Jari Isohanni

# Recognition of Subtle Colour Differences

A Comparative Study of Machine Learning and Colour Difference Metrics

ACTA WASAENSIA 557

University of Vaasa
VAASAN YLIOPISTO

ISBN 978-952-395-198-3 (print)
978-952-395-199-0 (online)
ISSN 0355-2667 (Acta Wasaensia 557, print)
2323-9123 (Acta Wasaensia 557, online)
URN http://urn.fi/URN:ISBN:978-952-395-199-0

PunaMusta Oy, Joensuu, 2025.

ACADEMIC DISSERTATION

*To be presented, with the permission of the Board of the School of Technology and Innovations of the University of Vaasa, for public examination on the 3rd of September, 2025, at noon.*

Dissertation of the School of Technology and Innovations at the University of Vaasa
in the field of computer science


| | |
|---|---|
| Author | Jari Isohanni, https://orcid.org/0000-0002-7154-2515 |
| Supervisor(s) | Associate Professor Jani Boutellier<br>University of Vaasa, School of Technology and Innovations, Computer Science |
| | University Lecturer Birgitta Martinkauppi<br>University of Vaasa, School of Technology and Innovations, Energy Technology |
| Custos | Associate Professor Jani Boutellier<br>University of Vaasa, School of Technology and Innovations, Computer Science |
| Reviewers | Professor of Biomedical Engineering Tapio Seppänen<br>University of Oulu, Faculty of Information Technology and Electrical Engineering |
| | Professor of Electrical Engineering Mattias O'Nils<br>Mid Sweden University, Department of Computer and Electrical Engineering (DET) |
| Opponent | Associate Professor Miguel Bordallo Lopez<br>University of Oulu, Center for Machine Vision and Signal Analysis (CMVS) |

# TIIVISTELMÄ

Väitöskirjassa tutkitaan hienovaraisten värisävyerojen tunnistamista eri menetelmillä. Hienovaraiset värisävyerot ovat tärkeitä eri sovelluksissa, kuten terveydenhuollossa, maataloudessa ja elintarviketeollisuudessa. Värisävyerojen avulla voidaan luokitella ja tunnistaa kuvista kiinnostavia piirteitä. Tässä väitöskirjassa keskitytään hienovaraisten värisävyerojen tunnistamiseen painotuotteista, ja yksi tutkimuksen sovelluskohteista ovat ns. funktionaaliset musteet. Funktionaalisten musteiden avulla voidaan valmistaa edullisia ei-elektronisia indikaattoreita, jotka ilmoittavat esimerkiksi tuotteen lämpötilan tai kosteuden värinmuutoksen avulla.

Väitöskirjassa käytetään värisävyerojen laskenta-algoritmeja sekä valvomatonta (unsupervised) ja valvottua (supervised) koneoppimista hienovaraisten värisävyerojen tunnistamiseen. Värisävyerojen laskennassa käytetään uusimpia ja tarkoitukseen parhaiten soveltuvia matemaattisia kaavoja. Valvomattomissa menetelmissä hyödynnetään erilaisia ryhmittelymenetelmiä (clustering). Valvottujen menetelmien osalta tutkitaan yleisimpiä neuroverkkorakenteita (convolutional neural network). Väitöskirjan kokeellisissa osissa pyritään löytämään ne menetelmät, joilla hienovaraiset sävyerot voidaan tunnistaa parhaiten.

Värisävyerojen laskenta-algoritmit pystyvät tunnistamaan riittävän suuria värieroja todellisissa käyttötapauksissa. Ryhmittelymenetelmät toimivat tarkemmin kuin laskenta-algoritmit ja mahdollistavat myös pienempien värisävyerojen havaitsemisen. Parhaat tulokset saavutettiin konvoluutioneuroverkoilla, joista ResNet-34 osoittautui testeissä tarkimmaksi. Tätä arkkitehtuuria muokattiin edelleen käyttötarkoitukseen sopivammaksi. Sopivin neuroverkko saatiin, kun viimeinen yhdistävä kerros muutettiin keskiarvolaskennasta maksimilaskentaan. Tässä arkkitehtuurissa käytettiin myös gradienttien keskittämistä osana oppimisprosessin takaisinkytkentää.

Eri menetelmiin vaikuttavat merkittävästi kuvien häiriöt ja kuvien laatu. Kuvien esikäsittelyllä on tärkeä rooli, kun väitöskirjassa esiteltyjä menetelmiä otetaan käyttöön eri sovelluksissa. Käyttötarkoituksesta riippuen ryhmittelymenetelmät voivat tarjota paremman hyöty–panossuhteen kuin monimutkaisemmat neuroverkot. Tämä johtuu siitä, että ryhmittelymenetelmät eivät vaadi aineiston keruuta tai neuroverkon kouluttamista. Nämä menetelmät eivät myöskään pyri oppimaan painotuotteen paperin ominaisuuksia.

Avainsanat: tekoäly, koneoppiminen, väriero

# ABSTRACT

The dissertation investigates the identification of subtle colour differences using different methods. Subtle colour differences are important in applications such as healthcare, agriculture, and the food industry. Colour differences can be used to classify and identify features of interest in images. In this dissertation, the focus is on identifying subtle colour differences in prints, and one use for the results of the dissertation are functional inks. Functional inks can be used to create inexpensive non-electronic indicators that can trough colour change tell the product's temperature or humidity, for example.

In this dissertation, colour difference calculation algorithms as well ass unsupervised, and supervised machine learning methods are used to identify subtle colour differences. The colour difference algorithms use the newest and most suitable formulas developed. For unsupervised methods, different clustering algorithms were used. For supervised learning, the most common architectures were explored. In the various experiments in the dissertation, the aim was to find the methods that are ideal for identifying subtle colour differences.

Colour difference algorithms can be used to identify colour differences that are significant enough in real-life use cases. Unsupervised methods work better than colour difference algorithms and can be used to identify smaller differences. The best results were achieved with convolutional neural networks, of which ResNet-34 proved to be the most accurate in the tests. This architecture was further modified to better suit the use case. The best architecture turned out to be the version in which the last connecting layer was changed from average pooling to maximum pooling. In this architecture, gradient centralisation was also used as part of the backpropagation of the learning process.

The different methods are significantly affected by the disturbances in the images and the quality of the images. Image preprocessing plays an important role when the method presented in the dissertation is put into practice. Depending on the use case, unsupervised clustering methods can lead to a better effort-outcome ratio than more complex supervised neural networks. This is due to the fact that unsupervised methods do not necessitate the prior collection of labeled datasets or the training of neural networks. Moreover, these methods do not explicitly aim to learn or model the underlying structure of the input data.

Keywords: artificial intelligence, machine learning, colour difference

# ACKNOWLEDGEMENTS

First of all, I would like to thank Kimmo Svinhufvud for his encouraging words during the ”Scientific Writing” course at the Kokkola University Consortium Chydenius — the course that sparked the beginning of this journey. It’s hard to believe how quickly time has passed since 2020, back when everything sounded so simple, it was not. Another person who had a major influence on starting this process, thank you former Rector / CEO of Centria University of Applied Sciences Kari Ristimäki, you nudged me over the start line. I am also grateful to our current rectory of Tapio Huttula, Jennie Elfving and Marko Forsell for supporting employees towards new degrees. Warm thanks to my stydying colleagues — and hopefully future doctors — Annika and Johanna, for sharing their insights and knowledge.

This dissertation would not have been possible without the ideas and innovative thinking of Sture Udd and UPC Konsultointi Oy. The visionary development of functional inks provided a real-world use case that anchored this thesis in practical relevance. I also extend my sincere thanks to VTT Technical Research Centre of Finland and Liisa Hakola for the sample materials, which were essential in carrying out the experiments.

Deep respect go to Kai Hermsen and Lorna Goulden — I can’t express how grateful I am that you kept the Twinds Foundation going strong while I focused on my studies.

The first article of this thesis was written under the supervision of Professor Johan Lilius from Åbo Akademi. I am thankful for the support I received during those early and important steps of the doctoral process. The initial phases of this work were financially supported by the Central Ostrobothnia Regional Fund of the Finnish Cultural Foundation — thank you for your trust.

The main supervisory effort was led by Associate Professor Jani Boutellier and University Lecturer Birgitta Martinkauppi from the University of Vaasa. Thank you both for your continuous guidance, thoughtful feedback, and unwavering support throughout this journey. I would not have been connected with the University of Vaasa without Professor Heidi Kuusniemi and her presentation in the Digitalisation Committee of the Ostrobothnia Chamber of Commerce — thank you for that connection.

I would also like to express my sincere gratitude to the two pre-examiners: Professor Tapio Seppänen from the University of Oulu and Professor Mattias O’Nils from Mid Sweden University. Your accurate comments and suggestions significantly im-

proved the quality of this thesis in its final stage. Dear Associate Professor Miguel Bordallo Lopez, thank you for being my opponent, your respectful and insightful feedback contributed meaningfully to the academic depth of this dissertation.

To my parents, Harri and Ulla, as well as my grandmother Kaisu, aunt Elisa and uncle Kauko — thank you for your continuous support and genuine curiosity about the progress of this work.

And finally, the most important, my heartfelt thanks go to Tanja, Aaron, and Leevi. You brought balance and perspective to this journey. Your everyday presence gave me strength to finish this thesis — and reminded me of what truly matters. I hope that one day I'll have the honour of attending your own dissertations.

# CONTENTS

# Figures

# Tables

# LIST OF PUBLICATIONS

The dissertation is based on the following four refereed articles:

(I) Jari Isohanni, Use of Functional Ink in a Smart Tag for Fast-Moving Consumer Goods Industry, **Journal of Packaging Technology and Research**, 6, 187–198, (2022). https://doi.org/10.1007/s41783-022-00137-4 © 2022, The Author. Published by Springer Nature. CC BY.

(II) Jari Isohanni, Recognising small colour changes with unsupervised learning, comparison of methods, **Advances in Computational Intelligence**, 4, 6, (2024). https://doi.org/10.1007/s43674-024-00073-7 © 2024, The Author. Published by Springer Nature. CC BY.

(III) Jari Isohanni, Using convolutional neural networks to classify subtle colour differences (In peer-review at ELCVIA, Electronic Letters on Computer Vision and Image Analysis journal)

(IV) Jari Isohanni, Customised ResNet architecture for subtle colour classification, **International Journal of Computers and Applications**, (2025). https://doi.org/10.1080/1206212X.2025.2465727 © 2025, The Author. Published by Taylor & Francis. CC BY.

*All of the articles are reprinted with the permission of the copyright owners.*

# AUTHOR'S CONTRIBUTION

### Publications I - IV

The author has contributed to the articles of this dissertation, having independently conceived, conducted, and analysed the research, as well as having authored the manuscripts in their entirety. This work represents the author's original contributions to the scientific literature, reflecting the author's experience and dedication to advancing knowledge in the field.

## Abbreviations

| | |
|---|---|
| ADC | Analog-to-digital converter |
| Adam | Adaptive Moment Estimation |
| Adagrad | Adaptive Gradient Algorithm |
| AI | Artificial intelligence |
| AMI | Adjusted Mutual Index |
| ANN | Artificial Neural Network |
| ARI | Adjusted Rand Index |
| AUC | Area under the curve |
| BERT | Bidirectional Encoder Representations from Transformers |
| BIRCH | Balanced Iterative Reducing and Clustering using Hierarchies |
| CCD | Charge-coupled device |
| CFA | Colour Filter Array |
| CFM | Colour Filter Mosaic |
| CIE | Commission Internationale de Photométrie |
| CMOS | Complementary metal-oxide-semiconductor |
| CMYK | Cyan magenta yellow black colour space |
| CNN | Convolutional Neural Network |
| $CO_2$ | Carbon oxide |
| DBI | Davies-Bouldin Index |
| DBSCAN | Density-based spatial clustering of applications with noise |
| DCT | Discrete Cosine Transform |
| DI | Dunn's index |
| DSC | Digital Still Camera |
| EMCCD | Electron Multiplying CCD |
| EU | European Union |
| FMI | Fowlkes-Mallows Index |
| FPN | Fixed pattern noise |
| GAN | Generative adversarial network |
| GD | Gradient Descent |
| GMM | Gaussian mixture model |
| HVS | Human Vision System |
| ICA | Independent Component Analysis |
| ICC | International Color Consortium |
| IoT | Internet of Things |
| IR | Infrared |
| ISO | International Organization for Standardization |
| JPEG | Joint Photographic Experts Group |
| LAB | LAB colour space |
| LED | Light emitting diode |
| LSTM | long short-term memory networks |
| LOF | Local Outlier Factor |

| | |
|---|---|
| LOOCV | Leave-One-Out Cross Validation |
| nm | Nanometre |
| MOSFET | Metal-oxide-semiconductor field-effect transistor |
| MI | Mutual Index |
| ML | Machine learning |
| MLP | Multilayer perceptron |
| NAG | Nesterov Accelerated Gradient |
| NLP | Natural Language Processing |
| NN | Neural Net |
| MAE | Mean absolute error |
| MSE | Mean squared error |
| MVCOR | Correlation Coefficients for Multivariate Data |
| OPTICS | Ordering Points To Identify the Clustering Structure |
| PCA | Principal Component Analysis |
| PCB | Printed Circuit Board |
| PRNU | Photo-response nonuniformity noise |
| QR code | Quick Response code |
| RGB | Red green blue colour space |
| ReLU | Rectified Linear Unit |
| RI | Rand Index |
| RMSE | Root Mean Squared Error |
| RMSprop | Root Mean Square Propagation |
| RMSSTD | Root-mean-square standard deviation |
| RNN | Recurrent neural networks |
| ROC | Receiver operating characteristic curve |
| RS | R-squared |
| sCMOS | Scientific CMOS |
| SDD | Slope Difference Distribution |
| SGD | Stochastic Gradient Descent |
| SLR camera | Single-lens reflex camera |
| SVM | Support Vector Machine |
| tSNE | t-Distributed Stochastic Neighbour Embedding |
| UV | Ultraviolet |
| VGG | Visual Geometry Group |
| WB | White balance |
| WCSS | Within-Cluster Sum of Squares |

# 1 INTRODUCTION

The recognition of colours, their classification, and separation is useful for many industries, including healthcare, the food industry, and manufacturing. This research was originally sparked by the fast-moving consumer goods (FMCG) and food industry sectors. Especially in these industries, consumers and end users will be more interested in the life cycle of products they use, and the exchange of information between consumers and producers has a high value (Golan et al., 2004; Isohanni, 2022). Consumers and industry are also more aware of $CO_2$ emissions, and especially in the food industry, of the safety of food and its sources. In response, an increasing level of data needs to be collected and used. IoT devices and printed electronics are looking to create data from various life-cycle events, but the prices of these devices are too high for certain low-cost items. Also, integration to most consumer items is not possible as many items have a short life-span and their unit price is low. In such cases, printed indicators with functional inks might play an important role. Functional inks work by interpreting their status through colour change; but reading these colour values with mobile devices is currently not reliable or accurate, especially when subtle changes are considered.

Colour recognition in controlled environments, or with clearly separate colours, is a trivial task in computer vision. Colour recognition can be done either by using colour difference algorithms or by machine learning. When colours become closer to each other, the task becomes more challenging and achieving the same accuracy as human vision is not certain. This applies especially in real-life use cases, and these subtle colour changes can be meaningful, as shown later in the dissertation. The word 'subtle' is not a scientific term and must be defined in more detail when used. In this dissertation, colours are seen to have a subtle difference if their Delta-E ($\Delta$E) is below 10. Delta-E is a metric that quantifies the difference between two colours in a colour space, typically LAB (Luo, Cui, & Rigg, 2001). Delta-E is widely used in industries such as printing, textile industry, healthcare, and digital imaging to measure colour accuracy.

An example of the role of colour change is presented in Figure 1. where the thermochromic ink works as an indicator. In this example, the indicator gives consumer information on whether the beer is at the correct drinking temperature by interpreting the temperature of the bottle. If the bottle is too warm (more than 7°C), the green bar at the bottom of the Datamatrix becomes transparent.

Functional inks have the characteristics of working through chemical reactions that change the state of the optical properties of the ink – i.e. the colour. Chemical reactions are influenced by current environmental conditions such as temperature, humidity, and some gasses. The functional inks appear in two stages; the stage that applies when condition A is valid, and the stage that applies in condition B. For example, the functional ink can appear transparent when the temperature is below


**Figure 1.** Thermochromic indicator in a beer bottle (photo: UpCode Ltd).

60°C, and red when the temperature is above 60°C (Harvey, 2007). A good example of a functional ink is the thermochromic ink used in the TagItSmart Horizon2020 project (Figure 2). This ink changes colour if the temperature of the ink is above or below 7°C. In this demonstration, the ink reversibly changes colour, but the change can also be irreversible (Bilgin & Backhaus, 2017). Functional inks have the advantage of being non-electronic which helps their recycling, and they are also suited for high-speed printing processes (conventional, non-conventional, and hybrid), which makes them suitable for many products. The mentioned properties are useful when consumers are the main user group, as functional inks will not require consumers to take any extra action. The indicator colour can be read either by visual inspection or by a mobile application (Isohanni, 2022).


**Figure 2.** Example of the colour change in functional inks.

Mobile devices are now used by almost all consumers, which makes them attractive for use in different applications. If functional ink is used in consumer products, it would enable consumers or other end-users to read indicator data using smartphones. This requires accurate colour recognition so that the consumer / end-user

can trust the data provided by the indicator. In recent history, we have seen many use cases in which mobile devices have been used for colour recognition. In their research, Angelico et al. (2021) developed an app to recognise a limited range of infant stool colours with mobile devices, and the accuracy of their solution was almost 100%. Yang et al. (2021) proposed a solution that can estimate the organic matter content of the soil under ideal lighting and soil preparation conditions; but faced a challenge of sharing the parameters between devices. Yulita, Amri, and Hidayat (2023) trained a DenseNet model for the detection of tomato leaf disease using a mobile application for image capture and achieved 95.7% accuracy. Terensan, Salgadoe, Kottearachchi, and Weerasena (2024) showed the power of smart phones and developed an app that can identify blast and brown spot diseases by using K-means clustering, and their app worked with an accuracy of 84.3% and could identify diseases without the knowledge of experts.

As shown in the previous research mentioned above, mobile devices have great potential in both colour recognition and in the recognition of colour differences. The characteristics of the devices and cameras must be taken into account when solutions are developed, and appropriate methods must be used to ensure the accuracy and reliability of the solution. These past applications also show that colour recognition can be done in many different ways. However, the previous mentioned and other research have focused on various other domains of colour recognition than printed ones, and the earlier research is more carefully examined in Section 1.1.

Printed colours depend on the printer (and its settings), ink, paper, and printing conditions, and combination defines how the final colour appears (Mangin & Silvy, 1997). If multiple printers are given the same command 'print 100% blue', the results will vary between printers, and even with the same printer, different results are produced over time.

In this research, the colours are recognised from printed samples (Figure 3). In the samples used, the colours are embedded inside a QR code for two reasons. Firstly, the areas within the QR code can be easily extracted for further processing. In addition, QR codes simulate the usage of functional inks as part of the item label, and in terms of the presented research, make it more efficient to incorporate the results of this dissertation into production (Isohanni, 2022).


**Figure 3.** Example of the dataset used.

The presented use case is prone to noise that comes from the structure of the paper, the printing quality, and the imaging pipeline. In optimal conditions, the colours

printed on the paper substrate would be constant throughout the surface. However, when the ink intensity becomes lower, the structure of the paper becomes more visible, leading to more noise. Another noise source is the quality of the print, and depending on the printing method used, the pattern of the print can produce noise and make the colour appear uneven (Figure 4). The print pattern can be so unique that it can be used to identify the printer (Ali et al., 2004). Partly, these sources of noise can be managed by image processing methods, and this dissertation looks at whether the methods used can manage noise as well. The last source of noise is the imaging pipeline, where electronic signals are converted into the final image (see Section 2.3). But this source of noise has only a limited influence on the use case presented, as other sources of noise are more meaningful, and the noise of the imaging pipeline is reduced by applying blur to images.

With the use case presented in this dissertation, uneven and varying conditions and ambient light have their influence on the colour information. The effect of uneven ambient illumination is mitigated by the limited spatial extent of the area from which colour information is acquired, thereby minimising the influence of lighting non-uniformities on the accuracy of colour measurement. However, dynamic or fluctuating conditions - such as changes in lighting, background, full shadows and paper colour - represent a more substantial challenge, as they can compromise the stability and reproducibility of the captured data (Figure 4) and in real life, dust, dirt, and tampered colours make the challenge even more complex. Ambient light conditions cannot be strictly controlled, as consumers use mobile applications in many different environments. To tackle this common issue, image auto-levelling can be used to make the solution presented more generally applicable. In image auto-levelling, the brightness and contrast of a digital image are automatically adjusted to enhance its overall appearance. The goal is to enhance the visual quality by distributing the intensity values (brightness) more evenly across the available range, often from the darkest black to the brightest white. Image auto-levelling adjusts the pixel values so that the darkest and brightest regions are stretched to fit the full dynamic range of the image. The references for darkest and brightest regions are known as source data and contain regions with no-colour and those which are totally black (Limare, Lisani, Morel, Petro, & Sbert, 2011). A more advanced solution would be to add white-balance or similar corrections to the image, which could be helpful when adapting the solution to different ambient light temperature conditions. But as matters of imaging scene in the use case, close range to object, and a low number of colours present, such algorithms are not suitable to be used in the presented use case.

This dissertation offers a novel contribution to the fields of computer vision and machine learning by providing a comparative analysis of various methods for subtle colour difference recognition. Additionally, the dissertation compiles a collection of approaches suitable for diverse use cases in the recognition of colour differences. The dissertation also supports the development of functional inks and their use cases

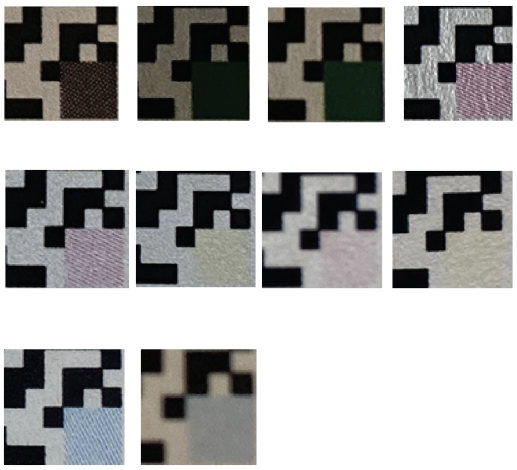


**Figure 4.** Examples of the different environments, paper and printer noise.

by using new methods that can accurately recognise colour or its change. Indirectly, the dissertation also provides new information for the development of QR codes if colours are embedded in the QR code for a specific reason. With this information, more use case-specific smart tags can be developed with functional inks, and the information that is read from the smart tags will be more trustworthy and accurate.

