RMSE increases when the prediction time and days of data used increase. Overall, the evaluation of LR model shows that lower RMSE values are achieved in predicting the position and velocity offsets, significantly better when compared to the KNR and RFR models.

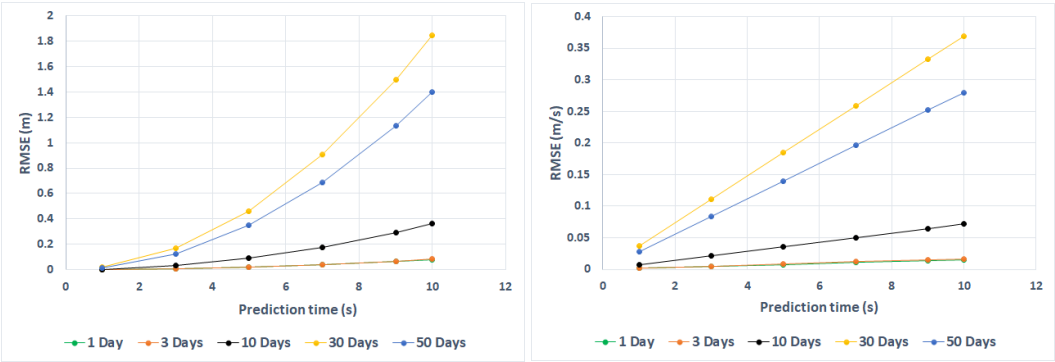


Figure 7: LEO position RMSE (left) and velocity RMSE (right) using LR Model for different prediction time intervals and according to different days of data used.

d). Polynomial Regression (PR) Model

In addition to the KNR, RFR and LR models, we have also implemented the PR model. PR model, by utilizing polynomial features of required degree, the model improved over the LR model by capturing nonlinear relationships in the data. As stated in the section V, PR model with degree 3, that is, third-degree polynomial regression was implemented to predict the position and velocity offsets. PR model performed better when using degree-3 with position RMSE of 0.00007 m and velocity RMSE of 0.00014 m/s for LEO SWARM-A satellite when using three days of data and the prediction time is one second. Higher or lower polynomial degrees resulted in increased RMSE, indicating overfitting or underfitting respectively.

Figure 8 shows the position and velocity RMSE obtained by PR model for different prediction time intervals and according to the number of days of data used. Unlike the other ML models, the best position and velocity RMSE results were not achieved when using one day of data. Instead, PR model achieved best RMSE when using three days of data. This indicates why RMSE results obtained using three days of data were considered to be reliable when best RMSE was obtained using one day of data

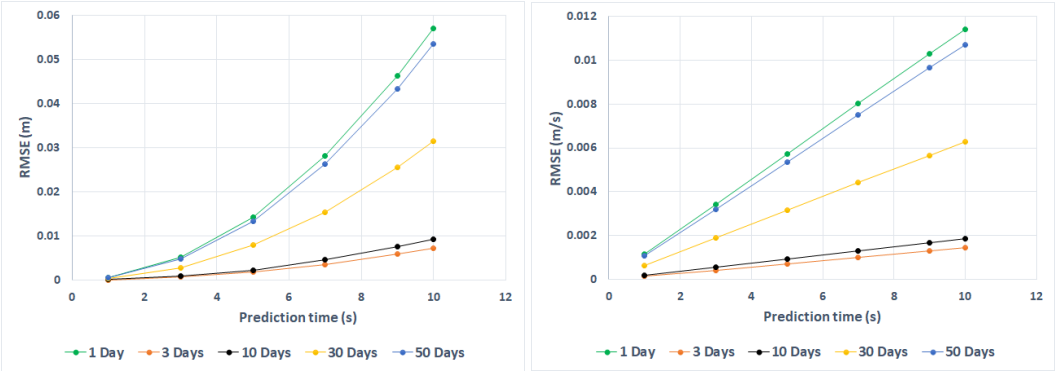


Figure 8: LEO position RMSE (left) and velocity RMSE (right) using PR Model for different prediction time intervals and according to different days of data used.

by other models. In addition, position and velocity RMSE using the PR model increases when prediction time and days of data used increase, similar to other models, as seen in the figure 8. Overall, PR model is more robust, reliable and efficient when using three days of data. The evaluation of PR model shows that lowest RMSE values are achieved in predicting the position and velocity offsets. PR model significantly outperformed all the other ML models such as KNR, RFR, and LR models.

In addition to the RMSE, since PR model outperformed other ML models, the difference plot for position (top) and velocity (bottom) are illustrated in figure 9. The position difference is the difference between the true position values and estimated positions values with respect to time in hours. Similarly, the velocity difference is the difference between the true velocity values and the estimated velocity values. The estimated position and velocity values are obtained using equation 6 based on the predicted position and velocity offsets by the PR model. The x-axis in figure 9 shows the date and hours of the test data. In this case, when using three days of data, first 70% of the data were used as the training data and last 30% of the data were used as test data. Therefore, the x-axis illustrates that test data from 04/10/2023 03:00 hour to 05/10/2023 00:00 hour were used in estimating the position and velocity values and their difference to actual values are plotted.

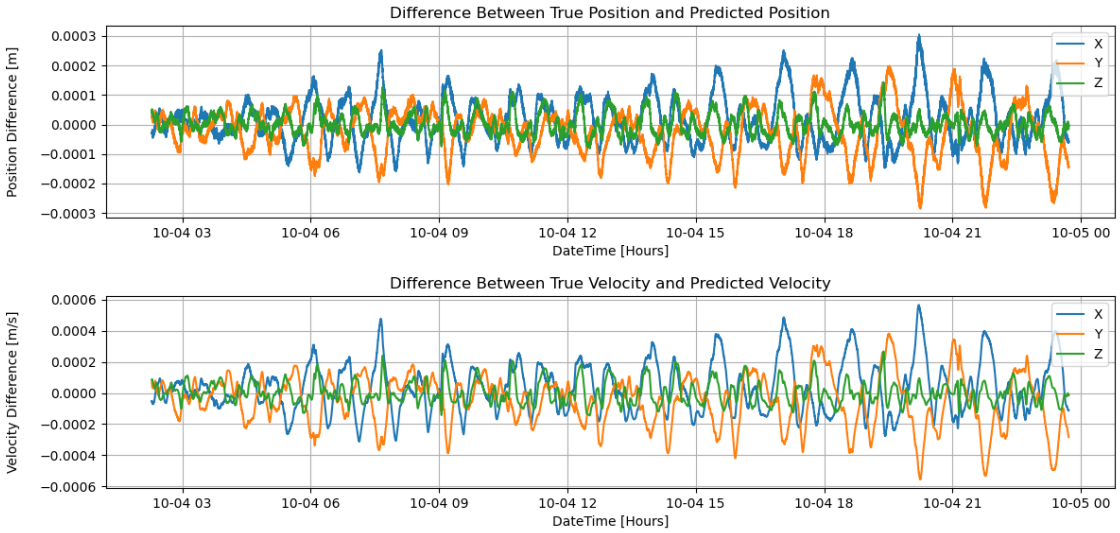


Figure 9: Differences of the true positions and velocities to that of estimated positions and velocities for SWARM-A satellite test data points.

3. Discussion of Results

From the results, a comprehensive evaluation of different ML models for estimating the position and velocity of LEO SWARM-A satellite and GPS satellites, based on the prediction of the offsets, reveal distinct and significant performance in terms of

accuracy, measured using RMSE. The performance of KNR model can be significantly influenced by the total number of neighbors, given by K. An optimal K value is determined by using cross validation for the optimized performance of KNR model. The analysis shows that $K = 2$ provided an optimal balance between the bias and variance. KNR model yielded higher RMSE overall compared to the other ML models despite optimizing the model performance using cross validation. The implementation of RFR model performed slightly better than KNR model. The performance of the RFR model can be influenced by the T value, i.e., the total number of trees. The best optimal T value is determined using a grid search with cross validation. In addition, feature scaling and hyperparameter tuning were done to optimize the performance of the model. The analysis shows that $T = 100$ provided an optimal balance, yielding to better RMSE. However, the RMSE values were higher than LR and PR models. LR model performed better for estimating the position and velocity of the LEO SWARM-A and GPS satellites, as evidenced from the RMSE provided in Table 1. A significantly lower RMSE was obtained when compared to the RFR and KNR models. In PR model, a third degree polynomial includes the original input features as well as their squared terms and cubic terms. The analysis shows that best RMSE results are obtained when using a third degree polynomial for LEO SWARM-A satellite. In case of MEO GPS satellite, second degree polynomial yielded the best RMSE results. The model demonstrated an outstanding performance compared to the other ML models due to incorporating higher-order terms, thereby effectively capturing the non-linear dynamics present in the satellite orbits.