## 1.1 Related past research

This dissertation is structured in such a way that the colour difference algorithm CIEDE2000, unsupervised and supervised learning are explored. The methods are then compared, and the relevant past research related to each of the methods is described in this section. In previous research, many approaches for colour recognition have been used. Notably, some of the earlier research has been conducted long ago in the 1970s and 1980s, before artificial intelligence emerged during the 2010s.

The mathematical methods in colour difference calculation are widely used and developed. The most recent version of industrial standard is the so called CIEDE2000 (CIE00) by CIE (International Commission on Illumination), which calculates the difference between two colours, building on the previous models CIE76 and CIE94. The formula is particularly designed to address perceived colour differences across the colour space. CIEDE2000 has been used in many studies where the colour difference is matched against human perception or in general colour comparison. Examples of such research are rice colour recognition by Nguyen, Vo, and Ha (2022) and mortar colour difference recognition by López, Guzmán, and Di Sarli (2016). Previous research by other researchers (e.g. Ghinea et al. (2011, 2010);

Mangine, Jakes, and Noel (2005); Pecho, Ghinea, Alessandretti, Pérez, and Della Bona (2016)) shows that mathematical methods are useful when the differences in colour are large or when the environment can be controlled. Using such an environment, the authors showed that mathematical methods can be used in small colour difference recognition. The work done in the context of dental ceramics (e.g. Ghinea et al. (2011, 2010); Pecho, Ghinea, Alessandretti, Pérez, and Della Bona (2016)) shows that the CIEDE2000 algorithm can be used when there are small differences in the colours. In all of these previous studies, mathematical algorithms have been used in a very controlled environment, and they have focused on capturing colour information from surfaces which are quite solid in colour. In this dissertation, mathematical methods are used to calculate the difference of two colours (Article I).

Unsupervised learning (Article II) has been mostly used for colour segmentation when it comes to colour recognition. Colour segmentation aims to identify and isolate distinct regions within an image based on their colour attributes. This process typically involves algorithms that analyse the pixel values of an image, grouping similar colours to clusters to create meaningful segments that can be further processed or analysed. Such clustering can be used for colour recognition if pixels with the same colour belong to one cluster. In the past, unsupervised learning, mainly clustering, has been used, for example, in agriculture (e.g. Abdalla, Cen, El-manawy, and He (2019); Al-Shakarji, Kassim, and Palaniappan (2017); Di Gennaro, Toscano, Cinat, Berton, and Matese (2019); Fuentes-Peñailillo, Ortega-Farias, Rivera, Bardeen, and Moreno (2018)) and healthcare (e.g. Khan et al. (2021); Rundo et al. (2020)). The mentioned past research articles show that unsupervised clustering has its use cases in colour image segmentation. But past research has also shown that unsupervised learning has its challenges when colour differences between segments are small or when segments do not have clear edges. The big difference from previously mentioned mathematical methods is that unsupervised methods have been used in environments which are not so strictly controlled. However, even though these past studies were working with colours, they do not directly focus on the recognition of colour differences in printed sources.

Past studies on supervised learning are loosely related to the approach presented in this study (Articles III & IV). However, they have explored the use of colours in artificial neural networks in different contexts or as features of broader object recognition tasks (Anandhakrishnan & Jaisakthi, 2022; Arsenovic, Karanovic, Sladojevic, Anderla, & Stefanovic, 2019; Lai & Westland, 2020; J. Wu et al., 2019). These research articles are examples of the use of supervised learning in the context of colour recognition. But although they only use colour as one feature and state that using many features is advantageous when it comes to colour recognition, they give directions for future research. Many other past studies have demonstrated the power of convolutional neural networks (CNNs) (e.g. Apriyanti, Spreeuwers, Lucas, and Veldhuis (2021); Atha and Jahanshahi (2018); Boulent, Foucher, Théau, and St-

Charles (2019); Büyükarıkan and Ülker (2022); Engilberge, Collins, and Süsstrunk (2017); Przybyło and Jabłoński (2019); Q. Zhang et al. (2018)) in use cases that relate but are different, as they do not cover subtle colour difference in printed sources. Of the mentioned research, Atha and Jahanshahi (2018) found out that CNN's, especially ZFNet, can classify items based on small differences. The work by Büyükarıkan and Ülker (2022) also proved that various CNN's can estimate the illumination of images obtained in varying light colours, and this study is quite well related to the approach presented in this dissertation. However Büyükarıkan and Ülker had a special setup for image capturing and a larger surface area for colour. Colour identification does not always need a deep network with a large amount of parameters, as Zhang et al. showed, as they were able to classify eight different colours with their custom lightweight CNN. Apriyanti et al. (2021) were able to use different CNN's to classify flower colours into five categories. As can be seen, past research has shown that CNNs are well suited for colour recognition or similar tasks. Research has further shown that CNN's ability to automatically learn hierarchical features (such as intricate patterns and variations) from images is useful for the recognition of colour differences.

As a summary of the past research, there are many studies that relate to this dissertation as they have looked into colour or colour difference recognition. What makes this dissertation different is the use of printed colours as a dataset. Such data sets, using printed colours and collected in a non-controlled environment, have not been collected in the past. Another new contribution is the recognition of subtle colour differences in printed sources, in which mathematical, unsupervised, or supervised methods were used, and such comparisons have not been performed in the past.

## 1.2 Objectives and research methods

Previous scientific research has focused on use cases where the coloured sample areas occupy a relatively large portion of the object (e.g., cars or fruits with a dominant surface colour), or scenarios with clear chromatic deviations. Many studies have relied on high-quality cameras designed for surveillance or photography, such as SLR cameras. In prior neural network research, colour has typically been treated as only one of many item features.

The main research problem addressed in this dissertation is the reliable detection and classification of subtle colour differences with machine learning and colour difference metrics. The objective is to investigate methods for recognising colours and their intensity from printed sources, particularly in cases where the colour sample areas are small and no accurate colour reference is available. The small colour area is not typically influenced by shadows or uneven ambient light, so research focusses purely on colour difference recognition.

The study aims to evaluate and compare different technical approaches including a non-learning-based colour difference algorithm, an unsupervised clustering methods, and supervised neural network models for their applicability in real-world use case, where images are captured with mobile devices. The research focusses on adapting and extending existing methodologies to a context where colours are embedded within a printed QR code marker, simulating an colour changing indicator.

## 1.3 Contributions

### 1.3.1 Article I

This article demonstrates that embedding functional ink within traditional markers does not negatively impact the decoding performance of smart tags. The article explores the effectiveness of the CIEDE2000 colour-difference algorithm in detecting indicator states, with a specific focus on its performance in different parameter combinations. CIEDE2000 parameters account for differences in lightness (L *), chroma (C *), and hue (H *), and these parameters incorporate corrections for chroma compression and hue interactions to align with human visual perception. Adjustable weighting factors of the parameters allow the algorithm to be tailored for specific applications, ensuring a more accurate assessment of colour. Among the parameter weights tested, the combination CIEDE2000(2.76, 1.58, 1) demonstrated the best performance, particularly in scenarios involving low-intensity functional ink. The article highlights the need for future studies to address absolute colour-value detection and improve colour recognition accuracy, especially for low-intensity colours in functional inks.

### 1.3.2 Article II

Article II evaluates the effectiveness of prevalent unsupervised learning techniques in detecting and differentiating printed colours on paper, with a specific focus on CMYK ink levels. This shows that unsupervised clustering methods can reliably identify colours within QR codes when the CMYK saturation difference is 20% or higher in at least one CMYK channel. The article highlights the performance of K-means in recognising colours even when CMYK saturation is equal to 10%, although its accuracy is notably reduced for the yellow and magenta channels. The results suggest that a minimum saturation of 20% in one CMYK channel is essential for reliable colour detection using unsupervised learning techniques. Using only a 10% difference is possible, but this might lead to incorrect predictions. The article highlights the need for further research or alternative unsupervised methods to

handle low ink densities below 5%.

### 1.3.3 Article III

Article III applies standard CNN architectures to identify printed colours from mobile phone captured images, where images are pre-processed and stored in datasets with varying CMYK colour intensities. Although most CNN models performed well with colour differences of 10% or more. The best models were DenseNet (77% accuracy) and ResNet (95% accuracy) when very subtle (under 10%) colour differences were used. ResNet's residual connections and skip connections contributed to its strong performance, although it is prone to overfitting. This study sets a baseline for CNN colour classification, but more research is required on fine-tuning, optimisation, and preprocessing methods to enhance performance.

### 1.3.4 Article IV

Article IV introduces a customised ResNet-34 architecture designed for the accurate recognition of subtle colour differences when the colour difference is 10% or less. By modifying the standard ResNet-34 based on previous research, the model achieved an accuracy of 98% in colour classification, validated through a five-fold cross-validation. The colour dataset undergoes colour correction, image pre-processing, and data augmentation prior to training. The proposed model highlights key modifications, such as adding max pooling after residual block operations or replacing the final average pooling layer with max pooling. These adjustments, combined with gradient centralisation, improve the model's accuracy in detecting variations in colour intensity, demonstrating the effectiveness of custom ResNet architectures for specialised image recognition tasks.

## 1.4 Structure of the dissertation

This dissertation is structured as follows, Sections 2 and 3 cover the various technologies to an extent necessary for the rest of the dissertation. Section 4 covers the datasets and methods used in Articles I-IV. Section 5 discusses the results of each article individually and as a whole. Section 6 presents a discussion of the research results, and Section 7 summarises the dissertation as conclusions.

# 2 DIGITAL IMAGES AND COLOURS

The colours we see are electromagnetic waves of different wavelengths (Figure 5) and visible radiation. Depending on the wavelength of the wave, it can be observed in different colours. The human eye recognises colours based on light that is reflected from an object or sent by a source into photoreceptor cells by light sources. The electric signals from the cones of photoreceptor cells are then sent to the brain to process and form vision. The spectrum that the human eye can recognise starts with violet at wavelengths below 400 nm. This lower limit of the spectrum cannot be exactly defined, but it starts around 380-400 nm. Human vision is most reactive at approximately 555 nm (yellowish green). And then starts to fade away, finally ending at far reds, somewhere around 700-780 nm. The human eye has three types of colour photoreceptor cells, which react independently to green, blue, and red light. Each colour has a specific wavelength range; however, the exact wavelength range for each colour cannot be strictly defined. (Tooms, 2015)

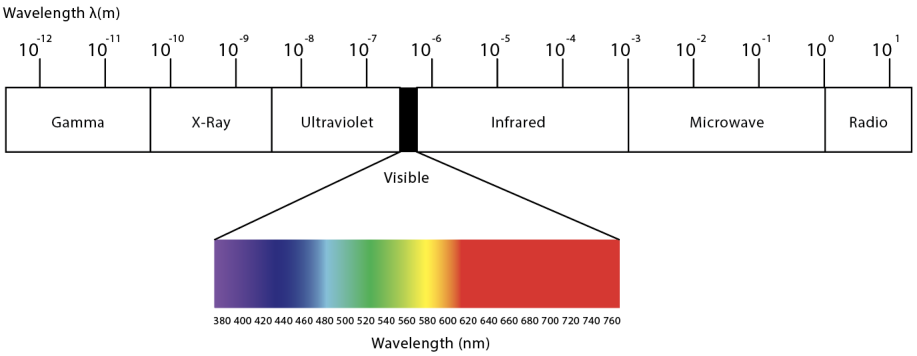



**Figure 5.** Electromagnetic wavelengths (reproduced from Tooms (2015)).

Violet has the shortest wavelength in the visible spectrum (around 380–450 nm). These high-energy wavelengths lie at the edge of what the human eye can perceive. The blue wavelengths are about 450–495 nm, and these colours are often perceived as cool. Green spans 495–570 nm. Green wavelengths are in the middle of the visible spectrum; this range is where human vision is most sensitive. The yellow wavelengths are between 570 and 590 nm. Orange ranges from 590 to 620 nm, and the orange wavelengths are longer and are often perceived as warm. Red has the longest wavelengths visible to the human eye, 620–700 nm, at the lower-energy end of the visible spectrum. (Tooms, 2015)

The digital camera works in principle in the same way as the human eye. It captures electromagnetic waves as analogue signals and converts them into electronic signals. The formation of digital images can be considered to occur in three phases. Things that happen before the actual sensor, what happens in the sensor, and the final processing of the sensor data. The purpose of the camera system as a whole

is to transform the visible light reflected from the object(s) into a digital file. All cameras are different depending on how much data storage and component space they use; however, in principle they still work in the same way and produce digital images. For example, high-quality single-lens reflex (SLR) camera's and cell phone cameras have a very different amount of physical space in their use. But in general, all cameras are still built from the same general components / modules. (Hirsch, 2022; Nakamura, 2017)

Digital cameras have developed during the last few decades into systems that can capture very fine-detailed information and are equipped with capacity to capture images which have $10^6$ ... $10^8$ pixels and around $10^2$ ... $10^3$ intensity levels present in several colour channels (Pak, Reichel, & Burke, 2022).

## 2.1 Pre-sensor

Before electronic waves arrive at the actual camera sensor and the final image is formed, some phases affect how the final image looks and how colours are represented. The following subsections describe the main parts of the image-forming process.

### 2.1.1 Lens

As Robert Hirsch explains in his book "Light and Lens: Thinking about Photography in the Digital Age" (Hirsch, 2022), the main purpose of the lens is to collect light rays coming from a subject in front of the camera and project them as images onto a sensor.

The lens allows the camera to capture light in a very short time to form the final image. The lens plays a crucial role in determining the field of view and its depth (Hirsch, 2022). In mobile devices or other devices with limited physical space available, camera lenses are usually stacked in a holder tube (Figure 6). All lenses serve a specific purpose in forming the final image. The holder tube can range from a few millimetres to a tenth of a millimetre. The small space of a mobile phone camera does not allow as fine-detailed control of parameters related to camera system lenses used in larger cameras (Skandarajah, Reber, Switz, & Fletcher, 2014).

The lenses used on mobile phones have different purposes. Normal lenses function without reduction or magnification to create two-dimensional images. These lenses attempt to replicate human vision as closely as possible, including the field of view. Wide-angle lenses (with short focal length) extend the field of view to wider scenes,

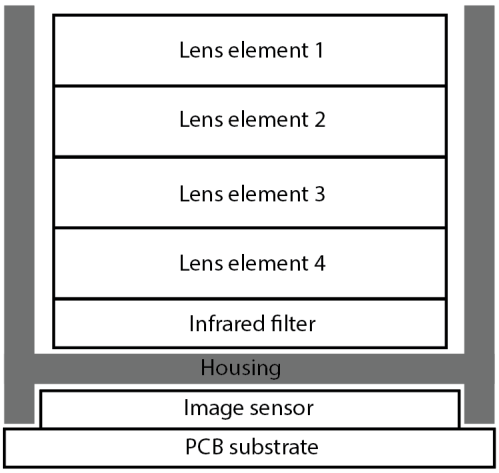



**Figure 6.** A traditional stack of lens elements in the holder tube (reproduced from: Bloss (2009)).

providing a greater depth of field. Telephoto lenses are the opposite of wide-angle lenses and have a smaller depth of field, and allow taking photographs from greater distances. The fourth lens type that can be found in mobile phones is the macro-lens, which allows sharp and clear close-up photography.(Hirsch, 2022)

Mobile phones have very limited space with regard to the overall camera structure. This also means that mobile phones cannot be equipped with highly moving lenses, as featured in traditional cameras. Moving lenses allow cameras to adapt to different distances, environments, and user needs. To overcome this restriction, manufacturers of cell phone cameras have added multiple cameras with different lenses for different purposes to their product range. One of the mobile phones with multiple cameras is the Nokia Pureview, which has five 12-megapixel cameras (*Nokia N9 product page*, n.d.). In today's popular mobile phones, users can easily find two or three cameras with different lenses for photography.

When it comes to using mobile devices for special imaging, special equipment for hyperspectral, thermal, or ultra-violet imaging has been developed.

### 2.1.2 Filters

Before visible light reaches the sensor within the camera, it passes through camera-specific filters. These are called either colour filter array (CFA) or colour filter mosaic (CFM). Camera sensors (see Section 2.2) feature a grid of millions of individual light-sensing elements (pixels), and are monochromatic sensors that react only to a specific wavelength range. The neighbouring pixels record different wavelengths, forming the basis for subsequent colour reconstruction. CFA or CFM is integrated

into each pixel sensor. Filters are responsible for selectively transmitting specific wavelengths of light, enabling each pixel to discern a particular colour channel of red, green, or blue, if a common Bayer pattern is used. (Nakamura, 2017)

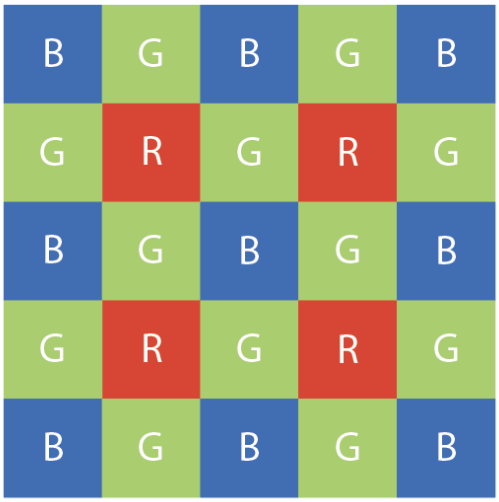



**Figure 7.** 5 × 5 Bayer CFA pattern (reproduced from: Bayer (U.S. Patent 3971065A, Jul. 1976)).

The CFA (or CFM) can be constructed in various ways, leading to different approaches to the final image reconstruction. One of the most common CFA designs is the Bayer pattern (Figure 7), which consists of a 2x2 grid where each pixel is assigned a red, green or blue filter. In the Bayer pattern, the green filters dominate 50% of the total. This pattern leverages the human eye's increased sensitivity to green light. However other designs than Bayer patterns are also used. (Bayer, U.S. Patent 3971065A, Jul. 1976)

Before CFA (or CFM), there are usually at least infrared (IR) filters, which prevent harmful wavelengths from entering the image sensor and negatively affecting image quality (Nakamura, 2017; Wilkes et al., 2016).

## 2.2 Sensor

A digital sensor array converts the light it receives into digital information (pixels). There are different versions of the sensor array (EMCCD, SCMOS, CMOS, and CCD), each of them working in a slightly different way and having their own characteristics. The most popular are CMOS and CCD, which can be found in cell phone cameras.

Charge-coupled devices sensors (CCDs) comprise an array of light-sensitive pho-

todiodes integrated into a semiconductor substrate, typically silicon. Upon photon absorption, each photodiode generates electron-hole pairs, initiating the accumulation of photogenerated charge within the depletion region of the semiconductor. The CCD orchestrates charge transfer through the lattice-like structure of the device. This is facilitated by a sequence of potential wells formed by electrodes placed on the semiconductor surface. At an operational level, the CCD sensor has an arrangement of clocked voltages to shuttle charge packets across the pixel array. This process is known as charge transfer, and it ensures that spatial information is preserved while also mitigating noise and distortion. (Barbe, 1975; Nakamura, 2017)

CCD sensors have some advantages over CMOS, they are less prone to noise, provide higher quality, and are more sensitive to light (Cabello et al., 2007; Magnan, 2003).

Complementary Metal-Oxide-Semiconductor (CMOS) sensors have photodetectors and active pixel readout circuitry embedded within a monolithic silicon substrate. CMOS sensors harness the principles of metal-oxide-semiconductor field-effect transistors (MOSFETs) to facilitate pixel-level signal amplification and readout. Each pixel on a CMOS sensor comprises a photodiode, responsible for photon-to-charge conversion, and associated transistor circuitry, including amplifiers and analog-to-digital converters (ADCs), enabling on-chip signal processing and digitization. CMOS sensors can be passive or active; In passive pixel arrays, charge amplifiers are located at the bottom of each column of pixels, and each pixel has only one transistor. Active pixel arrays implement an amplifier on every pixel. (Magnan, 2003; Nakamura, 2017)

The main advantages of CMOS sensors are low power consumption, low cost, and high speed. The disadvantages are low light sensitivity, low charge capacity, pixel uniformity, and noise. (Bigas, Cabruja, Forest, & Salvi, 2006; Cabello et al., 2007; Magnan, 2003)

Both types of sensors are highly reliable and suitable for use in mobile devices. In both sensors, noise is still generated, which affects image quality when closely observed (Naveed, Ehsan, McDonald-Maier, & Ur Rehman, 2019). In the context of the dissertation, both can capture 8-bit or higher colour depths. However, CMOS has the advantage of having all of the electronic components in one circuit, making the integration of CMOS sensors more feasible in smaller spaces. CMOS-based cameras also have lower power dissipation, which is an advantage in mobile devices. (Litwiller, 2001)

## 2.3 Post-processing

The final phase to produce a digital image is digital signal processing (Figure 8), which occurs in the image processor. This part is also called post-processing or the imaging pipeline. The purpose of the pipeline is to create a digital image from the raw data that the sensor produces. The aim of signal processing is to make the final image appear as natural as possible. In consumer use, post-processing might optimise images for viewing purposes, and the results can be somehow different from the actual imaging scene. Depending on the camera module manufacturer, the purpose of the camera, and the phases that occur before the digital imaging pipeline, various algorithms are used when the final image is produced. In addition, the order of operations used varies. An example of a possible imaging pipeline is presented in Figure 8. (Nakamura, 2017; Ramanath, Snyder, Yoo, & Drew, 2005)

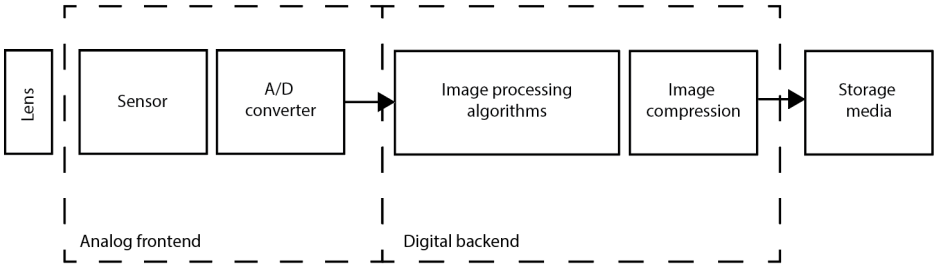



**Figure 8.** Example of an imaging pipeline (reproduced from: Nakamura (2017)).

The black-level adjustment ensures that the darkest parts of the image (where the sensor does not capture light) are set to zero. Even when the sensor does not capture any light (total darkness), some factors like electronic offset and thermal noise can produce non-zero values for the sensing element. This is called dark-current noise. Incorrectly addressing the dark current noise results in a whitening of the shadows and a reduced overall image contrast. (Schöberl, Senel, Fößel, Bloss, & Kaup, 2009; Zhou & Glotzbach, 2007)

White balance (WB) is performed by an algorithm to remove the colour cast caused by scene illumination. The objective of white balance is to make white objects appear as white in the image. The colour of the cast depends on the used light source; for example, fluorescent and incandescent lights produce differently coloured casts. (Ramanath et al., 2005; Zhou & Glotzbach, 2007) The automatic white balancing (default option in smart phones) does its best to estimate true white in the imaging scene and then adjusts the colour balance accordingly (Zhou & Glotzbach, 2007).

Demosaicing and colour interpolation are used to reconstruct the three-dimensional (R,G,B) colour image. The demosaicing receives information from the Bayer filter of the camera. The Bayer filter (or any other filter) used has an interleaved pattern of different colour filters, which results in a sensor output that also has an inter-

leaved pattern. In practice, this means that not all of the colour pixels of the final image receive a direct signal relating to them. Demosaicing uses algorithms like bilinear interpolation, adaptive colour-plane interpolation, and frequency domain to form the final image, where each pixel has a colour value. (Gunturk, Glotzbach, Altunbasak, Schafer, & Mersereau, 2005; Ramanath et al., 2005)

There are many sources that contribute to the noise during the imaging process. The black-level adjustment previously mentioned is intended to minimise so-called fixed pattern noise (FPN). Other noise that appears in the raw sensor data is called photo-response non-uniformity noise (PRNU). Some PRNU occurs because colour pixels have different sensitivities to light, which is initially caused by the non-homogeneity of silicon wafers and imperfections during the sensor manufacturing process. Environmental and scene-related variables also contribute to PRNU, and these are for example the light refraction of dust particles and settings that relate to camera behaviour (such as zoom use). Some of the noise in the camera sensor is so permanent and unique that it has been used to identify individual cameras (Lukas, Fridrich, & Goljan, 2006). An imaging pipeline typically contains an algorithm that reduces noise in the early phases of the imaging process.

Colour correction can be done from multiple different phases and with various algorithms. The final objective of colour correction is to adapt the camera to current scene illumination and make colours as natural as possible. This part is also called the colour constancy of the imaging pipeline. (Nakamura, 2017; Zhou & Glotzbach, 2007) In consumer use, imaging pipeline optimisations might aim to make the image both engaging and visually comfortable.

In addition to colour correction, which attempts to make colours as naturally as possible, other processing can also be done for images. Each image captured by a digital camera contains some degree of blur, at least in some parts of the image. To make the image sharper, especially with regard to the edges of objects, edge detection is used to sharpen edges. Edges are an important part of an image because they are the locations where objects start or end. Detecting the edges of objects is also a way to recognise objects (Zhou & Glotzbach, 2007). Edge detection first looks for edges that should be sharpened and then applies transformation to make edges appear more natural. Enhancing edges too much, on the other hand, makes images appear unnatural. (Tang, Astola, & Neuvo, 1994)

Other algorithms and methods are also used, most of them optimised for a certain camera model or even individual camera and conditions (Jiang, Tian, Farrell, & Wandell, 2017; Ramanath et al., 2005). Basically, it can be considered that two cameras that take images exactly at the same time and take images from the same scene still produce slightly different images. So, even if you capture a scene twice in a very short time frame, both images may vary slightly although this difference is hardly noticeable.

The final phase of the imaging pipeline is the storage of the image. In advanced cameras, image data can be stored without compression, as a so-called RAW image. However, in mobile devices and consumer use, this is rarely possible or feasible. If an image is stored without compression, it consumes more storage space and is slower to process and share. Typically, images are stored in JPEG format on mobile devices (Nakamura, 2017). The compression of images comes with a cost, as some fine details, as well as colour and tonal detail, can be lost during the compression. The compression also forces the image to 8 bits per channel, even if a deeper colour depth was used earlier in the imaging pipeline. JPEG compression is a lossy image compression method that reduces file size by discarding some of the image data. The algorithm for compressing the JPEG image uses 8x8 pixel blocks, and a discrete cosine transform (DCT) is applied to each block to represent the pixel data as frequency components (Rao & Yip, 2018). The higher frequency components, which typically represent less visually significant details, are quantised and often discarded to achieve compression. Finally, the remaining data are encoded using entropy-coding techniques such as Huffman coding, resulting in a smaller file size. (Pennebaker & Mitchell, 1992)

## 2.4 Presentation of colours

This subsection provides an overview of the most commonly used colour spaces and those related to this dissertation.

The presentation of colours depends on the medium used. Roughly colours are presented either in digital or non-digital format. Digital mediums like displays, cameras, etc. use different colour spaces (mathematical models) to present colours (X. Wang & Zhang, 2010). The colour space defines the colours a specific medium is able to show. The most commonly used colour spaces in digital imaging are various RGB colour spaces. In these colour spaces, red, green, and blue are represented by individual electronic signals. In a non-electronic medium like printing, colours are typically represented in the CMYK colour space. In CMYK, cyan, magenta, yellow, and black are mixed to achieve the desired colour. When working with different media, a conversion between colour spaces is required. In addition, there are many colour spaces that are used for different purposes; some of them are used for special devices, and some of them are made for mathematical purposes (Fan, Li, Guo, Xie, & Zhang, 2021).

Different mediums like displays or printers can display different ranges of colours and gamut. In the optimal world, all media would have the same colour space, but medias are somehow different and even two identical displays have a slightly different colour range that they can interpret. Another major difference comes from the way mediums work, where displays emit light that results in a colour, and paper

reflects wavelengths representing different colours. A colour space specifies what colours the current medium can interpret, but it also looks at standardising colours between mediums (Berns, 2019).

Colour spaces can be broadly categorised into device-dependent and device-independent colour spaces. Both of these space types serve different purposes. If colours are presented in a device-dependent colour space (most RGB, CMYK colour spaces) colour space and then another device is used to present them, the colours may vary. In addition, the colours captured by different imaging devices vary, as the devices cannot capture the same range of colours. The device-dependent colour space presents the colour range that a specific device can display. Device-dependent colour spaces are primarily used in tasks where the output is intended for a particular device type, such as graphic design for screens or preparing files for specific printer models. (Anderson, Motta, Chandrasekar, & Stokes, 1996; Nakamura, 2017; X. Wang & Zhang, 2010)

Device-independent colour spaces are designed to represent colours consistently, regardless of the device used to capture, display, or print them (Green & MacDonald, 2011). Examples of such colour spaces are CIE L*a*b* (LAB) and CIE XYZ. The LAB colour space represents colours based on human vision, and CIE XYZ serves as the foundation for many other colour spaces (a reference space) (Nixon, Outlaw, & Leung, 2020). Device-independent colour spaces are used in scenarios where accurate colour matching is essential, such as in professional photography, printing, and colour quality control (Ford & Roberts, 1998; Green & MacDonald, 2011).

Device-dependent colour spaces are essential for the specific requirements of individual devices, whereas device-independent colour spaces provide accurate and consistent colour representation across different devices and media.

### 2.4.1 CMYK

When colours are printed on paper or painted on the walls of a house, a subtractive colour space is used. Subtractive colour space works by subtracting (absorbing) certain wavelengths of light and reflecting other wavelengths. If the object (paper or label) has a white background colour, the CMYK colour printed on it reduces specific wavelengths of light that would otherwise be reflected. In the printing industry, the most common subtractive colour space is CMYK, which is used in large industrial printers and home and office printers. In the CMYK colour scheme (Figure 9), cyan (C), magenta (M), yellow (Y), and black (K) all have their own source of colour. All C, M and Y colours that are used are complements of additive primary colours (Figure 10). Mixing CMYK colours in their full intensity would produce a black colour, but for technical and cost reasons, black (K) is used as its own colour

in printing. (Anilkumar, KK and Manoj, VJ and Sagi, TM, 2018)

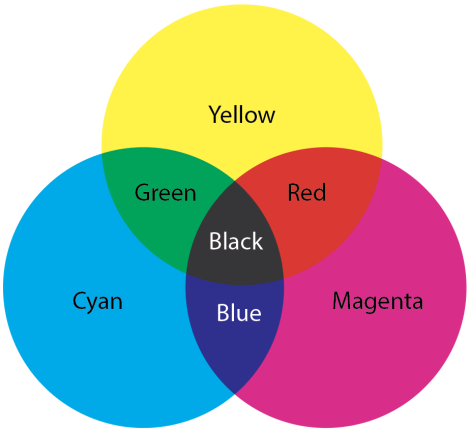



**Figure 9.** CMYK colour space.

The CMYK colour can be defined in many ways. For example, yellow can be defined as CMYK(0.0, 0.0, 1.0, 0.0), and in this presentation, each colour can vary in the range 0.0 ... 1.0. Here, 0.0 is no intensity and 1.0 is full intensity. Another way is to use 0%C, 0%M, 100%Y, and 0%K, in which the intensity of each colour varies between 0% and 100%. The percentage sign can sometimes be left out which leads to a presentation CMYK = 0, 0.0, 100, 0. Some features of CMYK colour and its printing process have been defined in the ISO 12647-2:2013 (2013) and ISO 2846-1:2017 (2017) standards.

### 2.4.2 RGB

The red, green and blue (RGB) colour space (Figure 10) is commonly used in digital devices. This colour space is additive, and the colours are created by combining red (R), green (G), and blue (B) light at various intensities (Palus, 1998). The colour space is built around each primary colour, and when primary colours are combined, different colours can be created. When all three colours are combined at full intensity, the result is white colour. If all three colours are set to zero (no signal), the result is black. (Nakamura, 2017)

In the RGB colour space, each channel (R,G,B) is represented by a numerical value. These values typically range from 0 to 255 in 8-bit systems. The value range can also be 0.0 ... 1.0, where 0.0 is no signal (Ford & Roberts, 1998). The RGB colour space can allow for more than 16 million possible colour combinations. RGB colour is commonly described by the text RGB: 255,0,0, which represents the full red colour. (Fan et al., 2021)

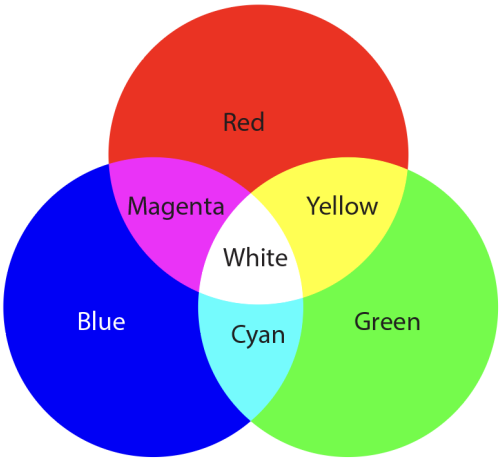



**Figure 10.** RGB colour space.

The ISO standard that closely relates to digital cameras is ISO 17321-1:2012 ”Graphic technology and photography — Color characterization of digital still cameras (DSCs).” ISO 17321 works as a quality assurance tool for digital camera manufacturers and helps to achieve consistent colours between devices, as well as providing a way to benchmark different devices. (SO 17321-1:2012, 2012)

### 2.4.3 LAB/CIELab

The LAB colour space (Figure 11) is a perceptually uniform colour space. The LAB colour space was developed by the International Commission on Illumination (CIE). The purpose of the LAB colour space is to represent all perceivable colours perceived by the human eye. The LAB colour space is designed to be device-independent, ensuring consistent colour representation across different devices. The LAB can accurately describe colours and their relationships. The perceptual uniformity of LAB makes it valuable for applications that require precise colour measurements and comparisons. (Berns, 2019)

LAB consists of three axes: L*, a*, and b*. L* defines lightness, ranging from 0.0 (black) to 100.0 (white). a* defines the green-red component, which ranges from -128 to +127. Negative a* values indicate a greenish hue, while positive values indicate a reddish hue. b* defines the blue-yellow component, ranging from -128 to +127. Negative b* values indicate a blueish hue, whereas positive values indicate a yellowish hue. In both a* and b*, zero means that there is no tint. (Fan et al., 2021) LAB colour space has a special relation to the human vision system because LAB is perceptually uniform, and the change in its component values equals a change in visual perception (Berns, 2019).

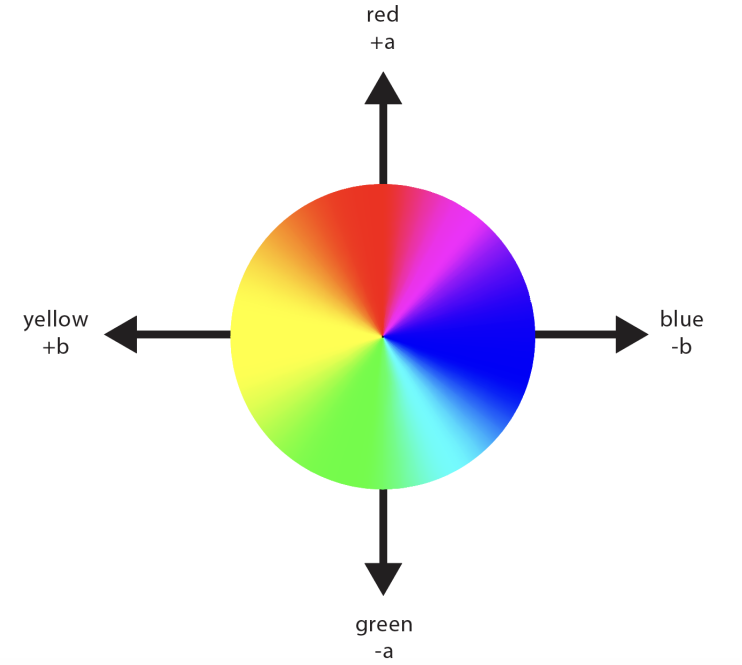



**Figure 11.** LAB colour space (reproduced from: Belasco et al. (2020)).

### 2.4.4 Colour depth

The colour depth / bit depth defines the number of bits used to represent the colour of a single pixel in the digital image. The colour depth is one single feature that is important when the accuracy of colour recognition is considered. The more depth an image has, the more different colours can be captured. The imaging pipeline (starting from the sensor) must be able to capture enough colours, and the final display device must be able to interpret them.

The depth of the colour is determined by the number of bits assigned to each pixel, and more bits per pixel allows for a greater variety of colours. If an image has only 1 bit per pixel, it can present two different colours, typically black and white. In RGB (see 2.4.2) each channel has an individual depth of colour. Most commonly in consumer devices, colour depth is 8-bits per pixel, which allows for 256 different intensities of colours to be captured on each channel. In total, 8 bits per channel on three channels (R, G, B) gives a total of 24-bits, and means 16.7 million possible colours. 24-bit image is considered to be true colour. (Doolittle, Doolittle, Winkelman, & Weinberg, 1997)

48-bit and higher, deep or high dynamic range, images are used for applications requiring extreme colour accuracy and detail. These images are used in professional photo editing, medical imaging, and scientific visualisations. If an image has 16 bits per channel, a total of 281 trillion possible colours can be presented. High colour

depth enables more accurate representation of subtle colour variations, which can be critical for detecting fine details in images.

The following image (Figure 12) shows various depths of colour. In the top-most colour bar there are 2 bits (4 intensities) and the most bottom bar has 8 bits, 256 intensities on the red colour channel.


**Figure 12.** Various colour depths presented on red channel.

Later in this dissertation, subtle colour differences are experimented with. In the experiments, the colour differences are split into different intervals of 5%, 10% and 20%. These differences are related to colour depth, and if intervals are in 5% steps there can be 20 different intensities, 10% steps have 10 different intensities and 20% steps have 5 different intensities (presented in Figure 13).


**Figure 13.** Different colour steps.

### 2.4.5 Colour conversions

As seen in the previous subsections, different colour spaces are used for different purposes. When colours from different mediums and different colour spaces are processed in another colour space, a colour space conversion is needed (Z. Su, Yang,

Li, Jing, & Zhang, 2022). Converting an image from a scanner (RGB) to a printer (CMYK) may involve first transforming the image into a device-independent colour space (e.g., LAB) to ensure accurate colour mapping. In addition, when printed colour (CMYK) is captured into a digital image, it is first stored in an RGB colour space.

The common way to process colours is to either use them in the source colour space or convert colours into a device-independent reference colour space like LAB. The role of colour space is a topic that has recently been addressed in many studies (e.g. Moreira, Magalhães, Pinho, dos Santos, and Cunha (2022); Nugroho, Goratama, and Frannita (2021); Zhbanova (2020)). Each digital colour space that represents the colours that a physical device can interpret has its own colour area (Figure 14), gamut; therefore, there must be a colour space that is large enough to act as a transformation colour space. For this purpose, it is possible to use LAB colour spaces (see 2.4.3) (Berns, 2019). The role of transformation between colour spaces is illustrated in Figure 14 where the CIEXYZ (CIE1931xy) reference colour space is significantly larger than the device-depend colour space (sRGB). (Palus, 1998)

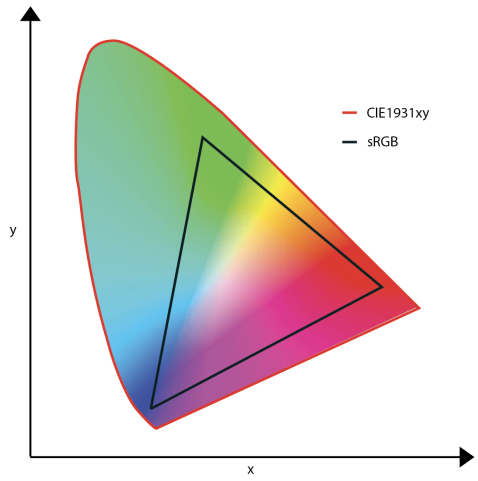



**Figure 14.** Different colour space gamuts (reproduced from: Palus (1998)).

Colour conversion between different colour spaces involves mathematical transformations that map values from the source to target colour spaces. This process usually begins with an understanding of the properties of both the source (e.g. RGB) and target (e.g. LAB) colour spaces (C.-H. Lee, Lee, Ahn, & Ha, 2001). Converting a device-dependent colour space to a device-independent colour space first requires transforming RGB values to CIE XYZ values using a linear conversion matrix, followed by conversion from XYZ to LAB using a non-linear transformation based on reference white points (Plataniotis, 2001). When transforming from a device-independent colour space to a device-dependent one, the process is reversed. The conversion process includes gamut mapping to handle colours outside the target space range (Morovic & Luo, 2001). Accurate colour conversion maintains the

consistency and fidelity of the colours across different devices.

## 2.5 Measurement of colour differences

Measurement of colour differences in digital images involves quantifying similarities or differences between two colours. The process is usually performed in a colour space like LAB because of its perceptual uniformity, ensuring that measured differences correlate well with human visual perception. (Nakamura, 2017)

The measurement of colour can be done either as an absolute difference or as a relative difference. Absolute colour difference refers to the measured colour difference between two colours in a standardised colour space, such as LAB, without considering the context or environment. Relative colour difference considers the colour difference in the context of surrounding factors, such as lighting, surface reflectance, or viewing conditions.

The mathematical formula (CIEDE2000) for calculating the ΔE00 colour difference is as follows (Luo et al., 2001):

$$\Delta E_{00} = \sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2 + \left(\frac{\Delta C'}{k_C S_C}\right)^2 + \left(\frac{\Delta H'}{k_H S_H}\right)^2 + R_T \left(\frac{\Delta C'}{k_C S_C}\right)\left(\frac{\Delta H'}{k_H S_H}\right)}$$

where the terms are defined as shown on the next page.

$$\Delta L' = L_2 - L_1$$
$$\bar{L} = \frac{L_1 + L_2}{2}$$
$$C_1 = \sqrt{a_1^2 + b_1^2}$$
$$C_2 = \sqrt{a_2^2 + b_2^2}$$
$$\bar{C} = \frac{C_1 + C_2}{2}$$
$$\Delta C' = C_2 - C_1$$
$$a_1' = a_1 + \frac{a_1}{2}\left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right)$$
$$a_2' = a_2 + \frac{a_2}{2}\left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right)$$
$$C_1' = \sqrt{a_1'^2 + b_1^2}$$
$$C_2' = \sqrt{a_2'^2 + b_2^2}$$
$$\bar{C}' = \frac{C_1' + C_2'}{2}$$
$$\Delta a' = a_2' - a_1'$$
$$\Delta b = b_2 - b_1$$
$$\Delta H' = \sqrt{\Delta a'^2 + \Delta b^2 - \Delta C'^2}$$
$$\bar{h}_1 = \text{atan2}(b_1, a_1')$$
$$\bar{h}_2 = \text{atan2}(b_2, a_2')$$
$$\bar{H}' = \frac{\bar{h}_1 + \bar{h}_2}{2}$$
$$T = 1 - 0.17\cos(\bar{H}' - 30^\circ) + 0.24\cos(2\bar{H}') + 0.32\cos(3\bar{H}' + 6^\circ) - 0.20\cos(4\bar{H}' - 63^\circ)$$
$$R_T = -2\sqrt{\frac{\bar{C}'^7}{\bar{C}'^7 + 25^7}}\sin(60^\circ \exp\left(-\left(\frac{\bar{H}' - 275^\circ}{25^\circ}\right)^2\right))$$
$$S_L = 1 + \frac{0.015(\bar{L} - 50)^2}{\sqrt{20 + (\bar{L} - 50)^2}}$$
$$S_C = 1 + 0.045\bar{C}'$$
$$S_H = 1 + 0.015\bar{C}'T$$
$$k_L, k_C, k_H = \text{parametric factors}$$

The parametric factors in the CIEDE2000 algorithm are scaling coefficients that

allow the user to adjust the weighting of the lightness, chroma, and hue components in the colour difference calculation. These factors are particularly useful for adapting the formula to specific viewing conditions, applications, or industries where the perception of colour differences may vary. Parametric factors in the equation are usually set to 1.0, but for different purposes other values can be used, and in this way the formula can give different weights to differences in chroma or hue (see del Mar Perez et al. (2011); He, Xiao, Pointer, Melgosa, and Bressler (2022); Isohanni (2022); Pereira, Carvalho, Coelho, and Côrte-Real (2019)).