Overall, PR model introduces adaptability by allowing the model to fit the data more accurately compared to LR model. The adaptability of the model leads to significant outperformance with lower RMSE when the underlying relationship between variables is non-linear. Though the RFR model is a powerful algorithm for capturing non-linear relationships, the model may find it difficult to capture certain non-linear relationships that a polynomial model capture directly. The non-linearity of the satellite orbit data used in this study might be better approximated without overfitting by a third degree polynomial than by an ensemble RFR model. On the other hand, KNR model is a local approximation method, making predictions on the nearest data points. While PR model uses a global approximation, fitting a polynomial equation to the entire dataset leading to better generalization for non-linear trends. In conclusion, for both LEO and MEO satellites, PR model outperformed other ML models such as KNR, RFR and LR models due to its ability to effectively model the non-linear characteristics in the satellite data with an optimal degree of complexity as determined by cross validation.

The study provides a comprehensive analysis of different ML models used in estimating the position and velocity of the satellites. The research focused on providing a simplified and alternative approach based on standalone ML-based method compared to traditional or hybrid methods. The models were trained and tested using real-world data, ensuring that the results are applicable to actual satellites. The study implemented ML models, considering also the computational costs and robustness of the models. Thus, it provides a balanced assessment of the trade-offs associated with selecting a model by evaluating both accuracy and computational costs. On the other hand, the study did not extensively address the computational costs involved in implementing ML models such as RFR and KNR models. The performance of the ML models might vary significantly when applied to different satellites in the LEO. The study mainly implemented ML models for short-term predictions of few seconds, which may be a limitation with respect to MEO as it requires accurate long-term predictions to ensure the reliability of PNT services. However, short term predictions are crucial in the context of LEO satellites.

In this study, different ML models were analyzed for different number of days of data (1, 3, 10, 30 and 50 days) and prediction time (1, 3, 5, 7, 9 and 10 seconds for LEO; 10, 20, 30, 40, 50 and 60 seconds for GPS) and evaluated using RMSE. The table 1 summarizes the performance of ML models which obtained best RMSE results.

Table 1: Best RMSE results obtained by different ML models for LEO SWARM-A and MEO GPS Satellite PRN-14.

LEO SWARM-A Satellite				
Number of Days Used	**ML Method**	**Prediction Time (s)**	**Position_RMSE (m)**	**Velocity_RMSE (m/s)**
3	KNeighbors Regression	1	11.65206606	0.011731784
3	Random Forests Regression	1	11.45781611	0.013498317
3	Linear Regression	1	0.000841591	0.001683116
3	Polynomial Regression	1	7.23E-05	0.000144123

MEO GPS Satellite PRN-14				
Number of Days Used	**ML Method**	**Prediction Time (s)**	**Position_RMSE (m)**	**Velocity_RMSE (m/s)**
3	KNeighbors Regression	10	12.9881978	0.001578546
3	Random Forests Regression	10	11.48159881	0.001818075
3	Linear Regression	10	0.001323894	0.000265484
3	Polynomial Regression	10	9.71E-05	3.35E-06

VII. CONCLUSION AND FUTURE WORK

This study aimed to evaluate the feasibility and effectiveness of using various machine learning models to predict the orbits of low Earth orbit (LEO) satellites, with a focus on improving prediction accuracy as measured by RMSE. Additionally, we extended our analysis to medium Earth orbit (MEO) GPS satellites to identify any potential differences in prediction accuracy between LEO and MEO satellites.

The study utilized different ML models, such as KNeighbors regressor (KNR), Random Forests regressor (RFR), linear regression (LR) and polynomial regression (PR). Despite the availability of more complex models, such as KNR and RFR, the polynomial regression model outperformed all the other ML models implemented, highlighting the significance of model selection for a certain application. This is evident from the primary evaluation metric, RMSE, used in assessing the performance of the ML models. Overall, a significantly lower RMSE was obtained by using PR, followed by LR, while higher RMSE were obtained using RFR and KNR models.

Furthermore, the estimation of position and velocity of the LEO SWARM-A and MEO GPS satellite was carried out based on the offset predictions. The ML models predicted the offsets for different time intervals such as 1, 3, 5, 7, 9 and 10 seconds in case of LEO satellite, while for MEO satellites, the prediction time intervals are 10, 20, 30, 40, 50 and 60 seconds. The study, therefore, highlights the critical importance of ML-based precise short-term predictions in LEO satellites, due to the satellites orbiting at high speed and dynamic space environment.

In future work, we will explore more advanced machine learning models for long-term orbit predictions and further enhance the orbit prediction accuracy. Additionally, investigation on the influence of other features such as atmospheric drag, solar position and satellite acceleration on the model performance will be assessed. In summary, this study contributes to the significance of standalone ML-based technique in estimating the position and velocity of especially the LEO satellite at millimeter levels, ultimately enhancing the performance and efficiency of space-based applications and services.

REFERENCES

[1] K. Selvan, A. Siemuri, F. S. Prol, P. Välisuo, M. Z. H. Bhuiyan, and H. Kuusniemi, "Precise orbit determination of LEO satellites: a systematic review," *GPS Solutions*, vol. 27, no. 4, p. 178, 2023.

[2] O. Montenbruck, *Space Applications*, P. J. Teunissen and O. Montenbruck, Eds. Springer International Publishing, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-42928-1_32

[3] T. Springer and U. Hugentobler, "IGS ultra rapid products for (near-) real-time applications," *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy*, vol. 26, no. 6, pp. 623–628, 2001, proceedings of the First COST Action 716 Workshop Towards Operational GPS Meteorology and the Second Network Workshop of the International GPS Service (IGS). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1464189501001119

[4] T. Hadas and J. Bosy, "IGS RTS precise orbits and clocks verification and quality degradation over time," *GPS Solutions*, vol. 19, no. 1, pp. 93–105, 2015. [Online]. Available: https://doi.org/10.1007/s10291-014-0369-5

[5] R. Flores, B. M. Burhani, and E. Fantino, "A method for accurate and efficient propagation of satellite orbits: A case study for a molniya orbit," *Alexandria Engineering Journal*, vol. 60, pp. 2661–2676, 2021.

[6] A. Nowak, R. Zajdel, and K. Sośnica, "Optimization of orbit prediction strategies for GNSS satellites," *Acta Astronautica*, vol. 209, pp. 132–145, 2023.

[7] O. Montenbruck, "Numerical integration methods for orbital motion," *Celestial Mechanics and Dynamical Astronomy*, pp. 59–69, 1992. [Online]. Available: https://doi.org/10.1007/BF00049361

[8] F. R. Hoots, P. W. Schumacher Jr., and R. A. Glover, "History of analytical orbit modeling in the U.S. space surveillance system," *Journal of Guidance, Control, and Dynamics*, 2004. [Online]. Available: https://doi.org/10.2514/1.9161

[9] J. F. San-Juan, I. Pérez, M. San-Martín, and E. P. Vergara, "Hybrid SGP4 orbit propagator," *Acta Astronautica*, vol. 137, pp. 254–260, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576516311535

[10] B. D. Tapley, B. E. Schutz, and G. H. Born, *Statistical Orbit Determination*. Elsevier Inc., 2004. [Online]. Available: https://www.sciencedirect.com/book/9780126836301/statistical-orbit-determination#book-info

[11] J. Haidar-Ahmad, N. Khairallah, and Z. M. Kassas, "A hybrid analytical-machine learning approach for LEO satellite orbit prediction," *25th International Conference on Information Fusion (FUSION), Linköping, Sweden*, pp. 1–7, 2022.

[12] D. A. Vallado and P. Crawford, "SGP4 orbit determination," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Honolulu, Hawaii*, 2008.

[13] C. Levit and W. Marshall, "Improved orbit predictions using two-line elements," *Advances in Space Research*, vol. 47, pp. 1107–1115, 2011.

[14] H. Peng and X. Bai, "Improving orbit prediction accuracy through supervised machine learning," *Advances in Space Research*, 2018. [Online]. Available: https://arxiv.org/pdf/1801.04856.pdf

[15] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. AMLBook, 2012.

[16] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997. [Online]. Available: http://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf

[17] C. Ampatzis and D. Izzo, "Machine learning techniques for approximation of objective functions in trajectory optimisation," in *Proceedings of the IJCAI-09 workshop on artificial intelligence in space*, 2009, pp. 1–6.