The $\Delta$E00 gives information about how different two colours are. If $\Delta E00 < 0.5$, the difference is practically invisible. If $\Delta E00 < 2.0$, the colours have a slight difference, not noticeable to the human eye. The actual $\Delta$E00 threshold human eye can recognise is different for each individual (Mokrzycki & Tatol, 2011).

The CIEDE2000 algorithm has been widely used and research in different fields has demonstrated its capabilities to some extent. As examples, these studies include ”Chroma-dependence of CIEDE2000 acceptability thresholds for dentistry” by Tejada-Casado et al. (2024), ”Colorimetric Evaluation of a Reintegration via Spectral Imaging—Case Study: Nasrid Tiling Panel from the Alhambra of Granada (Spain)” by Martínez-Domingo, López-Baldomero, Tejada-Casado, Melgosa, and Collado-Montero (2024), ”New Insights into Wine Color Analysis: A Comparison of Analytical Methods to Sensory Perception for Red and White Varietal Wines” by Hensel, Scheiermann, Fahrer, and Durner (2023) and ”Color difference of yarn-dyed fabrics woven from warp and weft yarns in different colour depths” by X. Wang et al. (2024). These are only a few examples of the use of the CIEDE2000 algorithm. Some of the past research has also found threshold values for the $\Delta$E00 value, and for example in research by Xu et al. the algorithm was not applicable if $\Delta$E00 was smaller than 2.0 (B. Xu, Zhang, Kang, Wang, & Li, 2012). The use of the CIEDE2000 algorithm is justified if the colours compared are clearly different, or if the threshold between colours is large enough. The threshold value depends on the use case, and needs to be examined case by case.

The challenge of measuring colour differences comes from aspects that have been mentioned earlier in this chapter. Measurements can be made to work if the devices used are standardised and the environment is controllable. However, if the context of use has variation (e.g. different devices, environments, and so on), it may be difficult to achieve satisfactory results.

## 2.6 Colour related applications

”The desire for quantitative measurements of colour exists across many fields” (Nixon et al., 2020). The most recent research in the field of computer vision,

during the years 2022–2024 where colours have been a critical feature, has found many applications that benefit from accurate colour recognition or the recognition of colour change.

Healthcare research has adopted the usage of colour differences in different cases, for example when matching or finding the correct tooth colour, finding tumours or other diseases, and also in skin and tongue colour classification. Accurate recognition of colours in healthcare makes disease detection more automated and helps healthcare professionals to make informed decisions (see Amakdouf et al. (2021); Balaji et al. (2020); Kumar, Singh, and Sachan (2024); Maiti, Chatterjee, and Santosh (2021); Ni, Yan, and Jiang (2022)).

Another industry that uses colour recognition and research is food and agriculture. Many use cases (see Adiwijaya, Romadhon, Putra, and Kuswanto (2022); de Brito Silva and Flores (2021); Keivani, Mazloum, Sedaghatfar, and Tavakoli (2020); M.-K. Lee, Golzarian, and Kim (2021); Nalhiati, Borges, Sperança, and Pereira (2023); Shrivastava and Pradhan (2021); X. Su et al. (2023); D. Wang, Wang, Chen, Wu, and Zhang (2023)) in which colour plays an important role relate to food quality. In addition, plant and soil analyses have used colour classification. Colour recognition in food and agriculture can help in sorting items and automatically discarding non-valid items during the production process. Machine-based colour sorting is also suitable for high-speed production lines.

Accurate recognition of colours can also help make life easier for people with disabilities. Othman et al. (2020) and Cho, Jeong, Kim, and Lee (2020) investigated devices that can recognise different colours. In these two studies, colours that can be easily recognised were used. Particularly, colour recognition via smartphone can help people with disabilities in the retail, commuting, and home environments.

Recently, the development of so-called functional inks, which can react to environmental values like temperature and humidity, has led to a need to recognize colours accurately. One use case for such functional inks is presented in Article I of this dissertation (Isohanni, 2022). Originally, the usage of FMCG functional inks and their application in the food industry has been sparked by various other research projects, and for example, the EU Horizon 2020 funded project TagItSmart, developed various interesting use cases for functional inks (Gligoric et al., 2019).

# 3 MACHINE LEARNING

In this section, the most relevant machine learning techniques applicable to this dissertation are presented. Machine learning (ML) is a subset of artificial intelligence (AI), and is a crucial part of almost any technical development that humankind is pursuing. The development of applications like ChatGPT has brought artificial intelligence into our daily discussions. Although high-level applications have gained a lot of attention, many things are happening 'under the hood', and artificial intelligence is an integral part of many ongoing developments.

Machine learning focusses on the development of algorithms and statistical models that enable computers to learn, make predictions, or the decisions taken based on the provided data. In traditional programming, where explicit instructions are provided by the programmer for each task, machine learning systems identify patterns and rules directly from the data. Machine learning systems also attempt to improve their performance based on new data over time, without the need for writing new code. (Samuel, 1959, 1967)

Advances in the development of machine learning systems during the last decade have been driven largely by the increasing availability of large datasets, new research findings, and improvements in computational power. Some remarkable recent events, such as the success of AlexNet in the ImageNet competition, Google's DeepMind achieving human-level performance in certain games, the development of generative adversarial networks (GANs), the introduction of pre-trained models such as BERT, and the recent rise of Large Language Models (such as ChatGPT), to mention just a few. But also, more datasets have become available for the development of machine learning algorithms as the sharing of datasets has become more common. However, data collection has also become more feasible with the development of intelligent devices and applications, like the Internet of Things (IoT).

The primary input to machine learning algorithms is a dataset, with or without predefined labels or outcomes. The data can be numerical, categorical, text, or image, and is often high-dimensional. Individual elements of the dataset have unique features and individual measurable properties or characteristics. These features play a crucial role in determining the relationships and patterns that the algorithms will uncover. Consequently, the quality and relevance of features can significantly impact machine learning performance. (Goodfellow, Bengio, & Courville, 2016)

A dataset is typically organised in a structured format such as a table, where each row represents a unique observation or record, and each column represents a feature or variable of interest. Datasets can vary in size and complexity, ranging from small, simple collections to large, high-dimensional datasets with millions of records and features. A dataset can also be a collection of files, such as images, or other data. Examples of open-source dataset can be found, for example, from

https://www.kaggle.com/datasets.

## 3.1 Unsupervised learning

In the unsupervised learning a set of $X_1, X_2, ...X_p$ features from $n$ observations is given to the algorithm; however, there is no associated response variable $Y$ (Bishop & Nasrabadi, 2006). Unsupervised learning uncovers hidden patterns, structures, or relationships within the data (Valkenborg, Rousseau, Geubbelmans, & Burzykowski, 2023). A common example of unsupervised learning is the recommendations we see on various online shopping sites. In these sites, unsupervised learning is used to find a group of items that the items that we have in our cart match the most (James, Witten, Hastie, Tibshirani, & Taylor, 2023).

### 3.1.1 Algorithms

Different algorithms are at the core of unsupervised learning, and these algorithms work with datasets and individual item features. The nature of unsupervised learning is that the dataset does not contain correct answers (like in supervised learning). Algorithms that solve problems in an unsupervised manner can be roughly classified into clustering techniques, dimensionality reduction, and anomaly detection algorithms. Some of these algorithms are slightly overlapping; for example, clustering can also be used for anomaly detection.

Clustering methods (Figure 15) are used for exploratory data analysis to investigate the underlying structure of the data. Clustering groups objects with similar features. Govender and Sivakumar have provided a well-described definition for clustering; ”objects in a cluster are more similar than objects in different clusters.” (Govender & Sivakumar, 2020)

Jain recognised three main purposes for clustering (Jain, 2010):

- Exploring, solves the underlying structure and this way provides insights into the data, generates hypotheses, detects anomalies, and identifies salient features
- Natural classification, finds the degree of similarity among items
- Compression, organises the data and summarises it

Clustering analysis can be broadly categorised into hierarchical and non-hierarchical algorithms (Figure 16). Hierarchical clustering groups similar datapoints into clus-

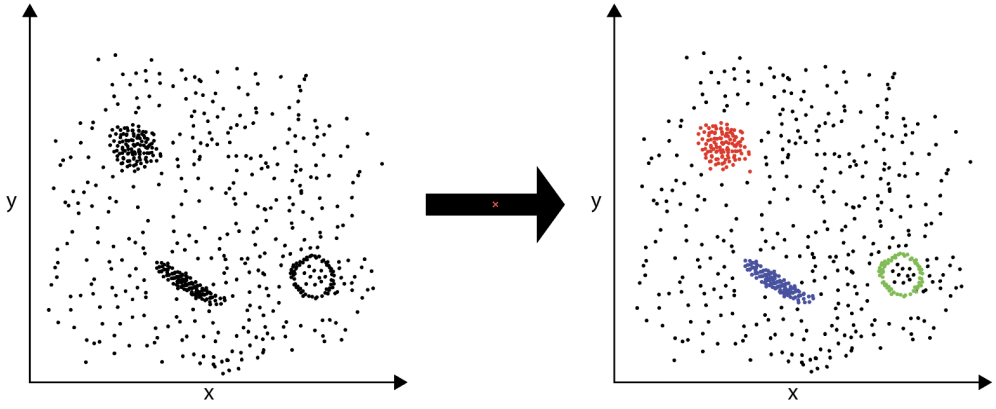



**Figure 15.** Example of clustering, left image source data, right image expected clustering (reproduced from: Jain (2010)).

ters based on distance or similarity. The hierarchy is built on a tree-like structure, where each node represents a cluster of datapoints. If the clustering process is agglomerative, the process begins with each datapoint being its own cluster. Then, the algorithm iteratively merges the closest pairs of clusters until all points are grouped into a single cluster. In a divisive approach, all points are initially placed in one large cluster, and then the algorithm recursively splits the data into smaller clusters until a predefined number of clusters are reached. Hierarchical clustering is useful because of its ability to visualise the nested relationships between datapoints, allowing for the determination of an appropriate number of clusters by cutting the tree at the desired level. (Jain, Murty, & Flynn, 1999)

Non-hierarchical clustering or partitional clustering is an unsupervised learning method that divides a dataset into distinct groups or clusters without creating a hierarchical structure. Non-hierarchical clustering directly assigns datapoints to clusters based on optimisation criteria. These clustering methods are generally faster and more scalable than hierarchical clustering. (Han et al., 2012a; Jain et al., 1999) In non-hierarchical clustering algorithms, datapoints can belong only to one cluster if hard clustering is used. In soft clustering, datapoints can belong, to some degree, to multiple clusters (Bishop & Nasrabadi, 2006). Non-hierarchical clustering can be divided into four subcategories (Figure 16): partitioning, density-based, grid-based, and others.

The partitioning methods divide the data into a predetermined number of clusters. A typical partitioning process is to first initialise as many centroids as clusters, then each datapoint is assigned to the most suitable cluster, and the centroids of the clusters are updated until convergence. (Jain & Dubes, 1988)

Density-based methods identify clusters based on their density, connectivity, and boundaries. In density-based clustering, a cluster can grow in any direction if

distance metrics and other parameters are satisfied. In practice, this means that density-based clustering can be used to adjust clusters to arbitrary shapes. Grid-based methods partition data space into a finite number of cells that form a grid structure. Partitioning of the data is performed on the basis of the algorithm parameters present. Other methods include model-based clustering and other methods that do not fit previous categories. In model-based methods, it is assumed that the data is a mixture of underlying probability distributions (mathematical models). (Han et al., 2012a)

When different clustering methods are computationally compared, hierarchical clustering has complexity $O(n^2)$, partitioning is O(n), grid-based method is O(n) and density-based method is O(n log n). Different clustering methods vary in terms of computational cost; however, these values can be used as a general rule of thumb when a suitable algorithm is selected. (Ezugwu et al., 2022)

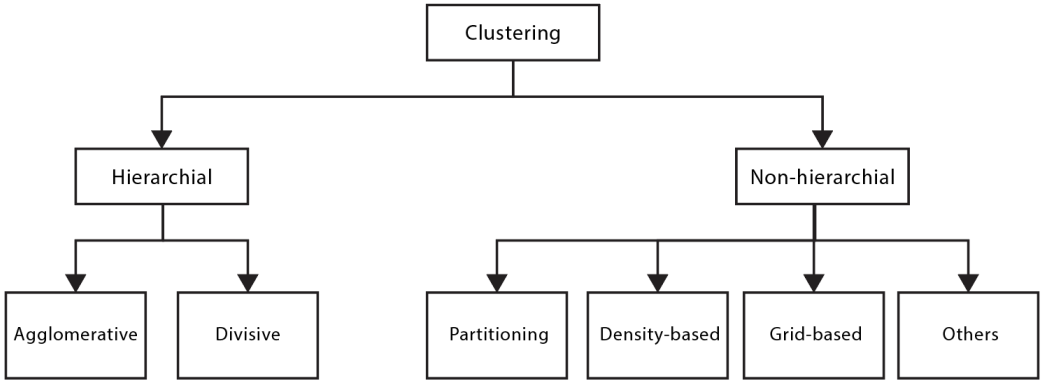



**Figure 16.** Different clustering methods (reproduced from: Han et al. (2012a)).

In clustering, two essential metrics are used (discussed in more detail in Section 3.1.2) - a distance measure to quantify similarity or dissimilarity between subjects, and an additional distance measure to quantify the difference between clusters or between a cluster and a subject (linkage). The clustering algorithm is then responsible for maximising the similarity within a cluster and the dissimilarity between clusters. (Jain & Dubes, 1988; Valkenborg et al., 2023)

Unsupervised clustering is useful when natural groupings or patterns are analysed from the data; but also if the dataset is large (Jain et al., 1999). Some challenges related to unsupervised clustering were identified in 1988 by Jain and Dubes. To address these challenges, clustering users must consider, for example, what features they want to use, whether the data have outliers, and how many clusters there are (Jain & Dubes, 1988). Depending on the domain knowledge and the answers to these questions, objectives and data, the most suitable algorithm is selected. Some of the most commonly used clustering algorithms are discussed and used in this dissertation Article II to identify which works best in the detection of subtle colour differences.

### 3.1.2 Distance Metrics

Distance metrics in unsupervised learning (non-hierarchial clustering) are mathematical measures used to quantify the similarity or dissimilarity between datapoints. Metrics are the most important value in determining the structure and formation of clusters; for example, the K-means algorithm uses distance metrics to minimise the within-cluster variance. (Jain & Dubes, 1988)

Over the years, multiple methods for calculating the distances between two or multiple points have been developed. One of the most commonly used methods is Euclidean distance, which is used to measure the straight-line distance between two points in a feature space. Euclidean distance $d(x, y)$ is the shortest possible distance between two points $(x, y)$. As a very simple distance measurement function, the Euclidean distance does not take into account the correlation between variables. The Euclidean distance is suitable for continuous data in which the magnitude of differences is meaningful, such as physical distances or measurements. (Mimmack, Mason, & Galpin, 2001) The Euclidean distance can be calculated using the following equation (Jain & Dubes, 1988):

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Here, $d$ is the distance between the datapoints $x$ and $y$. $n$ is the dimension of the data (in a three-dimensional system $n = 3$). $x_i$ and $y_i$ are the coordinates of the points in the $i^{\text{th}}$ dimension.

The Manhattan distance formula, also known as the L1 norm or the city block distance, calculates the distance between two points by summing the absolute differences in their coordinates. The result of the Manhattan distance formula is the distance if a grid-like path is made between two points. The Manhattan distance can be calculated using the formula (Jain & Dubes, 1988; Minkowski, 1910):

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Here, $d$ is the distance between the datapoint $x$ and $y$. $n$ is the dimension of the data (in a three-dimensional system $n = 3$). $x_i$ and $y_i$ are the coordinates of the points in the $i^{\text{th}}$ dimension.

The Hamming distance is used as a dissimilarity metric between categorical datapoints. The Hamming distance is described by counting the number of positions in

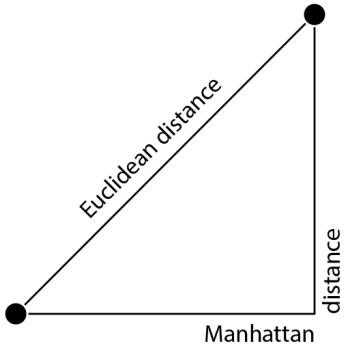



**Figure 17.** Euclidean and Manhattan distance.

which the corresponding symbols differ. The formula for the Hamming distance is (Hamming, 1950; Jain & Dubes, 1988):

$$d(x, y) = \sum_{i=1}^{n} \mathbf{1}(x_i \neq y_i)$$

Here, $d$ is the Hamming distance between the datapoint $x$ and $y$. $n$ is the dimension of the data (in a three-dimensional system $n = 3$). $\mathbf{1}(x_i \neq y_i)$ is an indicator function that equals 1 if $x_i! = y_i$ and 0 if $x_i == y_i$

The Minkowski distance is the generalisation of the Euclidean and Manhattan distances. In the Minkowski distance formula, the parameter $p$ defines the order of the norm. The value $p$ can be adjusted to form groups that best capture the underlying structure of the data. (Minkowski, 1910)

$$d(x, y) = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Here, $d$ is the Minkowski distance between the datapoint $x$ and $y$. $n$ is the dimension of the data (in a three-dimensional system $n = 3$). When $p = 1$ the Minkowski distance is equivalent to the Manhattan distance. When the value of $p = 2$ the Minkowski distance is equivalent to the Euclidean distance. As $p$ approaches infinity, the Minkowski distance approaches the Chebyshev distance $d(x, y) = max_i(|x_i - y_i|)$, which is the highest absolute value among all coordinate differences between points. (Fu & Yang, 2021)

Mahalanobis is an effective distance metric when data features have different scales and correlations; thus, it is often used in anomaly detection and classification tasks. The Mahalanobis distance is also more suitable if the clusters have an ellipsoidal

structure or if the datapoints depend on each other. The formula for Mahalanobis is (Mahalanobis, 1936):

$$d_M(x, \mu) = \sqrt{(x - \mu)^T S^{-1} (x - \mu)}$$

Here, $x$ is a vector representing the point for which the distance is calculated ($n$-dimensional vector). $\mu$ is the mean vector of the distribution, representing the centroid or mean values of each dimension in a multivariate space, and is also a vector of dimensions $n$. $S$ is the covariance matrix of the distribution, the matrix $n \times n$ that describes the variance and covariance between each pair of dimensions. $S^{-1}$ is the inverse of the covariance matrix and $(x - \mu)^T$ is the transpose of the difference vector, turning the row vector into a column vector for matrix multiplication. (Mahalanobis, 1936)

The right distance metric is crucial in clustering tasks because it defines the similarity or dissimilarity between datapoints. The distance algorithm directly affects the performance and accuracy of the clustering. (Arora, Khatter, & Tushir, 2019)

Different distance metrics capture different aspects of the data. For example, the Euclidean distance is sensitive to differences in magnitude and is suitable for continuous numerical data, whereas the Manhattan distance is more appropriate for data that follow a grid-like structure (R. Su, Guo, Wu, Jin, & Zeng, 2024; Ultsch & Lötsch, 2022). Euclidean and Manhattan distance tends to form spherical clusters, whereas the Manhattan distance may form clusters with different shapes (Hastie, Tibshirani, Friedman, & Friedman, 2009). When working with data with different scales, it is useful to normalise the data or use a metric like the Mahalanobis distance. When datapoints are in a high-dimensional space, some of the mentioned distance metrics, or other distance metrics, may lead to results where all points tend to appear equally distant from each other. In these situations, metrics such as cosine similarity (Han, Kamber, & Pei, 2012b) can be used because it measures the angle between vectors, making them more robust to the dimensionality problem.

The appropriate distance metric improves the algorithm's ability to identify meaningful patterns and relationships in the data, resulting in more accurate, interpretable, and relevant results. Choosing an inappropriate metric can result in misleading or suboptimal clustering results. (Arunachalam & Kumar, 2018; Han et al., 2012a)

### 3.1.3 Linkage

The linkage methods (Figure 18) in unsupervised clustering, particularly hierarchical clustering, determine the distance between the clusters used to form larger clusters from smaller ones or to divide larger groups into smaller ones. Linkage

methods influence the shape and composition of clusters. The used linkage method is selected on the basis of the specific requirements of the analysis and the nature of the data. Various methods to calculate linkage exist, with the following being the most popular.