[18] J. Hartikainen, M. Seppanen, and S. Sarkka, "State-space inference for non-linear latent force models with application to satellite orbit prediction," in *Proceedings of the 29th International Conference on Machine Learning, Edinburgh, Scotland, UK*, 2012.

[19] S. Sharma and J. W. Cutler, "Robust orbit determination and classification: A learning theoretic approach," *IPN Progress Report*, vol. 42, p. 203, 2015.

[20] F. S. Prol, M. Z. H. Bhuiyan, S. Kaasalainen, E. S. Lohan, J. Praks, K. Çelikbilek, and H. Kuusniemi, "Simulations of dedicated leo-pnt systems for precise point positioning: Methodology, parameter analysis, and accuracy evaluation," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–19, 2024.

[21] B. Wang, H. Peng, and K. Li, "Precision analysis of an analytical method in space debris orbit prediction," *International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2016)*, pp. 238–242, 2016. [Online]. Available: https://doi.org/10.2991/iccia-16.2016.43

[22] A. R. Muldoon, G. H. Elkaim, and I. Rickard. (2009) Improved orbital debris trajectory estimation based on sequential TLE processing. [Online]. Available: https://api.semanticscholar.org/CorpusID:8407890

[23] Q. Wang, C. Hu, T. Xu, G. Chang, and A. H. Moraleda, "Impacts of earth rotation parameters on GNSS ultra-rapid orbit prediction: Derivation and real-time correction," *Advances in Space Research*, vol. 60, pp. 2855–2870, 2017.

[24] C. Lei, B. Xian-Zong, L. Yan-Gang, and L. Ke-Bo, "Orbital error analysis based on historical data," in *Orbital Data Applications for Space Objects*, 2016, pp. 77–105.

[25] J. Sang, B. Li, J. Chen, P. Zhang, and J. Ning, "Analytical representations of precise orbit predictions for earth-orbiting space objects," *Advances in Space Research*, vol. 59, pp. 698–714, 2017.

[26] G. Jadala, G. N. Meedinti, and R. Delhibabu, "Satellite orbit prediction using a machine learning approach," *Workshops at the 5th International Conference on Applied Informatics*, pp. 28–46, 2022.

[27] Z. Y.-C. Liu, S. Tarlow, M. Akbar, Q. Donnellan, and D. Senkow, "Improved orbital propagator integrated with SGP4 and machine learning," *Small Satellite Conference, Utah*, 2021.

[28] J. Gou, C. Rösch, E. Shehaj, K. Chen, M. Kiani Shahvandi, B. Soja, and M. Rothacher, "Improving the accuracy of GNSS orbit predictions using machine learning approaches," *EGU General Assembly 2022, Vienna, Austria*, 2022. [Online]. Available: https://doi.org/10.5194/egusphere-egu22-1834

[29] H. Peng and X. Bai, "Machine learning approach to improve satellite orbit prediction accuracy using publicly available data," in *J Astronaut Sci*, vol. 67, 2020, p. 762–793.

[30] H. Peng and X. Bai, "Comparative evaluation of three machine learning algorithms on improving orbit prediction accuracy," *Astrodynamics*, vol. 3, no. 4, p. 325–343, 2019. [Online]. Available: https://doi.org/10.1007/s42064-018-0055-4

[31] H. Peng and X. Bai, "Improving orbit prediction accuracy through supervised machine learning," *Advances in Space Research*, vol. 61, pp. 2628–2646, 2018.

[32] I. E. Dawoodjee and M. Rajeswari, "Establishing a regression baseline for predicting satellite motion," *Journal of Applied Technology and Innovation*, vol. 5, pp. 47–58, 2021.

[33] H. Peng and X. Bai, "Artificial neural network–based machine learning approach to improve orbit prediction accuracy," *Journal of Spacecraft and Rockets*, vol. 55, pp. 1248–1260, 2018.

[34] F. S. Prol, R. M. Ferre, Z. Saleem, P. Välisuo, C. Pinell, E. S. Lohan, M. Elsanhoury, M. Elmusrati, S. Islam, K. Çelikbilek, K. Selvan, J. Yliaho, K. Rutledge, A. Ojala, L. Ferranti, J. Praks, M. Z. H. Bhuiyan, S. Kaasalainen, and H. Kuusniemi,

"Position, navigation, and timing (PNT) through low earth orbit (LEO) satellites: A survey on current status, challenges, and opportunities," *IEEE Access*, vol. 10, pp. 83 971–84 002, 2022.

[35] T. Mortlock and Z. M. Kassas, "Assessing machine learning for leo satellite orbit determination in simultaneous tracking and navigation," in *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–8.

[36] S. E. Kozhaya, J. A. Haidar-Ahmad, A. A. Abdallah, Z. M. Kassas, and S. S. Saab, "Comparison of neural network architectures for simultaneous tracking and navigation with LEO satellites," in *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), St. Louis, Missouri*, 2021, pp. 2507–2520.

[37] ESA. (2024) European Space Agency, SWARM Data Access. [Online]. Available: https://swarm-diss.eo.esa.int/#

[38] NGA Public Affairs. (2023) National Geospatial-Intelligence Agency, Office of Geomatics. [Online]. Available: https://earth-info.nga.mil/index.php?action=home

[39] H. Shou, "Orbit propagation and determination of low earth orbit satellites," *International Journal of Antennas and Propagation*, vol. 2014, 2014. [Online]. Available: https://doi.org/10.1155/2014/903026

[40] E. Friis-Christensen, H. Lühr, and G. Hulot, "Swarm: A constellation to study the earth's magnetic field," *Earth, Planets and Space*, vol. 58, pp. 351–358, 2006. [Online]. Available: https://doi.org/10.1186/BF03351933

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[42] L. Breiman, "Random forests," *Springer Link*, vol. 45, pp. 5–32, 2001.

[43] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, *Linear Regression*. Cham: Springer International Publishing, 2023, pp. 69–134. [Online]. Available: https://doi.org/10.1007/978-3-031-38747-0_3

[44] E. Ostertagová, "Modelling using polynomial regression," *Procedia Engineering*, vol. 48, pp. 500–506, 2012, modelling of Mechanical and Mechatronics Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705812046085

Publication VIII

Federated Learning and TinyML for Localization: Benefits, Applications, and Challenges

Akpojoto Siemuri
School of Technology and Innovation
University of Vaasa
Vaasa, Finland
Email: akpojoto.siemuri@uwasa.fi

Mohammed Elmusrati
School of Technology and Innovation
Computing Sciences
University of Vaasa
Vaasa, Finland
Email: mohammed.elmusrati@uwasa.fi

January 5, 2026

Abstract

The growing demand for intelligent and context-aware Internet of Things (IoT) systems has spurred the advancement of machine learning (ML) solutions for real-time localization. Deploying ML models on distributed, resource-constrained edge devices offers substantial opportunities while posing unique challenges. This article presents a comprehensive review of strategies for adapting and designing ML models for deployment on embedded hardware with limited resources. We emphasize Tiny Machine Learning (TinyML), a paradigm that facilitates ultra-low-power ML execution on microcontrollers and sensors, and explore its synergy with Federated Learning (FL), which enables collaborative model training across devices without centralizing sensitive data. We examine training strategies, deployment workflows from cloud to edge, model optimization techniques, and key applications of TinyML and FL in localization. Benefits such as real-time processing, privacy preservation, and energy efficiency are analyzed, along with challenges including computational constraints, interoperability, and security vulnerabilities. The article concludes by identifying research gaps and outlining future directions for integrating TinyML and FL in large-scale localization systems.

Keywords: Edge computing, Federated learning, TinyML, Machine learning, Localization.

1 Introduction

Edge computing and the Internet of Things (IoT) have revolutionized data collection, processing, and utilization. As IoT applications evolve toward greater intelligence, the demand for localized decision-making has surged. Localization—the determination of the position of individuals, assets, or devices—is essential for diverse applications, including smart homes, industrial automation, autonomous vehicles, and healthcare monitoring.

TinyML is a branch of machine learning focused on deploying small, efficient ML models directly on resource-constrained edge devices, such as microcontrollers and sensors. These devices typically have limited computational power, memory, and battery life, making it challenging to run conventional ML models. TinyML enables real-time, low-power data processing and analytics directly on these devices without relying on constant cloud connectivity [11].

By optimizing ML models to fit within the constraints of edge devices, TinyML allows for applications like voice recognition, image classification, anomaly detection, and predictive maintenance in environments where cloud access may be limited or where data privacy is critical. This field has grown rapidly with advancements in model compression, quantization, and specialized hardware accelerators tailored for on-device ML.

Conventional ML models depend on centralized cloud platforms for computation, but this paradigm is often inadequate due to latency, bandwidth limitations, and privacy issues. Edge Machine Learning (Edge ML) mitigates these concerns by performing inference directly on devices near data sources. A pivotal advancement in this domain is TinyML, which develops compact, efficient models suitable for microcontrollers, embedded processors, and other low-power devices [1,3]. Federated Learning (FL) complements TinyML by allowing distributed devices to collaboratively train models while keeping data localized, enhancing privacy and scalability [2].

By tailoring ML algorithms for resource-limited environments, TinyML and FL unlock opportunities in areas such as voice recognition, anomaly detection, image classification, and predictive maintenance.

Importantly, they support real-time localization while eliminating the need for constant cloud access. This article investigates the application of TinyML and FL in localization, highlighting their benefits and persisting challenges. To enhance readability, we have restructured the content for a general audience, added explanatory notes, and improved flow between sections.

2 Overview of TinyML and Federated Learning

TinyML refers to the deployment of ML models on extremely resource-constrained devices, typically with memory in the order of kilobytes and power consumption in microwatts. It involves techniques to shrink model size and computational requirements while maintaining acceptable performance.

Federated Learning (FL), on the other hand, is a distributed ML approach where multiple devices collaboratively train a shared model under the orchestration of a central server. Instead of transmitting raw data, devices send model updates, preserving data privacy. The integration of FL with TinyML—often termed Federated TinyML—allows resource-limited devices to participate in collaborative learning, making it particularly suitable for IoT scenarios where devices are numerous and heterogeneous [3, 4].

This combination is especially promising for localization tasks, as it leverages distributed sensing while addressing privacy and efficiency concerns. For example, in a network of IoT sensors, Federated Learning can aggregate local insights without exposing individual device data, enabling robust and privacy-preserving location estimation [5–7].

While modern cloud infrastructure may appear capable of handling vast data volumes, practical constraints such as cost, latency, and connectivity limitations often make centralized processing infeasible. Edge machine learning (Edge ML) addresses these challenges by enabling data processing directly at the source—on devices located at the network's periphery. This approach is essential for applications requiring low-latency responses, real-time predictions, or operation in environments with poor or nonexistent cloud connectivity. Additionally, legal restrictions and the need to preprocess large datasets locally further justify the use of edge computing [8].

Edge ML—also referred to as edge artificial intelligence or edge AI—is defined as the execution of ML algorithms on edge devices to make decisions and predictions as close as possible to the data source [9]. This can be implemented in a distributed manner, forming the basis of distributed edge intelligence—a rapidly evolving research area that enables the deployment of machine learning and deep learning models near the point of data generation [10].

3 Training Strategies for Edge ML Models

As edge computing continues to reshape the landscape of intelligent systems, training machine learning models directly on or for edge devices has become a critical area of research and development. Unlike traditional cloud-based workflows, edge ML training strategies must account for constraints such as limited memory, processing power, and energy availability—while still delivering accurate and timely predictions. This section explores three primary approaches to training edge ML models: on-device training, cloud-to-edge deployment, and federated learning. Each method offers distinct advantages and trade-offs in terms

of scalability, security and privacy, latency, and resource efficiency, particularly in applications like IoT localization and real-time inference [12–14].

3.1 On-Device Training

Training ML models directly on edge devices, such as Raspberry Pi, smartphones, or embedded boards, is feasible but resource-intensive. Devices like the Raspberry Pi 4 with at least 4 GB of RAM can handle lightweight training tasks. Performance can be boosted using hardware accelerators, including Google Coral USB, Intel Movidius Neural Compute Stick, or NVIDIA Jetson Nano.

The software stack typically comprises Raspberry Pi OS or lightweight Linux distributions, with Python as the primary language owing to its versatility and extensive libraries. Popular frameworks include TensorFlow Lite, PyTorch Mobile, Edge Impulse, and scikit-learn.

For IoT localization, on-device ML facilitates rapid predictions by eliminating the need to transfer voluminous raw data over networks. Everyday objects like smartwatches, appliances, and vehicles can embed ML models to infer context or location autonomously.

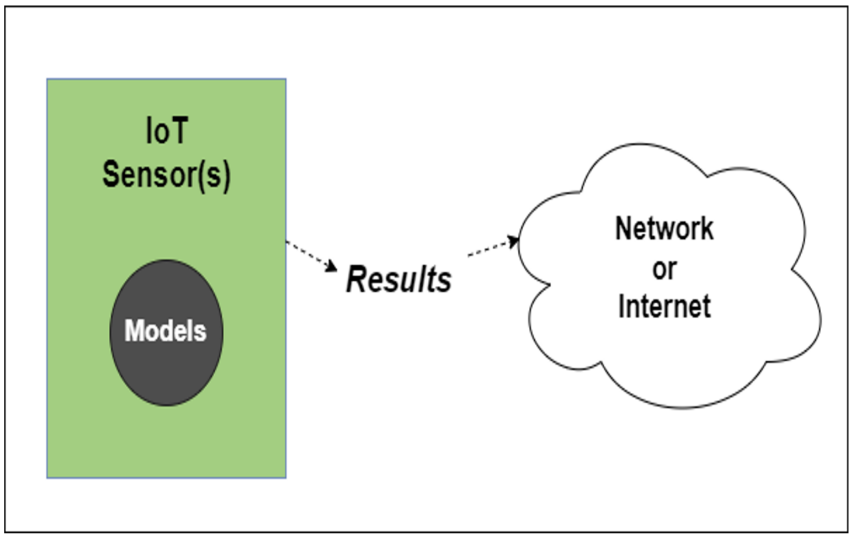


Figure 1: Machine learning model implemented on an edge device.

However, training intricate models on constrained devices is inefficient. Simple models can be trained using lightweight frameworks, whereas complex ones are typically developed in the cloud or on powerful machines before optimization for edge deployment. Key optimization methods encompass quantization (e.g., int8 or float16), pruning, and knowledge distillation [15, 16]. Optimized models are then converted to formats like TensorFlow Lite or ONNX for embedded compatibility [16].

The workflow generally includes: (1) data collection, (2) lightweight preprocessing, (3) training (on-device for simple models, offloaded for complex ones), and (4) deployment []. Inference is supported by runtimes such as TensorFlow Lite Interpreter or ONNX Runtime, with remote monitoring for ongoing enhancements [17]. To improve efficiency, hybrid approaches combining local fine-tuning with global models are increasingly adopted [15].

3.2 Cloud-to-Edge Deployment

A prevalent strategy is the cloud-to-edge pipeline, where training occurs on robust cloud infrastructure, and optimized models are deployed to edge devices. Platforms like Databricks, Google Cloud AI, AWS SageMaker, and Microsoft Azure offer GPUs and TPUs for scalable training [18].

Edge-collected data undergoes cloud-based preprocessing, including normalization, augmentation, and filtering. Training employs frameworks like TensorFlow, PyTorch, or Keras [17]. Subsequently, optimizations

such as pruning and distillation minimize model complexity without substantial accuracy loss [17]. Models are converted to edge-friendly formats like TensorFlow Lite, ONNX, or TensorRT [19]. Figure 2 demonstrates the deployment of ML model at the edge.

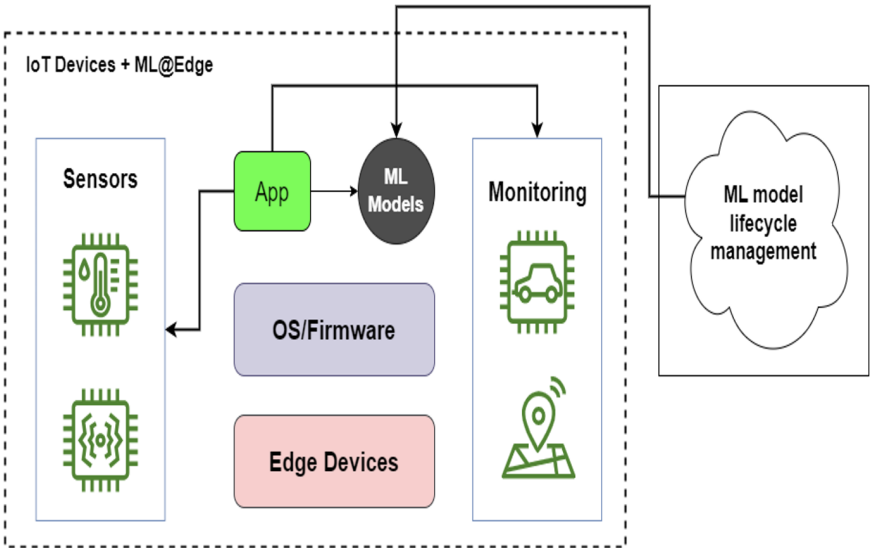


Figure 2: Cloud-based training with optimized edge deployment, adapted from [20]

Deployment leverages interpreters like TensorFlow Lite or PyTorch Mobile, augmented by accelerators such as Coral USB or Jetson Nano. This approach harmonizes scalability with efficiency, delivering potent training capabilities alongside low-latency edge inference. Exemplary applications include on-camera object detection for surveillance, predictive maintenance in industrial IoT, and real-time speech recognition in voice assistants [21]. Enhancements in this area include automated model conversion tools to streamline the deployment process.