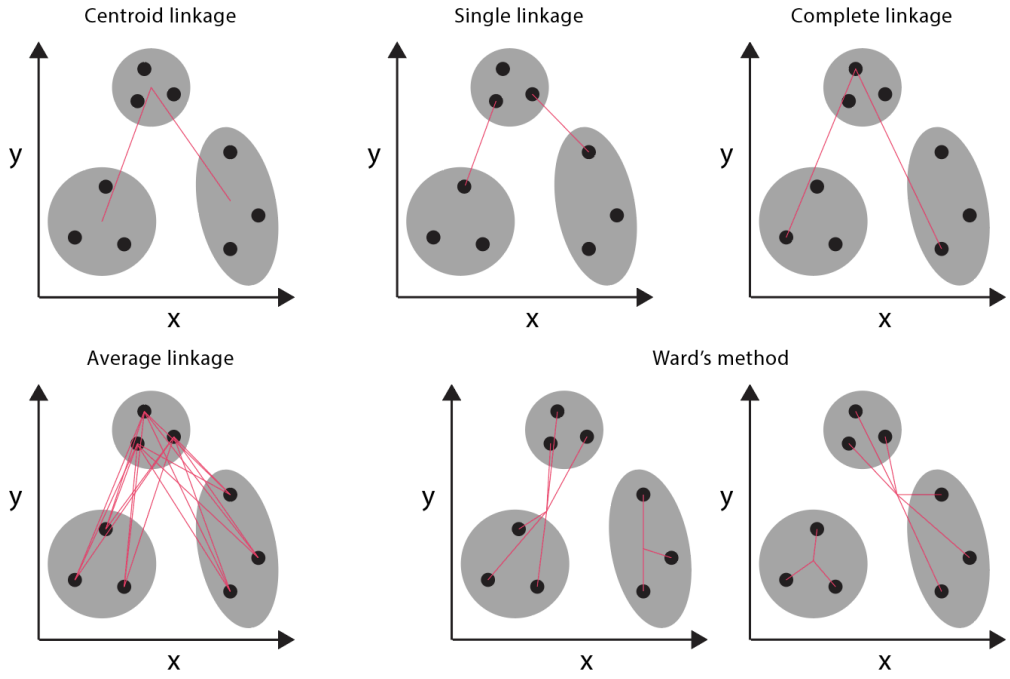



**Figure 18.** Illustration of different linkages (reproduced from: Jeon et al. (2017)).

A single linkage (minimum linkage or nearest-neighbour method) combines two clusters, A and B, if the measured distance between any point in cluster A is less than the defined minimum distance. A single linkage method can result in long chain-like clusters and is sensitive to noise and outliers with minimal distance. The single-linkage distance $d(A, B)$ between clusters $A$ and $B$ is defined as follows: (Gere, 2023; Johnson, 1967)

$$d(A, B) = \min_{x \in A, y \in B} d(x, y)$$

Here $d(A, B)$ is the single-linkage distance between clusters, $x$ and $y$ are the individual datapoints in clusters $A$ and $B$, respectively, and $d(x, y)$ is the distance between datapoints.

Complete linkage (maximum linkage or furthest neighbour method) defines the distance between two clusters as the maximum distance between any single pair of points in the two clusters. Complete linkage produces compact clusters but is overly sensitive to outliers. (Gere, 2023; Johnson, 1967)

The complete linkage distance $d(A, B)$ between the clusters $A$ and $B$ is defined as follows:

$$d(A, B) = \max_{x \in A, y \in B} d(x, y)$$

Here $d(A, B)$ is the complete linkage distance between clusters, $x$ and $y$ are the datapoints in clusters $A$ and $B$, respectively, and $d(x, y)$ is the distance between datapoints.

The average linkage defines the distance between two clusters as the average distance between all pairs of points in the two clusters. This linkage is between a single and a complete linkage and is a compromise that is less sensitive to noise and outliers. (Gere, 2023; Sokal, Michener, & of Kansas, 1958) The average linkage distance $d(A, B)$ between clusters $A$ and $B$ is defined as follows:

$$d(A, B) = \frac{1}{|A||B|} \sum_{x \in A} \sum_{y \in B} d(x, y)$$

Here $d(A, B)$ is the average linkage distance between clusters, $x$ and $y$ are the datapoints in clusters $A$ and $B$, respectively, and $d(x, y)$ is the distance between datapoints.

A single linkage has the tendency of linking clusters together in cases where datapoints in cluster A are close to clusters B datapoints, which might be further away from the clusters B centre than other datapoints. Complete and average linkages do not have this tendency and create more compact clusters with approximately equal diameters. However, they may not merge clusters close together if there are outliers. (Hastie et al., 2009)

The centroid linkage defines the distance between two clusters as the distance between the centroids (geometric centres) of each cluster. A centroid linkage can result in clusters that minimise the within-cluster variance. However, it is not always monotonic, meaning that it can sometimes produce non-intuitive clustering. (Gere, 2023; Lance & Williams, 1967)

The centroid linkage distance $d(A, B)$ between clusters $A$ and $B$ is defined as the distance between their centroids as follows:

$$d(S, B) = d(\mathbf{a}, \mathbf{b})$$

The centroids $\mathbf{a}$ and $\mathbf{b}$ were calculated as follows:

$$\mathbf{a} = \frac{1}{|A|} \sum_{x \in A} x$$

$$\mathbf{b} = \frac{1}{|B|} \sum_{y \in B} y$$

Ward's method seeks to minimise the total variance within the cluster. At each step, the pair of clusters that leads to the smallest increase in total variance within the cluster is merged. Ward's method tends to produce clusters of roughly equal size and is effective at minimising the overall variance within clusters. (Gere, 2023; Ward Jr, 1963)

The Ward linkage distance $d(A, B)$ between clusters $A$ and $B$ is defined as follows:

$$d(A, B) = \frac{|A||B|}{|A| + |B|} \|\mathbf{a} - \mathbf{b}\|^2$$

where centroids $\mathbf{a}$ and $\mathbf{b}$ are given by:

$$\mathbf{a} = \frac{1}{|A|} \sum_{x \in A} x$$

$$\mathbf{b} = \frac{1}{|B|} \sum_{y \in B} y$$

The purpose of the Ward method is not to optimise the clusters, but to find the clusters that are most homogeneous (having the most similar datapoints). The Ward method is computationally intensive and assumes spherical cluster shapes (Ward Jr, 1963).

The different linkage methods are used in Article II and are evaluated as part of the results presented in Section 5.

### 3.1.4 Challenges of unsupervised clustering

Unsupervised clustering faces several common challenges. One of the biggest challenges is the determination of the number of clusters. In many use cases, it is not known beforehand how many clusters should be found (Jain & Dubes, 1988). In some use cases, it is possible to give cluster amounts, making the evaluation of the unsupervised learning outcome more feasible. If the number of clusters is not known beforehand, it is possible to use methods like the elbow method (Thorndike, 1953), silhouette analysis (Rousseeuw, 1987), or the gap statistic (Tibshirani, Walther, & Hastie, 2001). These methods typically require human analysis and may not be conclusive. In this dissertation, the determination of the amount of

the clusters is not a challenge, as it is known that there should be three clusters. This knowledge can be used to evaluate clustering results and it can also be provided as a parameter to clustering algorithms.

If the data is very high-dimensional, it may be difficult to make clusters out of it. In practice, this is related to the dimension calculation. If multiple dimensions have multiple scales, the measurement distance metrics might be a challenge. The choice of the distance metric greatly influences the clustering results. Although the Euclidean distance is common, it may not be appropriate for high-dimensional data. In some cases, dimensionality reduction techniques such as PCA (Principal Component Analysis) (Hotelling, 1933; Pearson, 1901) or t-SNE (t-Distributed Stochastic Neighbour Embedding) (Hinton & Roweis, 2002) can address this challenge (Steinbach, Ertöz, & Kumar, 2004). This dissertation uses LAB colour data as its source data for clustering algorithms, and while the complexity of the data is not very high, but more important is to consider how three dimensions are related to correct distance metrics.

Unsupervised clustering algorithms are computationally intensive. This is a challenge if a large dataset is used. Some clustering algorithms, such as K-means or C-means, can work with large datasets. However, complex algorithms such as hierarchical clustering or DBSCAN may struggle with large datasets. (Hastie et al., 2009; D. Xu & Tian, 2015)

Many clustering algorithms, such as K-means, assume that the clusters are spherical and of equal size. However, real-world data often contain clusters of arbitrary shapes and varying sizes. The size and shape of the cluster may not be known beforehand; thus, selecting a suitable algorithm may not be a clear choice. Density-based methods such as DBSCAN and GMM can handle arbitrary shapes; however, they have their limitations. (Han et al., 2012a) The nature of the use case presented in this dissertation leads to spherical shapes. However, these shapes are not perfect in the form of a ball, rather being stretched towards each other as the colour changes to another.

One of the most typical challenges in clustering is noise and outliers. Depending on the algorithm used, these may have a large impact on the final clustering result. Some algorithms (such as DBSCAN) are designed to be robust against such anomalies, and in some cases preprocessing of the data is necessary. (Hastie et al., 2009; Hodge & Austin, 2004) Noise is also a challenge in this dissertation, as when a change from one colour to another occurs, there are pixel values between these two colours. Also, as seen in previous sections, noise has multiple sources.

Some algorithms, such as the K-means algorithm, are sensitive to initial conditions and can converge to different solutions based on the starting points. Multiple runs with different initialisation setups or using more sophisticated initialisation techniques like K-means++, can help mitigate this problem. (James et al., 2023; Sinaga

& Yang, 2020)

Depending on the dataset used, it is often necessary to scale or normalise the values prior to clustering, especially when using distance-based methods. In normalisation, values are scaled to a specific range, for example 0.0 ... 1.0. Inappropriate scaling can lead to misleading results because some features with large ranges may dominate the distance calculations. (Han et al., 2012a) Some normalisation methods commonly used are min-max normalisation, z-score normalisation, and normalisation by decimal scaling. Normalisation also speeds up data processing because calculations with smaller numbers are faster.

Addressing these challenges often requires a combination of algorithmic adjustments, visual data inspection, preprocessing steps, and domain-specific knowledge.

### 3.1.5 Model Evaluation and Validation

Evaluating clustering results, and their quality, is inherently challenging due to the lack of ground-truth in unsupervised learning. External validation can only be used to evaluate the quality of the clustering results if there is a ground-truth. External validation can be performed by comparing the results with an external set of labels or ground-truth data. Ground-truth data represent the true cluster memberships. External validation helps to assess how well the clustering algorithm has performed in relation to known classes or groups. (J. Wu, Chen, Xiong, & Xie, 2009) As such, external validation is suitable for selecting the best clustering algorithm for a given dataset (Liu et al., 2013), and can be performed using some of the following metrics which compare the clustering results to the ground-truth labels.

- The Rand Index (RI) measures the similarity between two data clusterings, which can be the result of two different algorithms or between algorithm and ground truth. The Adjusted Rand Index (ARI) adjusts the Rand Index for chance that may occur between clusters, providing a more accurate measure of clustering quality. The Adjusted Rand Index ranges between -1.0 and 1.0, where 1.0 indicates that clusters have perfect agreement, and 1.0 indicates that the clusters are totally different. The value 0 indicates random agreement. (Rand, 1971)
- Mutual Information (MI) measures the amount of information shared between clustering assignments and ground truth labels. The adjusted mutual information (AMI) version of the MI adjusts the mutual information for chance and offers a normalised measure. (Kreer, 1957; Shannon, 1948)
- The purity measures the extent to which clusters contain a single class, with higher purity indicating better clustering performance. (Rendón et al., 2011)

- Fowlkes-Mallows Index (FMI) considers the geometric mean of precision and recall evaluating clustering quality. (Fowlkes & Mallows, 1983)

In addition, external validation metrics dependent on the use case can be used, as shown in Section 5 and Article II, where the clustering results were evaluated against the ground-truth by using CIEDE2000 colour difference calculation between the ground-truth and the clustering result. Delta-E difference was then used to determine whether the clustering was performed correctly (success) or incorrectly (failure). The success rate for the entire dataset was calculated using the standard formula: *success rate = correctly clustered images / total images.*

Internal validation measures can be used to choose the best clustering algorithm if no external information (ground-truth) is available. In contrast to external validation, internal validation can also be used when choosing the optimal cluster amount. Internal validation of the clustering results have two main metrics: 1) compactness, i.e. how closely datapoint are in a cluster and 2) separation, i.e. how separated a cluster is from other clusters. Sometimes density-based measures are also used. The following methods are common for internal clustering validation. (Liu et al., 2013)

- Davies-Bouldin Index (DBI) measures the average similarity ratio of each cluster to its most similar cluster. DBI uses the ratio of within-cluster scatter to between-cluster scatter. A smaller DBI indicates better clustering. (Davies & Bouldin, 1979)
- The silhouette score measures how similar an object is to its own cluster compared to other clusters. The silhouette score ranges from -1.0 to 1.0, where higher scores indicate that objects are well matched to their own cluster and poorly matched to neighbouring clusters. (Rousseeuw, 1987)
- Within-Cluster Sum of Squares (WCSS) quantifies the total variance within each cluster. WCSS is calculated by summing the squared differences between each datapoint and the centroid of its respective cluster. WCSS reflects the compactness of the clusters and a lower WCSS indicates a more compact cluster. (Hartigan & Wong, 1979)
- Calinski-Harabaz Index (the Variance Ratio Criterion), measures the quality of clustering by assessing the ratio of the sum of between-cluster dispersion to within-cluster dispersion. Higher values indicate more dense and well-separated clusters.(Caliński & Harabasz, 1974)
- Dunn's index (DI) measures the ratio of the minimum inter-cluster distance to the maximum intra-cluster distance. Higher values indicate better clustering, which indicates well-separated clusters with small within-cluster variance. (Dunn, 1973)

- R-squared (RS), the ratio of the sum of squares between clusters and the total sum of squares of the entire dataset. RS is the degree of difference between clusters. (Sharma, 1995)

- Root-mean-square standard deviation (RMSSTD) is calculated by taking the square root of the pooled sample variance of all attributes. RMSSTD provides information about the homogeneity of clusters. (Sharma, 1995)

The mentioned metrics are just examples of ones that could be used; there are many others, and research of new methods is an interesting topic in the context of unsupervised clustering. If cluster analysis is performed by a human expert, then tools such as scatter plots, dendrograms, and heathmaps can be used to visualise the result of the clustering algorithm.

### 3.1.6 Unsupervised clustering in computer vision

As can be seen from the previous subsection, an unsupervised cluster can be implemented in different ways depending on the given dataset, the use case, and expected results. Unsupervised clustering is suitable for solving many machine learning problems, and is commonly used as an initial machine learning technique for evaluation due to its applicability and easy on-boarding. Unsupervised clustering has several use cases across various domains. These use cases are driven by the unsupervised clustering ability to identify patterns and group similar datapoints without pre-labelled examples.

In the context of computer vision, unsupervised clustering can be used at least for image segmentation / compression and object recognition. Clustering algorithms can partition an image into segments or regions with similar characteristics, which aids in object detection, image analysis, and feature extraction, where the identification of distinct regions is crucial. Gençtav, Aksoy, and Önder (2012) showed that unsupervised methods can segment and classify images of cervical cells. Another similar study by J. Wu et al. (2017) showed that unsupervised clustering can be used to reveal breast cancer subtypes. These studies demonstrate that the use of unsupervised clustering is an option when images need to be classified based on certain features. Bilius and Pentiuc used K-means and ISODATA clustering methods in their research to classify materials in the field, and their dataset was based on hyperspectral imagining. The approach of Bilius and Pentiuc (2020) helps in agriculture, mineralogy, and other industries when planning land use. Schäfer, Heiskanen, Heikinheimo, and Pellikka (2016) also showed that unsupervised clustering is useful for image segmentation when mapping the diversity of tree species in a tropical mountain forest

With unsupervised clustering, it is possible to obtain clusters which share some feature (like colour) from images. Depending on the use case conditions, their properties and/or relations can be used for decision making.

## 3.2 Supervised learning

Supervised learning (Figure 19) differs from unsupervised learning, as the data in the supervised learning is labelled. With labelled data, supervised learning looks to discover patterns related to the data. Thus, the goals of supervised learning models are predetermined and supervised learning attempts to create a model between input and output. This is done in a training process, where the goal is to learn a function that can accurately predict the output labels of new, unseen data. (Bishop & Nasrabadi, 2006)

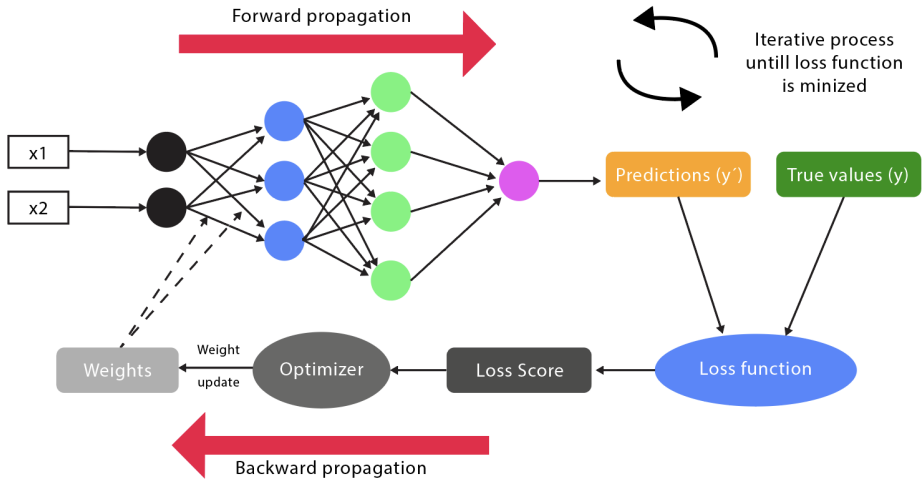



**Figure 19.** The process of supervised learning (reproduced from: Pramoditha (n.d.)).

Supervised learning consists of labelled data (dataset), a model, and a learning algorithm. The dataset is a collection of input-output pairs $(X, Y)$, where $X$ is the input feature vector and $Y$ is the corresponding output label (Bishop & Nasrabadi, 2006). $X$ can be an image or some other object, and $Y$ can be a label that best describes $X$. The dataset is usually divided into training, testing, and validation datasets. The training dataset is used to train the best possible model, the validation dataset is used in each iteration to validate the training results, and the test dataset is used to assess the model performance on unseen data. (Szeliski, 2022) Model is a mathematical function $f(X;\theta)$, where $X$ represents the input feature vector and $\theta$ represents the model parameters. Function $f$ maps input $X$ to the predicted output

$\hat{Y}$. (Deisenroth, Faisal, & Ong, 2020) The choice of model depends on the nature of the problem and the data; the model can be linear regression, logistic regression, decision tree, random forest, support vector or some other model (Shalev-Shwartz & Ben-David, 2014). In this dissertation, neural networks are used as a model for supervised learning. The learning algorithm is used to find the optimal parameters $\theta$ for the model that minimises the error between the predicted and actual results (Deisenroth et al., 2020).

In the supervised learning process, the learning algorithm adjusts the model parameters by minimising a loss function. The loss function measures the difference between the predicted and actual outputs. As a model and learning algorithm, the loss function can also be selected. Common loss functions include the mean squared error for regression and the cross-entropy loss for classification. The adjustment of the model parameters is performed in an iterative process, where the parameters are updated in each iteration. At the end of each iteration, the model is validated with data that are not used for training. Once the training process has gone through a certain number of iterations or some other goal has been reached, the model is ready and can be tested. During the evaluation of the testing model with unseen data, common metrics such as accuracy, precision, recall, F1-score, Mean Squared Error (MSE), and confusion matrix can be used to measure the performance of the model. (Hastie et al., 2009; Shalev-Shwartz & Ben-David, 2014)

### 3.2.1 Loss functions

The loss function is used as a cost or objective function during the training and validation processes. The loss function is a mathematical formula that calculates the discrepancy between the predicted outputs of the model and the actual target values. The loss function value provides a single scalar value that encapsulates how well the predictions of the model align with the true values. The training process looks to minimise loss function, where a smaller loss function value indicates a more accurate model. (Bishop & Nasrabadi, 2006; Hastie et al., 2009)

In a mathematical presentation, the loss function $L(y, \hat{y})$ is a function of the predicted output $\hat{y}$ and true target value $y$ (Hastie et al., 2009). The gradient of the loss function is used during backpropagation to update the model parameters (Hecht-Nielsen, 1992). The selection of the loss function affects how the model is trained and optimised.

The mean squared error (MSE) is used in regression tasks to measure the average squared difference between the predicted values ($\hat{y}$) and the true values ($y$). (Hastie et al., 2009)

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

where $n$ is the number of datapoints. MSE penalises larger errors more severely due to the squaring term; this means that outliers have a significant impact on MSE. RMSE (Root Mean Squared Error) is a version of the MSE, where the square root of the MSE is taken, providing a value that directly relates to the scale of the data.

The mean absolute error (MAE), also used for regression tasks, calculates the average of the absolute differences between predicted and actual values.

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

MAE uses absolute difference; thus, it is less sensitive to outliers compared to MSE. MAE can also be a better choice than MSE if there are anomalies in the errors. (Hastie et al., 2009).

Huber Loss attempts to combine the advantages of MSE and MAE, and is useful in the presence of outliers while maintaining the smoothness of MSE. (Huber, 1992)

$$\text{Huber Loss} = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{for } |y_i - \hat{y}_i| \leq \delta \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

where $\delta$ is a threshold parameter.

Hinge Loss is used for binary classification problems, particularly with Support Vector Machines (SVMs). The hinge loss method is designed to maximise the margin between the decision boundary and the datapoints. (Rennie & Srebro, 2005)

$$\text{Hinge Loss} = \max(0, 1 - y_i \cdot \hat{y}_i)$$

where $y_i$ is the true label and $\hat{y}_i$ is the predicted value. Hinge Loss not only penalises the model for incorrect classifications but also those too close to the decision boundary.

Cross-Entropy Loss (Log Loss) is commonly used in classification tasks where the model outputs probabilities. Log Loss measures the difference between the true probability distribution and the predicted distribution. For binary classification, the cross-entropy loss is calculated with: (Hastie et al., 2009; Rubinstein & Kroese, 2004)

$$\text{Cross-Entropy Loss} = -\sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

In multi-class classification Cross-Entropy Loss generalises to:

$$\text{Cross-Entropy Loss} = -\sum_{i=1}^{n} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c})$$

where $C$ is the number of classes, $y_{i,c}$ is an indicator of whether class $c$ is the correct label for instance $i$, and $\hat{y}_{i,c}$ is the predicted probability of class $c$.

Kullback-Leibler Divergence (information divergence or relative entropy) can be used to measure how one probability distribution diverges from a second, expected distribution. Kullback-Leibler Divergence is often used in variational auto-encoders and generative models. Kullback-Leibler Divergence is calculated as follows: (Kullback & Leibler, 1951)

$$\text{KL}(P \| Q) = \sum_{i} P(i) \log \frac{P(i)}{Q(i)}$$

where $P$ and $Q$ are the true and predicted probability distributions, respectively. Kullback-Leibler Divergence is not symmetric, meaning it is not a true distance metric. However, it quantifies the discrepancy between distributions. (Kullback & Leibler, 1951)

The selection of the loss function is a crucial step in the training process. The selection process depends on the given problem and a custom loss function may be required (Barton, Alakkari, O'Dwyer, Ward, & Hennelly, 2021; Brophy, Hennelly, De Vos, Boylan, & Ward, 2022).