3.3 Federated Learning Approaches

Federated Learning enables collaborative training across edge devices without centralizing data. In localization contexts, devices can aggregate insights from local RF signals or sensor data to build global models [24]. This is particularly advantageous for TinyML, as it distributes computational load and enhances model robustness through diverse data sources [20].

FL workflows involve local training on devices, followed by aggregation of model updates on a central server. Techniques like FedAvg (Federated Averaging) are commonly used [22]. For resource-constrained settings, optimizations such as secure aggregation and differential privacy are integrated to bolster security and efficiency [23]. Recent advancements include personalized FL, where models are adapted to individual device contexts for better performance [24].

4 Applications of TinyML and FL in Localization

The convergence of TinyML and Federated Learning (FL) has enabled a new class of intelligent localization systems capable of operating in constrained environments where power, latency, and privacy are critical [4]. By shifting computation to the edge and enabling collaborative learning across devices, these technologies support a wide range of applications—from smart homes to industrial automation—without relying on centralized infrastructure [25]. Below, we explore key use cases with expanded examples and supporting research [3].

Recent studies have demonstrated the effectiveness of TinyML and FL in localization tasks. Avellaneda et al. [32] proposed a deep learning-based TinyML solution for indoor asset tracking, achieving 94% accuracy after post-processing. Méndez et al. [28] compared Wi-Fi RSSI and CSI fingerprinting, showing that CSI outperformed RSSI by 5–6% in single-sample classification and over 50% in time-series scenarios. Emerging research also explores the integration of TinyML and FL with 5G networks, enabling ultra-low-latency and high-precision localization through edge-native architectures and enhanced spectrum sensing [33]. Some applications covered by the literature are presented below.

4.1 Zone Classification

TinyML models can classify physical zones (e.g., kitchen, living room, office) using ambient wireless signals such as Wi-Fi RSSI, Bluetooth Low Energy (BLE), or generic RF patterns [26, 27]. This enhances context awareness in smart environments, allowing systems to adapt behavior based on user location. For instance, entering a kitchen could trigger recipe suggestions on a smart display, while presence in a bedroom might activate sleep mode on connected devices. FL enables these models to be trained across multiple homes without compromising user privacy, improving generalization across diverse layouts and signal conditions [28].

4.2 Activity-Based Localization

TinyML can leverage inertial measurement unit (IMU) data to infer user activities and movement patterns, such as walking, climbing stairs, or standing still [29]. These insights support pedestrian dead reckoning and indoor navigation, particularly in fitness tracking and elderly care applications [4]. FL allows models to adapt to individual gait patterns and device placements, improving accuracy over time without centralized data collection [3].

4.3 RF Fingerprinting

RF fingerprinting enables anchor-free localization by learning unique signal characteristics at different locations. TinyML models can be trained to recognize these patterns using Wi-Fi RSSI, Channel State Information (CSI), or BLE signals, eliminating the need for fixed infrastructure [28, 30]. FL enhances this approach by aggregating fingerprints from multiple devices, allowing the system to adapt to environmental changes such as furniture rearrangement or signal interference [4]. This is especially useful in dynamic public spaces like airports, shopping malls, or exhibition centers [3].

4.4 Asset Tracking

TinyML-powered asset trackers can monitor the location of equipment, inventory, or medical devices in real time using wireless signals [25, 28]. In industrial settings, this supports logistics optimization, predictive maintenance, and theft prevention [4]. FL enables collaborative learning across devices deployed in different zones, allowing the system to adapt to layout changes, signal obstructions, or usage patterns [3].

4.5 Seamless Indoor–Outdoor Positioning

Combining GNSS (e.g., GPS), Ultra-Wideband (UWB), Wi-Fi, and IMU data with TinyML allows devices to maintain accurate positioning across indoor and outdoor environments [25, 31]. TinyML models can detect transitions between signal domains and adjust inference strategies accordingly [3]. FL supports the fusion of data from multiple users and devices, refining global positioning models and improving accuracy in challenging environments such as urban canyons, underground facilities, or multi-level buildings [4].

5 Benefits of TinyML and FL in Localization

The integration of TinyML and FL presents a transformative approach to localization, particularly in resource-constrained and privacy-sensitive environments [4, 25]. Their combined strengths enable intelligent, distributed decision-making at the edge, offering a range of benefits:

- **Low Power Consumption:** TinyML models are optimized for ultra-low-power devices, allowing inference to run on microcontrollers and sensors with minimal energy usage [21]. This is crucial for battery-operated IoT devices deployed in remote or mobile settings, where frequent recharging or replacement is impractical.
- **Real-Time Processing:** By performing inference locally, edge devices can make immediate decisions without waiting for cloud responses [3]. This is essential for time-critical localization tasks such as autonomous vehicle navigation, drone flight control, or emergency response coordination.
- **Reduced Latency:** Eliminating the need to transmit raw data to centralized servers significantly reduces communication delays [32]. In localization, this translates to faster position updates and smoother tracking, especially in dynamic environments like smart cities or industrial automation.
- **Privacy and Security:** FL ensures that sensitive location data remains on-device, with only model updates shared during training [22]. This decentralized approach aligns with data protection regulations such as GDPR and enhances user trust in systems that handle personal or geospatial information.
- **Robustness in GNSS-Denied Areas:** In environments where GNSS signals are weak or unavailable—such as indoors, underground, or in urban canyons—TinyML can leverage sensor fusion (e.g., IMU, Wi-Fi, BLE) to maintain accurate localization [31]. FL further strengthens this by aggregating diverse data sources across devices [4].
- **Adaptive Learning:** FL enables models to continuously learn from local conditions without centralized retraining [24]. This adaptability is vital for localization systems operating in changing environments, such as seasonal variations, infrastructure updates, or evolving user behavior.
- **Scalability:** As the number of edge devices grows, FL scales naturally by distributing computation and learning across the network [33]. This decentralized architecture supports large-scale deployments in smart homes, logistics networks, and environmental monitoring systems, without overwhelming central infrastructure.

Collectively, these benefits position TinyML and FL as a powerful combination for building efficient, secure, and context-aware localization systems. Their adoption not only enhances technical performance but also reduces infrastructure costs and regulatory risks, making them ideal for next-generation IoT applications.

6 Challenges of TinyML and FL in Localization

Despite their transformative potential, the deployment of TinyML and Federated Learning (FL) in localization systems faces several technical and operational challenges. These limitations stem from the inherent constraints of edge devices, fragmented development ecosystems, and evolving regulatory landscapes. To promote practical adoption, this section outlines key challenges along with mitigation strategies that address both engineering and compliance concerns.

- **Limited Computational Resources:** Edge devices often operate with constrained memory, processing power, and energy budgets [21]. Running complex models or performing frequent updates can overwhelm these systems. Mitigation includes model compression techniques such as quantization, pruning, and knowledge distillation.
- **Heterogeneous Hardware and Data:** Devices differ in architecture, sensor quality, and environmental conditions, leading to inconsistent data distributions [24]. FL must account for non-IID (non-independent and identically distributed) data and device heterogeneity through personalized models and adaptive aggregation strategies.
- **Communication Overhead:** FL requires periodic transmission of model updates, which can strain bandwidth and battery life, especially in mobile or remote deployments [22]. Solutions include update sparsification, asynchronous communication, and federated dropout.

- **Privacy and Security Risks:** Although FL avoids raw data sharing, model updates can still leak sensitive information [23]. Techniques like secure aggregation, differential privacy, and homomorphic encryption help mitigate these risks.
- **Lack of Standardization:** The TinyML and FL ecosystems are fragmented, with diverse frameworks, protocols, and deployment pipelines [33]. This complicates integration and scalability. Emerging standards and cross-platform toolkits aim to unify development practices.
- **Regulatory Compliance:** Localization systems must adhere to data protection laws such as GDPR, especially when handling geospatial or biometric data [25]. FL supports compliance by keeping data local, but auditability and transparency remain critical.
- **Model Drift and Environmental Dynamics:** Changing layouts, signal interference, and user behavior can degrade model performance over time [4]. Continuous learning and federated adaptation mechanisms help maintain accuracy in dynamic settings.

6.1 Computational and Memory Limitations

Edge devices such as microcontrollers and embedded sensors often operate with limited memory, processing power, and storage capacity [21]. These constraints restrict the deployment of large or complex machine learning models, which are typically designed for cloud-scale environments. Optimization techniques such as quantization (e.g., int8 or float16), pruning, and knowledge distillation are essential to reduce model size and computational load. However, these techniques may introduce trade-offs between efficiency and accuracy, particularly in high-precision localization tasks [3].

Mitigation: Employ model compression frameworks such as the TensorFlow Model Optimization Toolkit [34] to systematically reduce model complexity while preserving performance. Lightweight architectures like MobileNet and Tiny-YOLO can also be adapted for localization use cases [33].

6.2 Energy Management

Battery-operated edge devices must balance computational demands with energy efficiency to ensure long-term usability, especially in remote or mobile deployments [21, 25]. High-frequency inference or continuous data collection can rapidly deplete power reserves, limiting the practicality of real-time localization [33].

Mitigation: Implement energy-aware techniques such as dynamic voltage and frequency scaling (DVFS), duty cycling, and event-triggered inference [3]. These strategies reduce power consumption by adjusting processing intensity based on contextual needs.

6.3 Toolchain and Standardization Issues

The edge ML ecosystem is fragmented, with a wide variety of hardware platforms, operating systems, and machine learning frameworks [33]. This diversity complicates model portability, integration, and maintenance across devices. Developers often face compatibility issues when deploying models trained in one environment to another [25].

Mitigation: Adopt open standards such as ONNX (Open Neural Network Exchange) to facilitate cross-platform model interoperability [35]. Frameworks like TensorFlow Lite and PyTorch Mobile also support deployment across heterogeneous edge environments [34].

6.4 Security and Privacy Vulnerabilities

Edge-deployed machine learning models are inherently exposed to security threats due to their decentralized nature and limited computational safeguards [22, 23]. Common vulnerabilities include adversarial attacks, model inversion, and data extraction. Federated Learning, while designed to preserve privacy by avoiding raw data transmission, introduces new risks such as model poisoning and gradient leakage [24].

In the context of the EU AI Act which enforces strict requirements for high-risk AI systems, including transparency, robustness, and cybersecurity [37, 38], addressing these vulnerabilities is both a technical and

legal imperative [36]. The Act emphasizes privacy-by-design, accountability, and risk mitigation, particularly for systems involved in sensitive tasks like localization [39, 40].

Mitigation: Integrate advanced privacy-preserving techniques such as homomorphic encryption to enable secure computation on encrypted data, and federated differential privacy to protect individual data contributions during collaborative training [23]. These methods align with the EU AI Act's provisions on data governance and system resilience. Table 1 summarizes key mitigation techniques and maps them to relevant articles of the EU AI Act [37, 40].

Figure 3 presents a comprehensive ethical framework that incorporates transparency, accountability, and human-centric principles into AI system development [43]. The figure illustrates how existing risks (dashed lines on the left) should be addressed and replaced by essential requirements (solid lines on the right).

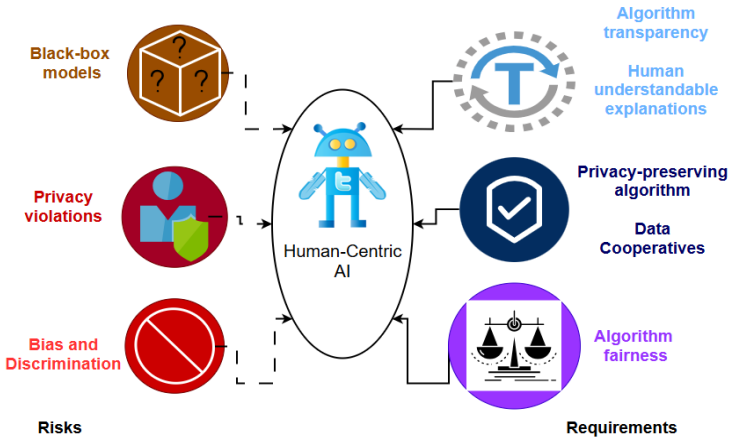


Figure 3: Human-centric AI, adapted from [43]

Table 1: Security and Privacy Mitigation Techniques in Compliance with the EU AI Act

Mitigation Technique	**Description**	**Relevant EU AI Act Provision**
Homomorphic Encryption	Enables computation on encrypted data without decryption, preserving confidentiality during model training and inference.	Article 15: Robustness and Accuracy
Federated Differential Privacy	Adds noise to local model updates to prevent leakage of individual data points during aggregation.	Article 10: Data Governance
Secure Aggregation Protocols	Ensures that model updates from multiple clients are aggregated securely without exposing individual contributions.	Article 9: Risk Management System
Adversarial Robustness Testing	Evaluates model resilience against adversarial inputs and poisoning attacks.	Article 15: Robustness and Accuracy
Model Explainability Tools	Enhances transparency by making model decisions interpretable to users and auditors.	Article 13: Transparency
Access Control and Audit Logging	Restricts unauthorized access and tracks system interactions to ensure accountability.	Article 14: Human Oversight

To ensure that AI systems contribute positively to society, it is important to embed ethical guidelines

throughout the AI lifecycle—from problem formulation and data collection to deployment and monitoring. This includes promoting transparency, implementing fairness-aware algorithms, safeguarding user privacy, and establishing clear accountability for outcomes generated by AI systems.

6.5 Heterogeneity and Convergence in Federated Learning

One of the fundamental challenges in FL is the heterogeneity of data across participating devices. In real-world localization scenarios, each edge device may collect data from vastly different environments—urban vs. rural, indoor vs. outdoor, or static vs. mobile contexts. This leads to non-independent and identically distributed (non-IID) data, which complicates model convergence and can degrade overall performance [22, 24].

Moreover, communication overheads pose additional barriers. Frequent synchronization between devices and the central server can be costly in terms of bandwidth and energy, especially in large-scale deployments [41]. These issues are exacerbated when devices operate asynchronously or intermittently due to connectivity constraints [23].

Mitigation: To address data heterogeneity, algorithms such as FedProx have been developed to improve convergence by introducing a proximal term that stabilizes local updates [42]. Additionally, communication-efficient strategies—such as gradient compression, update sparsification, and asynchronous aggregation—can significantly reduce overhead while maintaining model fidelity. These techniques can be particularly valuable in localization systems where devices vary in capability, connectivity, and data quality.

6.6 Accuracy–Efficiency Trade-offs and Hierarchical Federated Learning Architectures

Ultra-low-power edge devices employed in TinyML deployments impose strict constraints on memory, computational capacity, and energy consumption. These constraints necessitate the use of model compression and quantization techniques to enable on-device inference while maintaining acceptable performance levels [44, 45].

Post-training quantization, particularly the conversion from floating-point to 8-bit integer representations, typically results in a modest reduction in model accuracy, often in the range of 1–3% for classification tasks, while yielding substantial improvements in memory footprint, inference latency, and energy efficiency [44,46]. However, more aggressive optimization strategies—such as structured pruning, mixed-precision quantization, or extreme model compression—can introduce larger accuracy degradation. This effect is especially pronounced in regression-based localization models, where small numerical perturbations may propagate nonlinearly and lead to meter-level positioning errors [47–49]. Quantization-aware training (QAT) partially mitigates these effects by incorporating quantization noise during training, thereby improving robustness compared to post-training quantization alone [46].

Federated learning (FL) has emerged as a promising paradigm for collaborative model training across distributed edge devices while preserving data privacy by keeping raw data localized [50, 51]. Despite its advantages, conventional FL approaches such as FedAvg incur significant communication overhead due to frequent global model aggregation, which becomes a scalability bottleneck in dense IoT localization deployments [50].