### 3.2.2 Evaluation metrics

As a loss function, an important part of the neural network training process are evaluation metrics. Evaluation metrics are used to assess the performance and effectiveness of trained models. Evaluation metrics are mathematical formulas that provide the developer of a neural network with quantitative measures, and with these measures, it is possible to evaluate how well the model generalises to unseen data. As with the loss function, the selection of evaluation metrics depends on the nature of the problem. Common evaluation metrics include accuracy, precision,

recall, F1-score, receiver operating characteristic curve (ROC), and area under the curve (AUC). The loss functions MSE, MAE, and cross-entropy loss can be used as evaluation metrics in regression tasks. In addition, a confusion matrix is used to evaluate classification tasks. (Han et al., 2012b)

Accuracy is defined as the ratio of the number of correct predictions to the total number of predictions:

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

While accuracy is straightforward and easy to interpret, it may not be suitable for imbalanced datasets where certain classes are under-represented (Q. Gu, Zhu, & Cai, 2009). If the dataset has such a property, precision and recall provide more reliable metrics. Precision measures the proportion of true positive predictions out of all positive predictions made by the model, and recall (or sensitivity) measures the proportion of true positives out of all actual positives (Japkowicz & Shah, 2011):

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

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

The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both (Japkowicz & Shah, 2011):

$$\textit{F1-Score} = 2 \cdot \frac{\textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

The Receiver Operating Characteristic (ROC) curve is not a formula, but a graphical representation of the true positive rate versus the false positive rate across different threshold values. The Area Under the Curve (AUC) quantifies the overall performance of the model (Japkowicz & Shah, 2011):

$$\textit{AUC} = \int_0^1 \textit{ROC}(t)\, dt$$

An AUC value of 1 indicates perfect classification, and an AUC value of 0.5 suggests random guessing. The receiver operating characteristic curve and the AUC are useful for evaluating models in binary classification tasks with varying thresholds.

(X. Zhang, Li, Feng, & Liu, 2015)

A confusion matrix is used in classification tasks to evaluate model performance. The confusion matrix provides a detailed breakdown of the model predictions compared to the actual results, helping to identify not only the general accuracy but also where the model is making errors. This helps in the development of models and in the development of pre-processing and data acquisition. (Han et al., 2012b; Japkowicz & Shah, 2011)

A confusion matrix is typically presented as a structured square table with rows and columns representing different classes in the classification task. For a binary classification problem (labelled as Positive and Negative), the matrix is a $2 \times 2$ table:

| | Predicted Positive | Predicted Negative |
|---|---|---|
| Actual Positive | True Positive (TP) | False Negative (FN) |
| Actual Negative | False Positive (FP) | True Negative (TN) |

True positive (TP) denotes the number of instances correctly predicted as positive. True Negative (TN) is the number of instances correctly predicted as Negative. False Positive (FP) denotes the number of incorrectly predicted instances as positive. False Negative (FN) represents the number of incorrectly predicted instances as negative.

A confusion matrix for multi-class classification problems extends the concept of a binary confusion matrix to handle multiple classes. In a multi-class scenario, the confusion matrix is a square matrix, where both rows and columns represent different classes.

For a classification problem with $n$ classes, the confusion matrix will be an $n \times n$ matrix. Each element in the matrix at position $(i, j)$ represents the number of instances where the true class is $i$, and the predicted class is $j$. The diagonal elements represent the number of correct predictions for each class, and the off-diagonal elements represent mis-classifications.

In Figure 20, 13 images of *label 1* were classified as *label 1*, and no images were classified as other labels. Then, in *label 2*, some images have been labelled as *label 3*. Furthermore, all images of *label 3* are correctly labelled. Therefore, the model may require some improvement.

A confusion matrix can be used to obtain insight into the performance of a classification model beyond scalar metrics for example, from the confusion matrix it can be seen which classes are confused and where the model performs well or poorly.

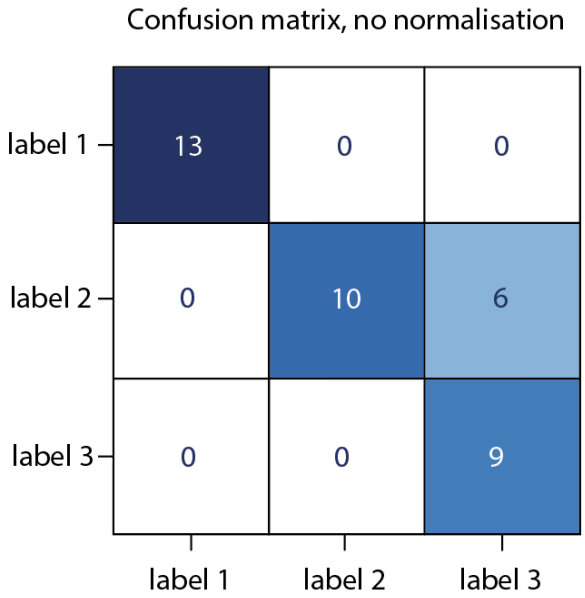



**Figure 20.** Example of confusion matrix for multi-classification.

## 3.3 Neural networks

Neural networks (artificial neural network ANN, or neural net NN) are models that have structure and functions that mimic biological neural networks (Hristev, 1998). Neural networks can be trained with unsupervised or supervised learning methods, supervised methods being the most common approach. Neural networks perform a wide range of tasks in various domains.

In classification tasks, neural networks assign input data to one of several predefined categories, classification can identify objects in images, text classification can identify language nuances, and speech recognition produces text (Bishop & Nasrabadi, 2006). Regression tasks predict continuous values based on input data, and can be used to predict things such as prices or weather conditions (Bishop & Nasrabadi, 2006). In image processing tasks, neural networks, particularly convolutional neural networks (CNNs), are suitable for processing grid-like data. The tasks suitable for CNNs include object detection, image segmentation, and analysis, as well as image generation (Goodfellow et al., 2016). In Natural Language Processing (NLP) tasks, neural networks process and generate human language. The NLP task can be translation, text summarisation, or text generation. Another language-related task is speech recognition, where CNNs are used to convert spoken language into text. (Abdel-Hamid et al., 2014) As in unsupervised learning, neural networks are also suitable for anomaly detection. Artificial Neural Networks (ANNs) can identify unusual patterns or outliers in data, which is useful for fraud detection, network security, and industrial maintenance. (H. Wang et al., 2021). In reinforcement learning tasks, neural networks are used to train agents that learn to make decisions by

interacting with the environment. This task is suitable for developing artificial intelligence gameplay, robotics, and autonomous driving. (Souchleris, Sidiropoulos, & Papakostas, 2023) In time series analysis, neural networks, especially recurrent neural networks (RNNs) and long- and short-term memory networks (LSTMs), are used to analyse and forecast time series data (Goodfellow et al., 2016). This is closely related to regression tasks and can be used to predict sales and forecasts. One of the latest developments in ANNs is generative models, such as generative adversarial networks (GANs). GANs can be used to create new data samples that resemble a given dataset. GANs can generate realistic images, music, and text. (Szeliski, 2022)

Neural networks are composed of individual units, called nodes or artificial neurons, which are connected to each other through edges. Depending on the architecture of the neural network (ANN model), there can be varying numbers of nodes across different layers of the network. These layers and the number of nodes in them define the width and depth of the model. The nodes in each layer receive input signals from the nodes in the previous layer, process the signal, and then pass the processed signal to the nodes in the next layer. The signal received by each node is a weighted sum of the inputs from the connected nodes, along with a bias term. The node then applies a non-linear activation function to this weighted sum to produce an output. The input to the neural network consists of the characteristic values relevant to the task, such as images, audio, or text documents. The final output of the network provides a solution to the problem, such as classifying an image or predicting an outcome based on the input features. (Hristev, 1998)

The layers of the ANN begin from the input layer and end at the output layer. These layers are called visible layers. The layers between the input and output layers perform different transformations to their inputs, and are referred to as hidden layers. An ANN is considered deep if it has at least two layers between the input and output layers. The ANN layers can therefore be broadly categorised into three types: input, hidden, and output layers. (Haykin, 2009; Hristev, 1998)

The input layer receives the raw input data. The neurons in the input layer do not modify or perform any computation; they simply pass the data to the subsequent layers. The number of nodes correlates with the size of the input. After the input layer, the hidden layers perform transformations and extract features from the input data. Depending on the model used and the complexity of the task, different numbers of layers and neurons in layers are used. (Hristev, 1998) In addition, the types of hidden layers vary; some common types of layers include (Goodfellow et al., 2016):

- Dense (Fully Connected) Layers connect to every neuron in both the preceding and subsequent layers. These layers compute a weighted sum of the inputs, followed by the application of an activation function to introduce non-

linearity.

- Convolutional Layers apply learnable convolutional filters (kernels) to the input data in order to extract spatial features such as edges, textures, and patterns. These layers are particularly effective for image and spatial data processing.
- Recurrent Layers are designed for processing sequential data. They maintain a form of memory by passing information across time steps, making them well-suited for tasks such as time series analysis and natural language processing.
- Pooling Layers reduce the spatial dimensions of the feature maps, typically by computing the maximum or average value within a sliding window (e.g., max pooling or average pooling). This operation lowers the computational burden and helps to mitigate overfitting by reducing spatial redundancy.
- Dropout Layers randomly deactivate a fraction of the input neurons during each training iteration. This regularization technique prevents co-adaptation of neurons and encourages the network to learn more robust and generalizable features.

The output layer is the final layer of the neural network. The number of neurons in the output layer corresponds to the number of classes in the classification task and the number of output values in the regression task. The output layer frequently uses activation functions, such as softmax for classification and linear activation for regression. (Goodfellow et al., 2016; Hristev, 1998)

### 3.3.1 Convolutional Neural Networks (CNN)

A Convolutional Neural Network (CNN) is a specialised artificial neural network. CNN is designed for processing and analysing grid-like data, particularly images. CNNs are designed to reduce the number of parameters and computational complexity, which is important when images and other high-dimensional grid-like data are processed. (Goodfellow et al., 2016)

The core component of a CNN (Figure 21) is the convolutional layer, and works as follows. The convolutional layers apply convolutional filters (kernels) to the local regions of the input data. These filters are passed across the input to produce feature maps. Feature maps capture local patterns, such as edges, colour information, textures, and shapes. Filters are specialised to recognise a specific type of feature, and during the learning process, the network learns the optimal filters for the given task. (Goodfellow et al., 2016)

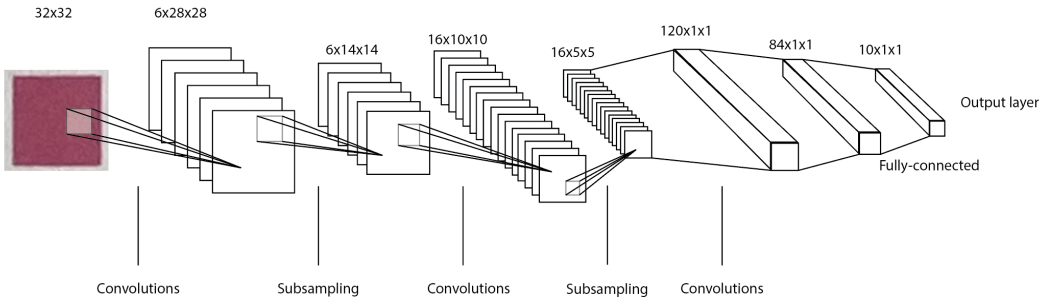



**Figure 21.** Example of LeNet CNN-architecture (reproduced from: J. Gu et al. (2018)).

CNNs use pooling layers to down-sample feature maps. Downsampling reduces the spatial dimensions and number of parameters in the network, thereby reducing the complexity and computational costs of the network. Downsampling also helps reduce overfitting and improves the ability of the network to generalise. Common pooling operations include max pooling, which takes the maximum value in a window, and average pooling, which computes the average value. (LeCun et al., 1989; LeCun, Bottou, Bengio, & Haffner, 1998)

One feature of a CNN is the usage of non-linear activation functions, such as the Rectified Linear Unit (ReLU). ReLU is designed to introduce non-linearity into the model, which is required when patterns and relationships within the grid-like data are recognised. ReLU helps to accelerate the convergence of training by mitigating the vanishing gradient problem, which often occurs with other activation functions like the sigmoid or tanh. (Goodfellow et al., 2016; Nair & Hinton, 2010)

In the final stages of a CNN, fully connected (dense) layers are used to integrate the high-level features extracted by the convolutional layers. The fully connected layers perform classification or regression tasks based on the learnt representations. (Goodfellow et al., 2016)

### 3.3.2 Multilayer Perceptrons (MLPs)

Multilayer Perceptrons (MLPs) (Figure 22) are a class of feedforward artificial neural networks composed of multiple layers of nodes, where each node (or neuron) is connected to every node in the subsequent layer. The fundamental structure of an MLP includes an input layer, one or more hidden layers, and an output layer. Each connection between nodes is associated with a weight that is adjusted during training to minimize prediction error. (Fiesler & Beale, 2020; Goodfellow et al., 2016)

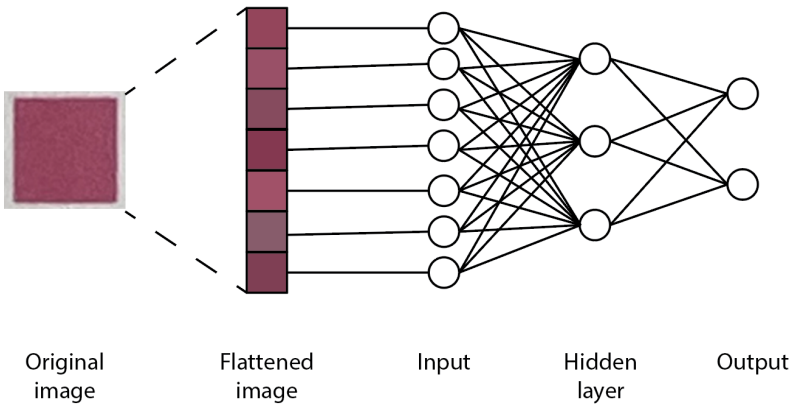



**Figure 22.** Multilayer Perceptron architecture.

An MLP operates by performing a series of linear transformations followed by non-linear activation functions (Fiesler & Beale, 2020).

$$\mathbf{y} = f(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})$$

where $\mathbf{x}$ is the input vector, $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector, and $f$ is a nonlinear activation function such as the sigmoid, hyperbolic tangent (tanh), or the more commonly used Rectified Linear Unit (ReLU). The nonlinearity enables MLPs to learn complex, non-linear mappings between inputs and outputs. (Bishop, 1995; Fiesler & Beale, 2020)

MLPs are trained using supervised learning, typically via backpropagation in conjunction with gradient descent optimization algorithms such as stochastic gradient descent (SGD), Adam, or RMSprop. During training, the weights are iteratively updated to minimize a chosen loss function, such as mean squared error for regression tasks or cross-entropy loss for classification. (Fiesler & Beale, 2020)

The primary distinction from CNNs lie in how these models process input data. MLPs operate on flattened input vectors, thereby disregarding any inherent spatial structure. CNNs also are more parameter-efficient than MLPs. While MLPs are fully connected—resulting in a large number of parameters that increase rapidly with input size, CNNs leverage weight sharing through local receptive fields, significantly reducing the number of learnable parameters. This efficiency allows CNNs to scale better and generalize more effectively in tasks involving high-dimensional input. While MLPs offer simplicity and computational efficiency for non-spatial data, CNNs are inherently better equipped for tasks involving image data due to their ability to model local dependencies and spatial hierarchies. (Botalb, Moinuddin, Al-Saggaf, & Ali, 2018; Haykin, 2009)

### 3.3.3 Forward and backward propagation

Forward propagation involves passing input data through the network. The process begins with the input layer, where the initial data $\mathbf{x}$ is input to the network. The layers after input process the data using a set of weights $\mathbf{W}$ and biases $\mathbf{b}$, applying an activation function $\mathbf{f}$ to introduce nonlinearity. (Bishop & Nasrabadi, 2006; Goodfellow et al., 2016)

For a layer $l$, the input $\mathbf{a}^{(l-1)}$ from the previous layer is transformed as follows (Bishop & Nasrabadi, 2006; Goodfellow et al., 2016):

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

where $\mathbf{z}^{(l)}$ is the weighted sum of the inputs. The activation function $f$ is then applied to $\mathbf{z}^{(l)}$ to produce the output $\mathbf{a}^{(l)}$:

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$$

The process is repeated layer by layer, propagating the activation forward through the network until it reaches the output layer. For example, in a neural network with $L$ layers, the output $\mathbf{a}^{(L)}$ is computed after processing through all intermediate layers. (Bishop & Nasrabadi, 2006; Goodfellow et al., 2016) If the network is working on a classification problem, the final result is commonly passed through a softmax function, which produces a probability distribution over the classes. (Bishop & Nasrabadi, 2006; Goodfellow et al., 2016)

During training, forward propagation provides the predicted output required to calculate the loss function, which quantifies the error between the predicted and actual outputs. This error is used in backpropagation to adjust the network weights and biases in order to minimise the loss function and improve model performance. (Hristev, 1998; LeCun, Touresky, Hinton, & Sejnowski, 1988)

Backpropagation works vice versa as forward propagation, and enables the model to learn from the data by adjusting weights and biases to minimise the loss function. In the iterative optimisation process, backward propagation follows forward propagation. After forward propagation generates the output, the loss function $\mathcal{L}$ measures the difference between the predicted output and the actual target. The backpropagation updates the network parameters to reduce this loss. (Hristev, 1998; LeCun et al., 1988)

Backward propagation begins by computing the gradient of the loss function relative to the output of the network. Using the chain rule of calculus, these gradients are

propagated backward through the network layers to calculate the gradients of the loss function relative to weights and biases. For a given layer $l$, the gradient of the loss $\mathcal{L}$ with respect to the weights $\mathbf{W}^{(l)}$ and biases $\mathbf{b}^{(l)}$ is computed as follows: (Hristev, 1998; Rumelhart, Hinton, & Williams, 1986)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \delta^{(l)} \mathbf{a}^{(l-1)^T}$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$$

where $\delta^{(l)}$ represents the error term of layer $l$, and $\mathbf{a}^{(l-1)}$ is the activation of the previous layer. The error term $\delta^{(l)}$ is computed recursively using:

$$\delta^{(l)} = (\mathbf{W}^{(l+1)^T} \delta^{(l+1)}) \odot f'(\mathbf{z}^{(l)})$$

where $\odot$ denotes the element-wise Hadamard product, and $f'(\mathbf{z}^{(l)})$ is the derivative of the activation function applied to the input $\mathbf{z}^{(l)}$.

Once the gradients are computed for all layers, the weights and biases are updated using an optimisation algorithm such as gradient descent as follows:

$$\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}}$$
$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}}$$

where $\eta$ is the learning rate, which controls the size of the steps taken to minimise the loss.

The backpropagation is repeated for many iterations over the training dataset until the model converges to a solution with minimal loss. This process allows the neural network to learn the optimal parameters that best fit the data, thus improving its performance for the given task. (Goodfellow et al., 2016)

### 3.3.4 Optimisation algorithms

Optimisation algorithms enable neural networks to learn from data by adjusting their weights and biases to minimise prediction errors. Learning algorithms use optimisation algorithms to iteratively update network parameters to reduce the loss function. Previous research has developed several learning algorithms suitable for

different problems. (Goodfellow et al., 2016)

One of the most used optimisation algorithms is Gradient Descent (GD), which works by iteratively adjusting the model parameters in the direction of the steepest descent of the loss function, based on the gradient (partial derivatives) of the loss relative to the parameters. The update rule for $\theta$ at each iteration $t$ is computed as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

where $\eta$ is the learning rate, which is a hyperparameter that controls the step size, and $\nabla L(\theta_t)$ is the gradient of the loss function $L$ at $\theta_t$. (Cauchy et al., 1847). The correct learning rate is crucial for optimal parameters to be found; too large a learning rate might make the algorithm overshoot the minimum or cause divergence; too small a learning rate makes the training process slow, and the process might not reach the best parameters within the given training epochs. In GD and other functions that do not use the adaptive learning rate, the learning rate can be adjusted by a certain constant, such as 0.1, for every fixed number of training epochs. (Takase, Oyama, & Kurihara, 2018)

GD variants have been developed to address the challenges of GD in optimising convergence speed and computational efficiency. These include the Stochastic Gradient Descent (SGD) and the momentum. SGD updates the model weights using a single training example at a time rather than the entire dataset. This mechanism introduces randomness, which helps escape local minima and explore the parameter space more effectively. The update rule for SGD is calculated with: (Goodfellow et al., 2016)

$$w := w - \eta \nabla L_i(w)$$

where the other variables are the same as in GD, $L_i(w)$ is the loss for a single training example. SGD differs from GD so that it approximates the gradient using only a single datapoint. This saves time compared to GD. However, it can cause the weights to oscillate around the minimum rather than converge smoothly. (Goodfellow et al., 2016)

Momentum has been developed as an enhancement of SGD. The aim of Momentum is to accelerate gradient vectors in the right direction so that learning convergence is faster and oscillation is reduced. The approach used in Momentum is to add a fraction of the previous update to the current update to maintain a velocity vector that accumulates gradients over time. The momentum is calculated using the following equation: (Polyak, 1964; Rumelhart et al., 1986)

$$v := \gamma v + \eta \nabla L(w)$$
$$w := w - v$$

where $v$ is the velocity and $\gamma$ is the momentum term. The momentum term is typically set between 0 and 1.

The previously mentioned optimising algorithms are examples of traditional (non-adaptive) optimisers, and use a fixed or globally adjusted learning rate for all parameters throughout the training process. Versions in which individual parameters are up-dated dynamically can be used instead of traditional algorithms, such Adagrad, Adam, RMSprop and AdaDelta algorithms as examples.