Hierarchical federated learning (HFL) architectures address this limitation by introducing intermediate aggregation layers, such as edge gateways or cluster heads, between local devices and the central server [52,53]. By aggregating local updates at intermediate nodes before forwarding them upstream, HFL significantly reduces communication frequency and bandwidth requirements while preserving convergence stability. This hierarchical paradigm is particularly well suited for large-scale IoT localization systems, where devices exhibit heterogeneous computational resources, non-identically distributed data, and intermittent connectivity [51, 53].

From a regulatory perspective, the combination of on-device TinyML and hierarchical FL aligns well with the principles of the EU AI Act, including data minimization, robustness, and accountability. By reducing centralized data exposure and enabling localized risk management, these architectures support the development of trustworthy and compliant AI-enabled localization systems [54]. Overall, balancing accuracy

degradation introduced by TinyML optimizations with communication-efficient hierarchical learning architectures remains a critical research direction for scalable, privacy-preserving, and energy-efficient localization in future IoT systems.

7 Future Research Directions

Future work should focus on Federated TinyML for RF fingerprinting-based localization, utilizing ultra-low-cost IoT devices for collaborative model building [25,28]. Devices perform local feature extraction from RSSI or CSI using simple algorithms, contributing updates via FL to a global model [4].

Advantages include scalability with device density, energy efficiency, robustness to changes, and privacy preservation. Challenges encompass minimizing communication overhead, handling non-IID data, and enhancing security [24, 41].

Exploring hierarchical FL architectures, advanced aggregation techniques, and integration with emerging hardware could further advance adaptive, pervasive localization systems for smart cities and industrial IoT [3]. Additional directions include quantum-resistant security for FL [38] and AI-driven hyperparameter tuning for TinyML models [33].

8 Conclusion

The convergence of TinyML and Federated Learning (FL) marks a pivotal shift in how localization systems are designed and deployed. By enabling intelligent, privacy-preserving, and energy-efficient inference directly on edge devices, this paradigm supports scalable and adaptive solutions across diverse environments—from smart cities to industrial IoT.

Throughout this work, we have explored key applications such as RF fingerprinting, asset tracking, and seamless indoor–outdoor positioning, highlighting the technical benefits and operational advantages of decentralized learning. We also examined the challenges posed by resource constraints, data heterogeneity, and evolving regulatory frameworks, offering mitigation strategies grounded in current research and best practices.

Looking ahead, future research should prioritize federated TinyML architectures tailored for ultra-low-cost devices, robust security mechanisms aligned with global AI regulations, and intelligent model optimization techniques. These innovations will be essential for realizing resilient, context-aware localization systems that meet the demands of next-generation IoT ecosystems.

The integration of TinyML and FL has the potential to not only enhances localization performance but also redefines the boundaries of edge intelligence—ushering in a new era of autonomous, trustworthy, and human-centric computing.

References

[1] Abreha, H. G., Hayajneh, M., & Serhani, M. A. (2022). Federated Learning in Edge Computing: A Systematic Survey. *Sensors*, 22(2), 450.

[2] Mughal, F. R., He, J., Das, B., et al. (2024). Adaptive Federated Learning for Resource-Constrained IoT Devices through Edge Intelligence and Multi-Edge Clustering. *Scientific Reports*, Nature Publishing Group.

[3] Ren, H., Anicic, D., & Runkler, T. A. (2023). TinyReptile: TinyML with Federated Meta-Learning. In *International Joint Conference on Neural Networks (IJCNN)*. arXiv:2304.05201. https://doi.org/10.48550/arXiv.2304.05201

[4] Ficco, M., Guerriero, A., Milite, E., Palmieri, F., Pietrantuono, R., & Russo, S. (2024). Federated Learning for IoT Devices: Enhancing TinyML with On-Board Training. *Information Fusion*, 104, 102189. https://doi.org/10.1016/j.inffus.2023.102189

[5] Etiabi, Y., Njima, W., & Amhoud, E. M. (2024). FeMLoc: Federated Meta-learning for Adaptive Wireless Indoor Localization Tasks in IoT Networks. *arXiv preprint arXiv:2405.11079*. https://doi.org/10.48550/arXiv.2405.11079

[6] Ye, F., Wang, R., Tang, S., Duan, S., & Xu, C. (2023). Federated Learning-Enabled Cooperative Localization in Multi-agent System. *International Journal of Wireless Information Networks*, 31, 61–72. https://doi.org/10.1007/s10776-023-00610-0

[7] Dritsas, E., & Trigka, M. (2025). Federated Learning for IoT: A Survey of Techniques, Challenges, and Applications. *Journal of Sensor and Actuator Networks*, 14(1), 9. https://doi.org/10.3390/jsan14010009

[8] R. Shahbazian, G. Macrina, E. Scalzo, and F. Guerriero, "Machine Learning Assists IoT Localization: A Review of Current Challenges and Future Trends," Sensors 2023, 23, 3551. https://doi.org/10.3390/s23073551.

[9] Filho, C. P., Marques Jr., E., Chang, V., dos Santos, L., Bernardini, F., Pires, P. F., Ochi, L., & Delicato, F. C. (2022). A Systematic Literature Review on Distributed Machine Learning in Edge Computing. *Sensors*, 22(7), 2665. https://doi.org/10.3390/s22072665

[10] Truong, H.-L., Truong-Huu, T., & Cao, T.-D. (2022). Making Distributed Edge Machine Learning for Resource-Constrained Communities and Environments Smarter: Contexts and Challenges. *Journal of Reliable Intelligent Environments*, 9, 119–134. https://doi.org/10.1007/s40860-022-00176-3

[11] Ray, P. P. (2022). A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4), 1595–1623. https://doi.org/10.1016/j.jksuci.2021.11.019

[12] Khouas, A. R., Bouadjenek, M. R., Hacid, H., & Aryal, S. (2024). Training Machine Learning Models at the Edge: A Survey. *arXiv preprint arXiv:2403.02619*. https://arxiv.org/pdf/2403.02619

[13] Patil, S. G. (2024). Systems for Machine Learning on Edge Devices. Master's thesis, University of California, Berkeley. EECS Department Technical Report No. UCB/EECS-2024-1. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-1.pdf

[14] Amaral, R., Murthy, A., Stronach, M., Jain, K., Khurmi, K., & Murthy, P. (2024). Train, Optimize, and Deploy Models on Edge Devices Using Amazon SageMaker and Qualcomm AI Hub. *AWS Machine Learning Blog*. https://aws.amazon.com/blogs/machine-learning/train-optimize-and-deploy-models-on-edge-devices-using-amazon-sagemaker-and-qualcomm-ai-hub/

[15] Qualcomm Developer Blog, *Optimizing Your AI Model for the Edge*, June 2025. Available: `https://www.qualcomm.com/developer/blog/2025/06/optimizing-your-ai-model-for-the-edge`

[16] Google AI for Developers, *Model Optimization — Google AI Edge*, 2025. Available: `https://ai.google.dev/edge/litert/models/model_optimization`

[17] Xubin Wang and Weijia Jia, *Optimizing Edge AI: A Comprehensive Survey on Data, Model, and System Strategies*, arXiv preprint arXiv:2501.03265, Jan. 2025. Available: `https://arxiv.org/abs/2501.03265`

[18] Infoservices Team, *Edge AI + Multi-Cloud: The Future of Smart Infrastructure*, May 2025. Available: `https://blogs.infoservices.com/embedded-system/edge-ai-multi-cloud-smart-infrastructure/`

[19] Mahesh Vaijainthymala Krishnamoorthy, *Edge AI: TensorFlow Lite vs. ONNX Runtime vs. PyTorch Mobile*, DZone, June 2025. Available: `https://dzone.com/articles/edge-ai-tensorflow-lite-vs-onnx-runtime-vs-pytorch`

[20] Dinesh Subramani and Samir Araujo, *Demystifying Machine Learning at the Edge Through Real Use Cases*, June 2022. Available: `https://aws.amazon.com/blogs/machine-learning/demystifying-machine-learning-at-the-edge-through-real-use-cases/`. Accessed: 2025-09-24.

[21] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017. doi: 10.1109/JPROC.2017.2761740.

[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. of AISTATS*, 2017.

[23] Mikko A. Heikkilä, "On Using Secure Aggregation in Differentially Private Federated Learning with Multiple Local Steps," arXiv preprint arXiv:2407.19286, Mar. 2025. Available: `https://arxiv.org/html/2407.19286v2`

[24] Bifta Sama Bari and Kumar Yelamarthi, "Advancing Federated Learning: A Comprehensive Solution for Model Aggregation, Heterogeneity, Privacy, and Security," *SN Computer Science*, vol. 6, article no. 411, Apr. 2025. Available: `https://link.springer.com/article/10.1007/s42979-025-03934-1`

[25] Leandro Antonio Pazmiño Ortiz, Ivonne Fernanda Maldonado Soliz, and Vanessa Katherine Guevara Balarezo, "Advancing TinyML in IoT: A Holistic System-Level Perspective for Resource-Constrained AI," *Future Internet*, vol. 17, no. 6, article 257, June 2025. doi: 10.3390/fi17060257.

[26] Thanaphon Suwannaphong, Ferdian Jovan, Ian Craddock, and Ryan McConville, "Optimising TinyML with Quantization and Distillation of Transformer and Mamba Models for Indoor Localisation on Edge Devices," arXiv preprint arXiv:2412.09289, Dec. 2024. Available: `https://arxiv.org/pdf/2412.09289`

[27] B. Yuen, Y. Bie, D. Cairns, G. Harper, J. Xu, C. Chang, X. Dong, and T. Lu, "Wi-Fi and Bluetooth Contact Tracing Without User Intervention," *IEEE Access*, vol. 10, pp. 91027–91044, 2022. doi: 10.1109/ACCESS.2022.3201645.

[28] Marco Zennaro, Diego Méndez, Moez Altayeb, Daniel Crovo, and Gianpaolo Manzoni, "Indoor Positioning Systems: Edge-Based RF Fingerprinting," SciTinyML Workshop, Harvard University, May 2024. Available: `https://tinyml.seas.harvard.edu/SciTinyML-24/assets/slides/5-Diego-Location-Fingerprinting.pdf`

[29] Jared Maks, *TinyML on the Edge: IMU Sensors on Arduino Nano 33 BLE Sense*, GitHub repository, 2025. Available: `https://github.com/jaredmaks/tinyml-on-the-edge`

[30] F. Zafari, A. Gkelias, and K. K. Leung, "A Survey of Indoor Localization Systems and Technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019. doi: 10.1109/COMST.2019.2911558

[31] Su Liu, *Seamless Indoor–Outdoor Positioning Using TinyML and Sensor Fusion*, MASc Thesis, McMaster University, April 2023. Available: `https://macsphere.mcmaster.ca/bitstream/11375/28513/2/Liu_Su_202304_MASc.pdf`

[32] D. Avellaneda, D. Méndez, and G. Fortino, "A TinyML Deep Learning Approach for Indoor Tracking of Assets," *Sensors*, vol. 23, no. 3, article 1542, Jan. 2023. doi: 10.3390/s23031542

[33] Rakhee Kallimani, Krishna Pai, Prasoon Raghuwanshi, Sridhar Iyer, and Onel L. A. López, "TinyML: Tools, Applications, Challenges, and Future Research Directions," *Multimedia Tools and Applications*, 2023. doi: 10.1007/s11042-023-16740-9

[34] TensorFlow Team, *TensorFlow Model Optimization Toolkit*, 2025. Available: `https://www.tensorflow.org/model_optimization`

[35] ONNX Community, *Open Neural Network Exchange (ONNX)*, 2025. Available: `https://onnx.ai`

[36] European Commission, *Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act)*, 2024. Available: `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021PC0206`

[37] European Parliament, "EU AI Act: First regulation on artificial intelligence," 2024. Available: https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence. Accessed: 2025-05-28.

[38] Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev, "Beyond Memorization: Violating Privacy Via Inference with Large Language Models," arXiv preprint arXiv:2310.07298, 2024. Available: https://arxiv.org/abs/2310.07298

[39] Donghyeok Lee, Christina Todorova, and Alireza Dehghani, "Ethical Risks and Future Direction in Building Trust for Large Language Models Application under the EU AI Act," in *Proc. of the 2024 Conference on AI Regulation*, Dec. 2024, pp. 41–46. doi: 10.1145/3701268.3701272

[40] European Commission, "General-Purpose AI Code of Practice now available," 2025. Available: https://ec.europa.eu/commission/presscorner/detail/en/ip_25_1787. Accessed: 2025-07-26.

[41] Peter Kairouz et al., "Advances and Open Problems in Federated Learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021. doi: 10.1561/2200000083

[42] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith, "Federated Optimization in Heterogeneous Networks," in *Proc. of MLSys*, 2020. Available: https://arxiv.org/abs/1812.06127

[43] Bruno Lepri, Nuria Oliver, and Alex Pentland, "Ethical machines: The human-centric use of artificial intelligence," *iScience*, vol. 24, no. 3, article 102249, 2021. doi: 10.1016/j.isci.2021.102249. Available: https://www.sciencedirect.com/science/article/pii/S2589004221002170

[44] C. R. Banbury, V. J. Reddi, P. T. Chaudhary, et al., "Benchmarking TinyML Systems: Challenges and Direction," *arXiv preprint arXiv:2102.07638*, pp. 1–13, 2021.

[45] Kunran Xu, Huawei Zhang, Yishi Li, Yuhao Zhang, Rui Lai, Yi Liu, "An Ultra-low Power TinyML System for Real-time Visual Processing at Edge," arXiv:2207.04663, 2023.

[46] B. Jacob, S. Kligys, B. Chen, et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," *arXiv:1712.05877* , 2017.

[47] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv:1510.00149*, 2016.

[48] K. Bregar and M. Mohorčič, "Improving Indoor Localization Using Convolutional Neural Networks on Computationally Restricted Devices," in IEEE Access, vol. 6, pp. 17429-17441, 2018, doi: 10.1109/ACCESS.2018.2817800.

[49] F. Zafari, A. Gkelias, and K. K. Leung, "A Survey of Indoor Localization Systems and Technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.

[50] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, 2017.

[51] P. Kairouz, H. B. McMahan, B. Avent, et al., "Advances and Open Problems in Federated Learning," arXiv:1912.04977 10.1561/9781680837896, 2021.

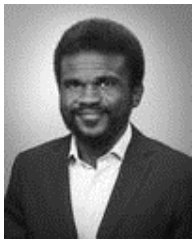
[52] C. Briggs, Z. Fan, and P. Andras, "Federated Learning with Hierarchical Clustering of Local Updates to Improve Training on Non-IID Data," *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, 2020.

[53] Qiu, C., Wu, Z., Wang, H., Yang, Q., Wang, Y., Su, C. (2025). Hierarchical Aggregation for Federated Learning in Heterogeneous IoT Scenarios: Enhancing Privacy and Communication Efficiency. Future Internet, 17(1), 18. https://doi.org/10.3390/fi17010018

[54] European Parliament and Council of the European Union, "Regulation (EU) 2024/1689 Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act)," Official Journal of the European Union, 2024. https://eur-lex.europa.eu/EN/legal-content/summary/rules-for-trustworthy-artificial-intelligence-in-the-eu.html

Authors Biography

Akpojoto A. Siemuri



Akpojoto A. Siemuri received his B.S. degree in Electrical and Computer Engineering from the Federal University of Technology Minna, Nigeria, in 2010, and his M.S. degree in Wireless Industrial Automation with a minor in Industrial Management from the University of Vaasa, Finland, in 2019. He is currently pursuing a Ph.D. in Automation Technology at the University of Vaasa.

From 2018 to 2019, he worked as a Research Assistant in the Smart Energy Systems Research Platform (SESP) project at the University of Vaasa. He also served as a Project Researcher in Machine Learning and Global Navigation Satellite System (GNSS) Technologies at the Digital Economy Platform, University of Vaasa from 2020 - 2025 and now works as a Data and AI Engineer at Mirka Oy from 2024 to date. His research interests include machine learning, GNSS technologies, smart devices, embedded systems, communication systems, and game theory.

Prof. Mohammed S. Elmusrati

Prof. Mohammed S. Elmusrati received his B.Sc. (with honors) and M.Sc. (with high honors) degrees in Electrical and Electronic Engineering from the University of Benghazi, Libya, in 1991 and 1995, respectively. He earned his Licentiate of Science in Technology (with distinction) and Doctor of Science in Technology (D.Sc.) degrees in Automation and Control Engineering from Aalto University, Finland, in 2002 and 2004, respectively. He is currently a Full Professor and Head of the Digitalization Unit at the School of Technology and Innovations, University of Vaasa, Finland.

His research interests include wireless communications, artificial intelligence, machine learning, biotechnology, big data analysis, stochastic systems, and game theory. Prof. Elmusrati has authored more than 130 papers, books, and book chapters. He is an active member of several scientific societies, including Senior Member of IEEE, Member of the Society for Industrial and Applied Mathematics (SIAM), and Member of the Finnish Automation Society.