Adagrad (Adaptive Gradient Algorithm) adapts the learning rate for each parameter by scaling it inversely proportional to the square root of the sum of all historical squared gradients. Adagrad can perform larger updates for infrequent parameters and update frequent parameters with smaller values. In the Adagrad model, the learning rate continuously decays, leading to stopping the learning process when it is running for too long. The Adagrad update rule is as follows: (Duchi, Hazan, & Singer, 2011)

$$w := w - \frac{\eta}{\sqrt{G_{ii}} + \epsilon} \nabla L(w)$$

where $G$ is the sum of the squares of past gradients, and $\epsilon$ is a small constant that prevents division by zero.

As Adagrad aggressively decreases the learning rate, AdaDelta was developed to allow the algorithm to continue learning even after many updates. AdaDelta uses a moving window for gradient updates, and this window restricts cumulative past gradients to a fixed size. When AdaDelta is used, there is no need to set the initial learning rate because AdaDelta adapts the learning rates based on historical gradient information. (Zeiler, 2012)

$$G_t := \rho G_{t-1} + (1 - \rho)(\nabla L(w))^2$$
$$\Delta w := -\frac{\sqrt{\Delta w_{t-1} + \epsilon}}{\sqrt{G_t + \epsilon}} \nabla L(w)$$

RMSprop (Root Mean Square Propagation) addresses Adagrads decaying learning rate problem by maintaining a moving average of squared gradients to normalise the gradient. RMSprop tries to maintain the effective learning rate throughout training. The RMSprop update rule is calculated as (Tieleman & Hinton, 2012):

$$G_t := \rho G_{t-1} + (1-\rho)\nabla L(w)^2$$
$$w := w - \frac{\eta}{\sqrt{G_t} + \epsilon}\nabla L(w)$$

where $\rho$ is the decay rate. The decay rate is typically set to 0.9. (Zou, Shen, Jie, Zhang, & Liu, 2019)

The Adam (Adaptive Moment Estimation) optimising algorithm combines the advantages of RMSprop and Momentum. Adam computes adaptive learning rates for each parameter using estimates of the first and second moments of the gradients. The Adam update rules are as follows: (Kingma & Ba, 2014)

$$m_t := \beta_1 m_{t-1} + (1-\beta_1)\nabla L(w)$$
$$v_t := \beta_2 v_{t-1} + (1-\beta_2)(\nabla L(w))^2$$
$$\hat{m}_t := \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t := \frac{v_t}{1-\beta_2^t}$$
$$w := w - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where $\beta_1$ and $\beta_2$ are the decay rates for moment estimates, typically set at 0.9 and 0.999, respectively. (Zou et al., 2019)

Comparison and research of optimising algorithms is a common topic in neural networks research. This ongoing development has led to various other optimising algorithms and also results that guide choosing the most suitable learning algorithm in relation to the used neural network and challenge.

### 3.3.5 Over- and underfitting

The training process of a neural network can face two major problems, overfitting and underfitting. In overfitting, the model learns the details and noise in the training data. The core of the overfitting problem lies in the excessive use of neurons in the network. In overfitting, the model captures not only the underlying patterns but also the random fluctuations and outliers in the training data. (Goodfellow et al., 2016)

This negatively impacts the performance of the model on unseen data and prevents the model from generalising on a given task. The reasons for overfitting vary. The

most common reasons include a small or biased dataset, noise in training samples, high variance in model predictions, too complex a model, and not stopping the training procedure before convergence. (Bejani & Ghatee, 2021).

In overfitting, the model performs exceptionally well on the training data, which can be observed as low error rates. However, during the validation phase the model fails to generalise. This leads to high error rates and a significant gap between training and validation performance metrics, such as accuracy, precision, recall, and loss values. (Deisenroth et al., 2020; Goodfellow et al., 2016)

There are many ways to control overfitting; Bejani and Ghatee proposed three control mechanisms: passive, active, and semi-active. Passive methods look for the best possible neural network model and hyper-parameter optimisation techniques (Bejani & Ghatee, 2021). Common L1 and L2 regularisation are considered as passive control methods. L1 adds a penalty proportional to the absolute value of the coefficients in the loss function and L2 adds a penalty proportional to the square of the coefficients to the loss function (Tibshirani, 1996). Active control methods introduce noise into the learning model or the training algorithm so that the model cannot memorise the connection between the data and the output (Bejani & Ghatee, 2021). The methods in this control scheme include dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), data augmentation (Szeliski, 2022), normalisation (Rafiq, Bugmann, & Easterbrook, 2001), gradient centralisation (Yong, Huang, Hua, & Zhang, 2020) and the well-chosen activation functions. Semiactive methods can also modify the network by pruning (Sietsma & Dow, 1988) or by building the network as seen in (Bejani & Ghatee, 2021).

One technique not mentioned in the Bejani and Ghatee categorisation is early stopping, which can be considered a form of passive regularisation. Early stopping halts the training process when performance of the model on a validation set ceases to improve, thereby preventing overfitting. If training is stopped too early, the model may be underfitted — too simplistic to capture the underlying patterns in the data. Conversely, if training continues too long, the model risks overfitting to the training data, reducing its ability to generalise to unseen samples (Goodfellow et al., 2016; Prechelt, 2002).

Underfitting typically occurs when the model is too simple or lacks sufficient capacity (i.e., too few parameters) to represent the complexity of the problem. In such cases, the model fails to perform well both on training data and on new, unseen data. (Deisenroth et al., 2020; Goodfellow et al., 2016)

An underfitted model has high bias and low variance. High bias indicates that the model has made strong assumptions about the data and could not adapt to the nuances of the dataset. Low accuracy and/or high error rates in training and validation indicate underfitting. (Goodfellow et al., 2016; Hastie et al., 2009)

Strategies to reduce underfitting include increasing model complexity and feature engineering. Increasing the complexity of the model can be achieved by incorporating additional layers and neurons in a neural network or by using more complex models (Höge, Wöhling, & Nowak, 2018). In feature engineering, more informative features are created from existing data, providing the model with more inputs (Beaugnon & Chifflier, 2018; Emmert-Streib & Dehmer, 2019).

### 3.3.6 Cross-validation

Cross-validation is a process that can be used to obtain more reliable information about a model's performance, especially when available data is limited. Cross-validation also makes it possible to maximise the use of the dataset. Compared to processes without cross-validation, cross-validation provides a more robust estimate of model accuracy, variance, and generalisation error. Cross-validation involves partitioning the dataset into multiple subsets or folds, allowing the model to be trained and tested on different combinations of these folds. Cross-validation not only provides a more accurate estimate of model performance but also mitigates issues such as overfitting. (Craven & Wahba, 1978; Deisenroth et al., 2020)

K-Fold cross-validation (Figure 23) is a commonly used cross-validation technique. The K-fold cross-validation divides the dataset into $k$ equal sized folds. The model is then trained $k$ times, each time using $k - 1$ folds for training and the remaining fold for testing. The items belong to the same fold in each fold. For each iteration of training, different folds were used for training and testing. The overall performance is obtained by averaging the performance metrics for each of the $k$ iterations. (Deisenroth et al., 2020; Hastie et al., 2009) The value of $k$ is not fixed, but some common values include 5 and 10 (Wong & Yeh, 2019). If 5-folds are used, then 80% of the data is used for training in each iteration.

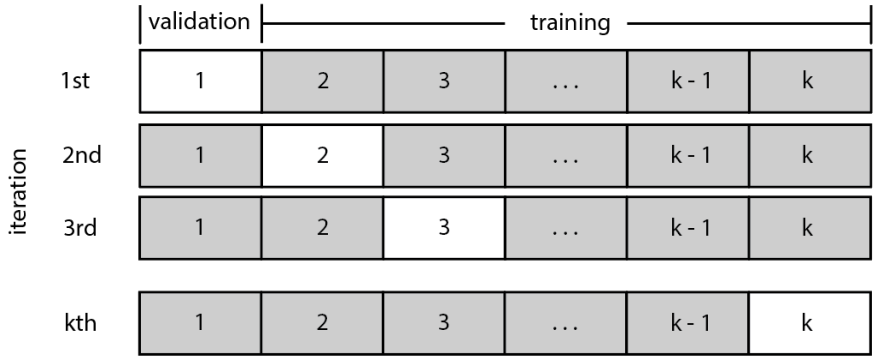



**Figure 23.** Cross-validation process.

Stratified cross-validation is a variation of K-fold cross-validation, where folds are

created to preserve the proportion of each class in each fold. Stratified cross-validation ensures that each fold is representative of the overall class distribution. Stratified cross-validation is especially suitable for unbalanced datasets, where skewed performance metrics may appear. (Han et al., 2012b)

Leave-One-Out Cross Validation (LOOCV) is a special case of K-fold Cross Validation, where $k$ equals the number of items in the dataset. Each fold contains exactly one item for testing, and the remaining $k - 1$ datapoints are used for training. LOOCV is useful for small datasets; however, it can be computationally expensive. LOOCV trains the model on all item points except for one, and tests with a single item. As with the standard K-Fold, the performance metrics are the average of all iterations. (Han et al., 2012b; James et al., 2023)

### 3.3.7 Transfer learning

Recent developments in neural networks and machine learning have led to models that are well trained but are task specific. Transfer learning (Figure 24) can be used to fine-tune these models for another task. Transfer learning leverages knowledge gained from previously solved problems and adapts to solve new but related tasks. Transfer learning can be used to speed up the training process. However, it is also useful if a limited amount of data is available for the new task. In transfer learning, the weights of the pre-trained model are fine-tuned with the new dataset. (Han et al., 2012a; Yosinski, Clune, Bengio, & Lipson, 2014)

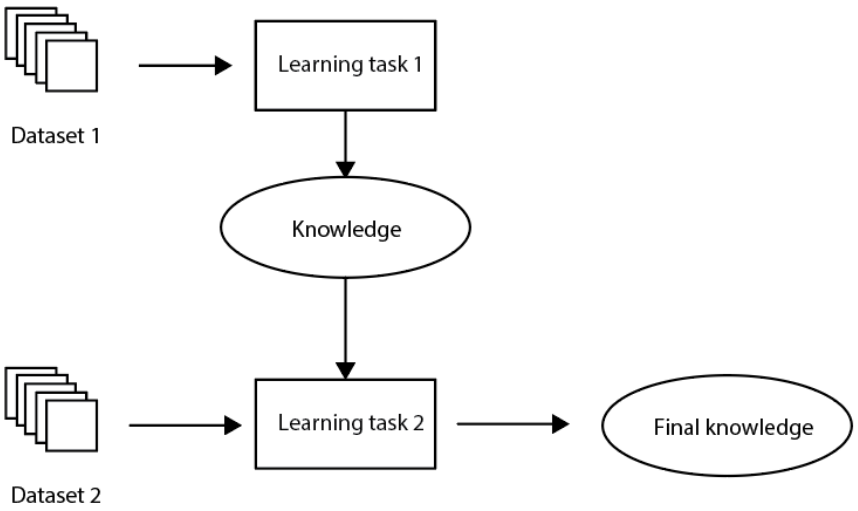



**Figure 24.** Transfer learning process.

Transfer learning is a popular research topic in machine learning, which has led to different categorisation of transfer learning, as well as different approaches. One

way is to categorise transfer learning into inductive, transductive, and unsupervised transfer learning. In inductive transfer learning, the target task differs from the source task. The domain of the task can be similar; however, it does not need to be. In an inductive transfer teaching model, a new domain is considered as multi-task learning. (Niu, Liu, Wang, & Song, 2020)

Multitask learning is an inductive transfer method that enhances generalisation by leveraging the domain knowledge embedded in the training data of related tasks as an inductive bias. This is achieved by learning multiple tasks using common models. In multitask learning, the knowledge gained from another task can help improve the learning of other tasks. (Caruana, 1997)

Multitask learning uses hard or soft parameter sharing. In hard parameter-sharing networks, the shared part is followed by the task-specific section. In soft parameter sharing, each network has its own set of parameters, but the network architecture is the same. Multitask learning is also considered as either homogeneous or heterogeneous. In homogeneous learning, each task corresponds to a single output, and in heterogeneous learning, each task corresponds to a unique set of output labels. This dissertation uses multitask learning, where hard parameter sharing is used to solve homogeneous transfer learning. (Sun, Panda, Feris, & Saenko, 2020)

A typical case of multitask learning is when the model has been trained with a certain (image) dataset, and then this model is modified to solve the problem of another dataset. This can be achieved by maintaining the sharing of the lower-level representations of the model and training only the higher-level layers.

The transfer learning process is typically done in the following steps: 1) selection of the pre-trained model, 2) configuration of the pre-trained model, and 3) training the model for the target domain. The process of configuration can be performed by freezing the pre-trained layers, removing the last (task-specific) layer, and adding a new layer that fits the new task. (Zhong & Ban, 2022). During transfer learning, general feature extraction layers can have their weights ”frozen,” meaning that they are not updated during training on the new task. This helps to preserve the pre-trained knowledge and often leads to faster and more effective learning on the new task. (Goodfellow et al., 2016) These early layers capture fundamental patterns like edges or textures, applicable across various visual tasks.

# 4 METHODS

## 4.1 Dataset for the study

This thesis uses various datasets for Articles I - IV, some datasets are used in multiple articles. To support the creation of a solid and replicable dataset, all materials were printed using a conventional LaserJet office printer on A4-sized plain paper. While not formally standardized, the printing process was kept consistent across all samples to minimize variability in print quality and ensure comparability. The materials were captured with an iPhone 7, an iPhone 11 Pro and a Nokia TA-1032, with their built-in camera applications. All of the camera applications used were set to use automatic settings for white balance and other settings. The purpose was to mimic a consumer taking a photo.

In the first article, colour recognition was performed for a dataset containing 1260 images of a QR code with colour embed inside as a bar The colours used were magenta, cyan, yellow, black, red, green, and purple. For each colour five different intensity levels (100%, 80%, 60%, 40%, 20%) were used. Article I also uses some of the QR codes that were printed with functional inks in black (C = 0.0, M = 0.0, Y = 0.0, K = 1.0 ), magenta (C = 0.0, M = 1.0, Y = 0.0, K = 0.0) and green (C = 0.4, M = 0.0, Y = 0.4, K = 0.0 ). The distribution of samples is shown in Table 1, and each class had at least 12 images that were used for the calculations.

**Table 1.** Distribution of samples in Article I.

| Intensity | Cyan | Purple | Green | Black | Magenta | Red | Yellow |
|---|---|---|---|---|---|---|---|
| 0% | | | | 50 | | | |
| 20% | 19 | 19 | 76 | 19 | 83 | 19 | 13 |
| 40% | 19 | 19 | 83 | 18 | 64 | 19 | 13 |
| 60% | 19 | 19 | 58 | 15 | 71 | 19 | 13 |
| 80% | 19 | 16 | 63 | 19 | 63 | 20 | 13 |
| 100% | 19 | 16 | 129 | 20 | 85 | 19 | 12 |
| Total | 95 | 105 | 409 | 91 | 366 | 96 | 80 |

For Article I, the images were captured both in a controlled environment and in everyday situations (Figure 25). The controlled environment was a 25 cm × 20 cm × 40 cm white box. Inside the box was a moving sledge for distance control and adjustable LED lights for ambient lighting control. In the controlled environment, two levels of ambient light, 400 lx (office ambient light) and 15 lx (dark environment) were used. The images captured in everyday situations were taken in normal home, office, and retail environments. These environments had a varying (warm or cold) colour of light, with ambient light level over 200 lx. However the light level

was not measured due multiple locations used, but it was sufficient for completing general tasks.

**Figure 25.** Samples of images used in Article I.

The dataset used in Article I had challenges that were fixed for the dataset 1 (DS1) used in the Article II. The challenges were related to the bar which was used to embed colour inside the QR code. For DS1 the different colours (black, white and colour) were clearly separated from each other and given square area. In this way, the extraction of the data was more feasible. The final DS1 (Isohanni, 2023) used 25 different modified QR codes with different colour and intensities. QR codes had three colour zones in them: black, white, colour. The colour zone was the colour which was sought to be recognised correctly. The black zone was printed with pure black (100K / CMYK(0.0, 0.0, 0.0, 1.0)) colour, the white zone had no colour (paper white), and the third zone was printed with specific colour. In total DS1 had 722 images distributed as shown in Table 2.

**Table 2.** Distribution of samples in DS1.

| Intensity | Cyan | Green | Black | Magenta | Yellow |
|---|---|---|---|---|---|
| 0% | | | 20 | | |
| 20% | 25 | 25 | 26 | 33 | 22 |
| 40% | 27 | 34 | 18 | 31 | 34 |
| 60% | 33 | 25 | 20 | 42 | 26 |
| 80% | 27 | 31 | 25 | 32 | 23 |
| 100% | 29 | 40 | 21 | 29 | 24 |
| Total | 141 | 155 | 130 | 167 | 129 |

In the example Figure 26, the colour area has saturation 20% in magenta (20M / CMYK(0.0,0.2,0.0,0.0)). The dataset was created by printing colour areas with different ink saturations (20%, 40%, 60%, 80% and 100%). Figure 27, shows an

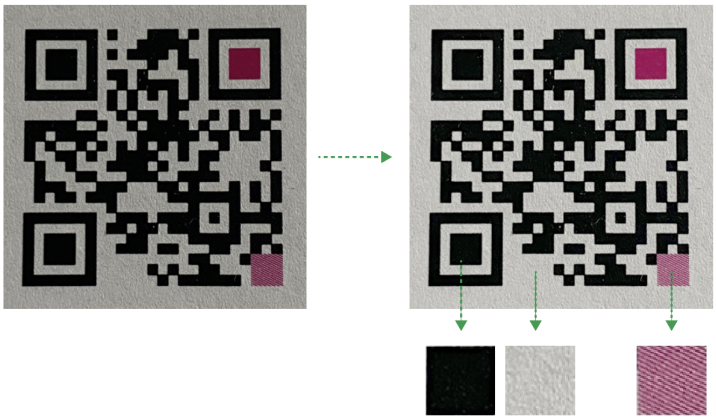


**Figure 26.** One sample of the DS1 and DS2 image, and extraction of colour areas .

example of QR codes with colour saturation 20% ... 100% in a magenta channel (M). The other colours and combinations of colours were C (cyan), M (magenta), Y (yellow), K (black), and CY (green). This dataset was further developed to dataset 2 (DS2) (Isohanni, 2024a) by including additional saturation levels (10% and 5%). This was done so that the algorithms and models could be tested with very subtle differences. DS2 had a total of 561 images, distributed as shown in Table 3.

**Table 3.** Distribution of samples in DS2.

| Intensity | Cyan | Green | Black | Magenta | Yellow |
|---|---|---|---|---|---|
| 0% | | | 52 | | |
| 5% | 45 | 55 | 50 | 47 | 50 |
| 10% | 53 | 55 | 49 | 51 | 54 |
| Total | 98 | 110 | 151 | 98 | 104 |

The images in both datasets were acquired using two consumer-grade mobile devices (distinct iPhone models) under typical ambient lighting conditions. The conditions which are commonly found in residential and office settings. The datasets cover a range of variations in colour temperature, illumination levels, imaging distances, and device-specific camera settings. This variability was intentionally preserved to simulate various real-world scenarios encountered in everyday contexts.

The dataset used was relatively small, so for supervised learning, data augmentation was used with the following options to extend the dataset, rotation of 90 degrees, 0.2 width shift, 0.2 height shift, 0.2 shear range, 0.2 zoom range, horizontal flip and vertical flip.

In Articles III and IV DS1 and DS2 were split into training (80%) and validation (20%) datasets, and in the last experiments of Article IV K-Fold cross-validation

(five times) was also used.


**Figure 27.** Samples of DS1 with varying colour intensity.

## 4.2 Article I - Mathematical methods

Article I explores the embedding of functional ink inside a QR code and how accurately the colour of the indicator can be read using mathematical methods. For this dissertation, the most important part of Article I is colour recognition. Article I uses two different colour areas, white and colour, the mean colour value of these areas is calculated, and then values are compared. The colour difference is calculated using the CIEDE2000 algorithm using different parametric values for $K_L$, $K_C$, and $K_H$. These parameters were CIEDE2000 (1, 1, 1), CIEDE2000 (2.76, 1.58, 1), and CIEDE2000 (2, 1, 1).

## 4.3 Article II - Unsupervised learning

Article II uses the DS1 and DS2 datasets and focusses on using common unsupervised learning methods to recognise colours. In this article, the process used is shown in the following Figure 28. The process takes an RGB JPEG image as input and outputs the LAB values of three (white, black, and colour) cluster centres. After clustering, the cluster closest (smallest DeltaE) to CIELAB(1.0, 0.0, 0.0) is labelled as ”white”. The cluster closest to CIELAB(0.0, 0.0, 0.0) gets the label ”black”. The final cluster is labelled ”colour”. If only two clusters, or more than three major clusters are found, the result of the clustering process is considered to have failed. Finally, Delta-E between the white and colour clusters is calculated. This value indicates how different or similar colours are on a scale 0 ... 100. Clusters were considered to represent different colours if their Delta-E was $> 2.0$. Article II compares the K-mean, Fuzzy C-Means, DBSCAN, Mean shift, Hierarchical clustering, Spectral clustering, Gaussian Mixture Model (GMM), BIRCH and OPTICS algorithms.

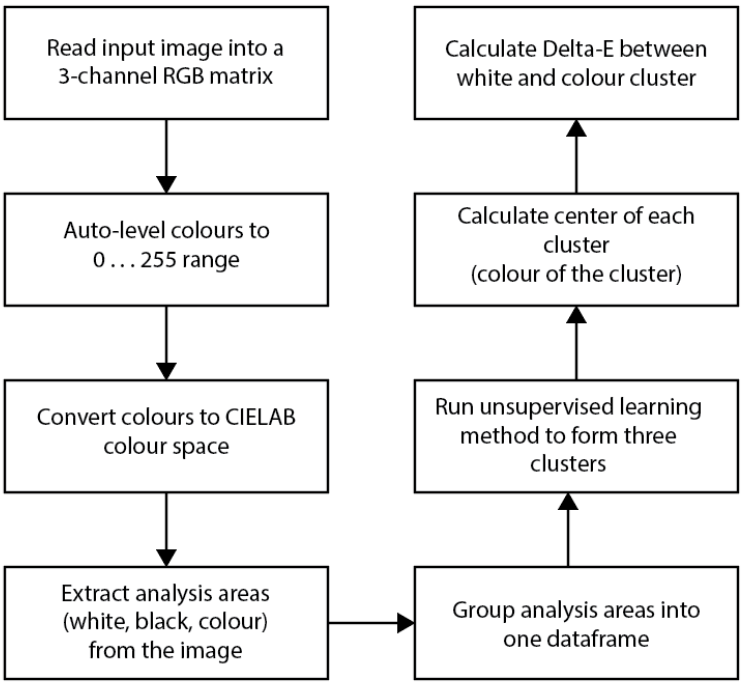



**Figure 28.** The process used in Article II.

## 4.4 Articles III & IV - Convolutional neural networks

Articles III and IV use the DS1 and DS2 datasets. The images are processed partly through the same process as described in Article II, image auto-levelling and an extraction of colour areas. The difference from Article II is that the images are slightly blurred, with Gaussian Blur, to reduce noise that might impact the training process. After this, a difference image is calculated and the image is resized to 256x256. This difference image represents the difference between paper-white and colour, and is labelled with a label that defines colour.

In Article III, the dataset is divided 80% into training data and 20% into validation data. Article IV used K-Fold cross-validation for a final evaluation of the CNN model. These articles use data augmentation to expand the training set. Data augmentation was performed with rotation, width shift, height shift, shear, and zoom options.

Articles III and IV also used transfer learning to transfer knowledge from DS1 to DS2. First, the neural network is trained with DS1. Then, the top layers of the network were changed to match the DS2, and the network is trained again. For the DS1 models were trained with 30 epochs and for the DS2 with 15 epochs.

The Article II used standard versions of each architecture. Optimisations regarding hyper-parameters, such as momentum, learning rate, and batch size, were not used, and exploring them with different architectures was left for further research. All CNN models were compiled using the cross-entropy loss function and the Adam optimiser. The CNNs used in Article III were AlexNet and ZFNet, VGG, ResNet, GoogLeNet, DenseNet and EfficientNet.

Article IV has the same general approach as taken in Article III. In this article, the standard versions of the ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-153 architectures are trained with DS1 and DS2. The architectures were compared with two different approaches, the first being the standard architecture and the second using gradient centralisation (GC). GC is an optimisation technique that normalises gradients during training (Yong et al., 2020). Normalisation of gradients can help the training process to focus on the relevant signals rather than being dominated by noise or extreme gradients. Finally, a custom ResNet-34 was built by experimenting with different modifications to the standard architecture. These modifications include adding a dropout layer to the fully connected layers, adding batch normalisation at the end of the residual block, and using different pooling methods.

# 5 RESULTS

## 5.1 Article I - Mathematical methods

In Article I, first experiments discuss how the indicator embedded inside a QR code affects the decoding distance. The results show that the distance change is only small. This is related to the Reed-Solomon error correction, which reconstruct lost or unreadable data. These results give insight on how embedding should be done, but do not directly contribute to the main theme of the dissertation. The second experiment is more interesting as it experiments different CIEDE2000 algorithms with three different parametric ($K_L$, $K_C$ and $K_H$) values (1, 1, 1), (2.76, 1.58, 1) and (2, 1, 1). These parametric values were selected based on findings of past research done in the field.

The best-performing algorithm in the study was CIEDE2000 with the parameter configuration (2.76, 1.58, 1). This version of the algorithm demonstrated the highest and most consistent accuracy in recognising sensor states, particularly in situations where the colour intensity of the functional ink was low. Compared to the other tested configurations, CIEDE2000(2.76, 1.58, 1) yielded broader and more distinct value ranges, which improved the clarity of sensor state differentiation. Furthermore, in the third experiment involving real-world printed markers with actual functional inks, this parameter setup maintained strong performance by producing stable and interpretable sensor values. Configuration (2.76, 1.58, 1) was determined to be the most accurate and robust option, of the used ones, for detecting the functional ink state in smart tags.

The results of the experiment two in Article I are shown Tables 4, 5 and 6. In these tables each row corresponds to a specific CMYK colour used in the simulated sensor area, including both primary colours (such as cyan, magenta, yellow, and black) and secondary or mixed colours (such as red, green, and purple). The first row (”No sensor”) provides the baseline colour difference range for a non-active sensor area. The columns represent different intensity levels of the sensor ink, ranging from 20% to 100%. For each combination of colour and intensity, tables shows the range of colour difference values calculated by the algorithm. These values reflect how strongly the sensor area differs from the reference (non-sensor) area, with higher values indicating a more detectable change.

The results in Article I show that the CIEDE2000 algorithm, with any of the parametric settings mentioned, can reliably recognise the indicator state accurately if the indicator intensity is 40%. When the indicator intensity decreases to 20% or below, incorrect results are occasionally provided, and the reliability decreases. These incorrect results appeared especially with the yellow ink. The results also demonstrate

| CMYK | 20% | 40% | 60% | 80% | 100% |
|---|---|---|---|---|---|
| No sensor | [0.00 ... 0.05] | | | | |
| (0, 0, 0, 1) | [0.16 ... 0.27] | [0.31 ... 0.60] | [0.58 ... 0.87] | [0.82 ... 0.95] | [0.60 ... 1.00] |
| (1, 0, 0, 0) | [0.07 ... 0.17] | [0.17 ... 0.36] | [0.28 ... 0.36] | [0.37 ... 0.49] | [0.47 ... 0.61] |
| (0, 1, 0, 0) | [0.06 ... 0.24] | [0.16 ... 0.31] | [0.29 ... 0.46] | [0.29 ... 0.61] | [0.43 ... 0.80] |
| (0, 0, 1, 0) | [0.04 ... 0.12] | [0.08 ... 0.17] | [0.08 ... 0.16] | [0.12 ... 0.14] | [0.14 ... 0.23] |
| (1, 1, 0, 0) | [0.15 ... 0.28] | [0.41 ... 0.51] | [0.64 ... 0.69] | [0.81 ... 0.88] | [0.75 ... 0.95] |
| (0, 1, 1, 0) | [0.08 ... 0.18] | [0.17 ... 0.31] | [0.22 ... 0.47] | [0.35 ... 0.61] | [0.44 ... 0.61] |
| (1, 0, 1, 0) | [0.10 ... 0.25] | [0.22 ... 0.31] | [0.31 ... 0.55] | [0.39 ... 0.42] | [0.53 ... 0.75] |

**Table 4.** Results from the Article I, second experiment, CIEDE2000(1, 1, 1) algorithm.

| CMYK | 20% | 40% | 60% | 80% | 100% |
|---|---|---|---|---|---|
| No sensor | [0.00 ... 0.05] | | | | |
| (0, 0, 0, 1) | [0.16 ... 0.27] | [0.31 ... 0.60] | [0.58 ... 0.87] | [0.82 ... 0.95] | [0.60 ... 1.00] |
| (1, 0, 0, 0) | [0.08 ... 0.16] | [0.19 ... 0.36] | [0.30 ... 0.37] | [0.41 ... 0.50] | [0.50 ... 0.63] |
| (0, 1, 0, 0) | [0.10 ... 0.24] | [0.23 ... 0.34] | [0.36 ... 0.49] | [0.36 ... 0.63] | [0.48 ... 0.81] |
| (0, 0, 1, 0) | [0.11 ... 0.15] | [0.16 ... 0.21] | [0.21 ... 0.39] | [0.30 ... 0.33] | [0.27 ... 0.32] |
| (1, 1, 0, 0) | [0.19 ... 0.29] | [0.28 ... 0.38] | [0.65 ... 0.72] | [0.82 ... 0.90] | [0.75 ... 0.97] |
| (0, 1, 1, 0) | [0.09 ... 0.18] | [0.17 ... 0.31] | [0.23 ... 0.47] | [0.36 ... 0.61] | [0.46 ... 0.63] |
| (1, 0, 1, 0) | [0.14 ... 0.29] | [0.28 ... 0.38] | [0.36 ... 0.66] | [0.45 ... 0.53] | [0.56 ... 0.76] |

**Table 5.** Results from the Article I, second experiment, CIEDE2000(2.76, 1.58, 1) algorithm.

| CMYK | 20% | 40% | 60% | 90% | 100% |
|---|---|---|---|---|---|
| No sensor | [0.00 ... 0.05] | | | | |
| (0, 0, 0, 1) | [0.16 ... 0.27] | [0.31 ... 0.60] | [0.58 ... 0.87] | [0.82 ... 0.95] | [0.60 ... 1.00] |
| (1, 0, 0, 0) | [0.07 ... 0.16] | [0.18 ... 0.36] | [0.30 ... 0.37] | [0.39 ... 0.49] | [0.48 ... 0.62] |
| (0, 1, 0, 0) | [0.08 ... 0.24] | [0.20 ... 0.32] | [0.33 ... 0.48] | [0.33 ... 0.62] | [0.45 ... 0.80] |
| (0, 0, 1, 0) | [0.08 ... 0.13] | [0.13 ... 0.19] | [0.15 ... 0.29] | [0.23 ... 0.25] | [0.22 ... 0.26] |
| (1, 1, 0, 0) | [0.17 ... 0.29] | [0.43 ... 0.52] | [0.64 ... 0.70] | [0.81 ... 0.89] | [0.75 ... 0.95] |
| (0, 1, 1, 0) | [0.09 ... 0.18] | [0.18 ... 0.31] | [0.23 ... 0.48] | [0.36 ... 0.61] | [0.46 ... 0.63] |
| (1, 0, 1, 0) | [0.12 ... 0.27] | [0.25 ... 0.34] | [0.34 ... 0.60] | [0.42 ... 0.47] | [0.54 ... 0.75] |

**Table 6.** Results from the Article I, second experiment, CIEDE2000(2, 1, 1) algorithm.

that the algorithms yield values that vary widely, especially with higher intensities. This means that a threshold can be set for recognition if colours are different, but accurate recognition of colours is not feasible with the CIEDE2000 algorithm when printed sources are used in different environments. In such environments, camera accuracy, ambient light, print quality, and paper quality have a significant impact, so only a limited number of colour shades can be recognised accurately.

## 5.2 Article II - Unsupervised learning

Article II compares different unsupervised clustering methods. In these experiments, the clustering process was considered successful if it could recognise three different clusters and the clusters were formed correctly. The clusters were correctly formed if the Delta-E value between the white and the colour cluster was equal to or smaller than 2.0, compared to the ground-truth. The ground-truth was obtained from the images before the datapoints were combined and sent to an unsupervised algorithm. The Delta-E value 2.0 is considered to be the smallest colour difference that an inexperienced human observer can notice (Mokrzycki & Tatol, 2011).

In Article II, the success rate was calculated for each image in the dataset.

$$success\ rate = \left( \frac{correctly\ clustered\ images}{total\ images} \right)$$

The first experiment explored various unsupervised clustering methods. Feasible parameters or their combinations were used for each method. The results are shown in the following Table 7, where the runtime presented is the average runtime per image.

The results of the experiment show that multiple clustering methods are suitable if the difference in colour between white paper is sufficiently large (equal to or greater than 20%). K-Means, with K-Means++ initialisation, performed well overall, particularly benefiting from its hard clustering nature, although it struggled with low-density ink colours and outliers. C-Means, while generally effective, faced difficulties with close clusters and low-density colours, requiring high-fuzziness parameters for better results. DBSCAN frequently failed due to incorrectly classifying datapoints as noise, especially in mid-intensity colours. GMM performed similarly to K-Means but with fewer issues, and although it struggled with noisy data, it was still the best algorithm in larger differences. BIRCH performed better than expected for non-hierarchical data, but was sensitive to outliers and data ordering. Hierarchical clustering was effective, particularly with Ward linkage, but struggled

**Table 7.** Results of the clustering process algorithm with colour difference $>= 20\%$.

| Method (parameters) | Success rate | Runtime (s.) |
|---|---|---|
| K-means (algorithm = elkan) | 98.1% | 0.052 |
| K-means (algorithm = full) | 98.1% | 0.051 |
| C-means (m = 1.0) | 96.1% | 0.034 |
| C-means (m = 2.0) | 96.6% | 0.027 |
| C-means (m = 3.0) | 97.1% | 0.039 |
| C-means (m = 4.0) | 97.6% | 0.051 |
| C-means (m = 5.0) | 97.6% | 0.050 |
| C-means (m = 6.0) | 97.6% | 0.068 |
| DBSCAN (eps = 2.5, min_samples = 25%) | 19.5% | 0.172 |
| DBSCAN (eps = 5.0, min_samples = 25%) | 70.1% | 0.176 |
| DBSCAN (eps = 10.0, min_samples = 25%) | 78.1% | 0.191 |
| DBSCAN (eps = 10.0, min_samples = 16.7%) | 81.6% | 0.177 |
| DBSCAN (eps = 10.0, min_samples = 33.3%) | 25.2% | 0.167 |
| DBSCAN (eps = 15.0, min_samples = 25%) | 70.1% | 0.181 |
| GMM (covariance_type = full) | 99.0% | 0.085 |
| GMM (covariance_type = tied) | 97.6% | 0.071 |
| GMM (covariance_type = diag) | 99.0% | 0.071 |
| BIRCH (threshold = 1.0, branching_factor = 50) | 78.5% | 0.285 |
| BIRCH (threshold = 0.5, branching_factor = 50) | 81.9% | 0.344 |
| BIRCH (threshold = 0.25, branching_factor = 50) | 85.3% | 0.351 |
| BIRCH (threshold = 0.25, branching_factor = 100) | 84.0% | 0.354 |
| BIRCH (threshold = 0.25, branching_factor = 25) | 83.5% | 0.410 |
| BIRCH (threshold = 0.1, branching_factor = 50) | 85.0% | 0.380 |
| Hierarchical (affinity=euclidean, linkage=ward) | 97.7% | 0.537 |
| Hierarchical (affinity=euclidean, linkage=complete) | 82.9% | 0.414 |
| Hierarchical (affinity=euclidean, linkage=average) | 92.7% | 0.468 |
| Hierarchical (affinity=euclidean, linkage=single) | 84.2% | 0.417 |
| Spectral (affinity=radial basis function) | 98.1% | 2.870 |
| Spectral (affinity=nearest_neighbour) | - | - |
| Meanshift (quantile = 0.5) | 5.2% | 18.30 |
| Meanshift (quantile = 0.75) | 0% | 17.50 |
| Meanshift (quantile = 0.4) | 5.0% | 15.60 |
| OPTICS (eps = 2.5, min_samples = 25%) | 17.7% | 5.120 |
| OPTICS (eps = 5.0, min_samples = 25%) | 70.0% | 24.12 |
| OPTICS (eps = 10.0, min_samples = 25%) | 17.7% | 5.12 |

with specific low-density colours. Spectral clustering performed reasonably well, but was affected by outliers. Meanshift was slow and ineffective in clustering the data. OPTICS, although similar to DBSCAN, achieved only a success rate of 70% and was slower. A closer inspection of each algorithm is presented in Article III. Some common observations are that as clusters become closer to each other, it is very challenging for unsupervised learning to identify to which cluster datapoint belongs to. Most algorithms also suffer from noise and outliers, noise is common in the presented use case and comes from many source and transitions of colours.

Article II also presents another experiment in which the intensity difference with respect to paper-white was 0%, 5%, or 10% in the specified CMYK channels. For this experiment, the best algorithms were chosen K-means, C-means (m = 3.0), GMM (covariance = full), hierarchical clustering (affinity = Euclidean, linkage = Ward) and spectral clustering (affinity = radial basis function). The results of this experiment are shown in Table 8.

**Table 8.** Results of the second experiment, colour difference $<= 10\%$.

| Method (parameters) | 10% intensity | 5% intensity |
|---|---|---|
| C-means (m = 3.0) | 94.8% | 83.3% |
| K-means | 96.3% | 89.3% |
| GMM (covariance = full) | 92.5% | 84.5% |
| Hierarchical clustering (affinity=euclidean, linkage=ward) | 93.4% | 84.5% |
| Spectral clustering (affinity = radial basis function) | 97.8% | 76.2% |

From the results, it can be seen that the success rate of all algorithms was lower than in the first experiment. The algorithms performed quite well when the intensity of the colour was 10%; however, with a small intensity of 5%, only the K-means achieved an almost 90% success rate. The challenges come mainly from the magenta and yellow colours. This is shown in Figure 29, where images a) and b) have a colour intensity of 10% and c) and d) have an colour intensity of 5%.

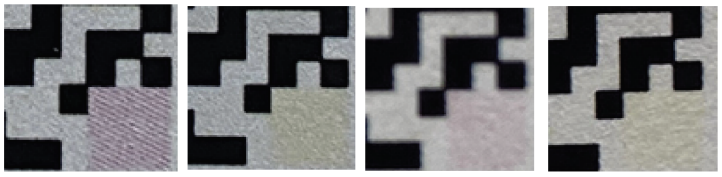



**Figure 29.** Failed images in experiment two.

The results of the Article II show that unsupervised clustering methods, K-means, C-means, GMM, hierarchical clustering, and spectral clustering, can be used to

recognise colour differences in printed CMYK colours. These methods are feasible, especially when the colour has an intensity difference of 20% or more. The best option for these use cases is GMM, which in the experiment presented achieves a 99.0% success rate, and spectral clustering, and a K-means also reached a success rate greater than 98%. All of the best algorithms also performed well when the ink levels were 10%. But when the difference dropped to only 5%, none of the algorithms achieved a success rate higher than 90%. With very low colour intensity differences (5%), the best algorithm was K-means followed by GMM and Hierarchial clustering. Each of the top three algorithms have very different approaches to clustering. Where K-means and Hierarchial clustering are hard clustering, GMM is soft. GMM and Hierarchial clustering assume elliptical or arbitrary clusters and K-means spherical.

The results demonstrate that the unsupervised clustering methods are very sensitive to outliers and noise when they are used for colour data clustering. The noise comes from the change of colour; for example, when black changes to white, there are grey colours in the data. Noise datapoints weaken the results of the clustering by drifting centroids away from their actual locations. Using techniques to filter out noise and outliers would be beneficial, however, the challenge lies in the recognition of noise because they lie close to actual datapoints. With the best approaches, it is noticeable that clustering of yellow and magenta does not work as well as with green and blue. This is interesting as red and yellow lie on the lower end of the colour spectrum as well as on the positive A and B axis in the LAB colour space.

## 5.3 Article III - Convolutional neural networks

Article III used standard CNN architectures to classify images based on their colour difference to paper-white. The same approach as described in Article II was used: first a dataset with larger colour intensity difference was used, and then another dataset with smaller differences. The results of the training process for the first dataset are shown in Table 9, which contains the accuracy and train time/epoch for each architecture.

The results demonstrate that most CNNs outperform unsupervised algorithms. Deeper networks have better performance than shallow ones, except AlexNet, which can be considered shallow with only eight layers. One thing to note is that the AlexNet training speed is the fastest. The results demonstrate that the problem may not require complex or deep networks, as the data used is not very complex. Regarding the best algorithm, DenseNet, which employs a learning approach through residuals, all images were classified correctly. Other architectures made mistakes in some images. ResNet and VGG-16 sometimes make confusions between different intensities of the same colour (VGG-16 in a total 24 images and ResNet in 32 images).

**Table 9.** Results of the first experiment (difference $>= 20\%$) in Article III.

| Architecture | avg. train time / epoch (s.) | Accuracy |
|---|---|---|
| DenseNet | 690 s. | 1.00 |
| ResNet | 626 s. | 0.98 |
| VGG-16 | 570 s. | 0.97 |
| AlexNet | 110 s. | 0.93 |
| EfficientNetB1 | 359 s. | 0.90 |
| ZFNet | 340 s. | 0.88 |
| GoogLeNet | 1639 s. | 0.87 |

AlexNet on the other hand confused some colours with other colours (in 88 images).

Article III also included another experiment with smaller intensity differences, equal to or below 10%. In this experiment, the four best CNN architectures were used. In Article III transfer learning was also used to speed up the training process and to transfer knowledge from DS1 to the DS2 training process. Transfer learning used models of the first experiment to obtain initial weights and biases for DS2 training. These results are shown in the following Table 10.

**Table 10.** Results of the second experiment, colour difference $<= 10\%$.

| CNN | avg train time / epoch (s.) | Accuracy |
|---|---|---|
| VGG-16 | 690 s. | 0.34 |
| ALEXNET | 13 s. | 0.47 |
| DENSENET | 201 s. | 0.77 |
| RESNET | 725 s. | 0.95 |

With smaller colour differences, it becomes a challenge for supervised learning to classify images correctly. When the models are compared, AlexNet and ResNet learnt through the process. However, VGG-16 and DenseNet stopped learning at a certain epoch.

According to the confusion matrix, and further investigation of the per-class performance of the models, most of the models could classify the items correctly if there was no colour presented (class = 0) or when there is at least 10% intensity of some colour. However, when there is only 5% intensity, different classes are confused.

Based on the results presented, the best accuracy was achieved with the ResNet architecture, which exhibited the best overall performance if both experiments are considered. The ResNet architecture can accurately classify other colours, with the exception that low-intensity yellow and magenta are sometimes confused. In addition, in some cases ResNet cannot differentiate 5% yellow from paper-white,

which is also quite challenging for the human eye.

All of the selected CNNs were effective in recognising colour differences, especially those with large intensity differences ($>= 20\%$). When the colour difference decreased to only 5% or 10% (see later Figure 34) in some CMYK channels, most architectures struggled to classify images correctly.

The results demonstrate that different CNN architectures can be used for colour-difference recognition. Deeper CNNs can typically identify more complex features; however, they are more difficult to train. Wider networks can capture more fine-grained features, but they experience difficulties when handling high-level features (see Tan and Le (2019)). In this use case, it is interesting that the depth of the network and the number of parameters are directly correlated with the accuracy of the network when the colour difference is large.

The best architectures (i.e. ResNet and DenseNet) use an approach that connects layers not only to the previous and subsequent layers. However, they either have a dense connection (DenseNet) in which the architecture connects each layer to other layers separately in a feed-forward manner (Zhu & Qiu, 2021), or as seen in ResNet, where the architecture allows data to flow from other layers directly to subsequent layers (W. Zhang, Li, & Ding, 2019). These connectivity types appear to significantly affect performance when the model attempts to classify items based on small differences. As with unsupervised learning, the most challenging colours for neural networks were yellow and magenta.

## 5.4 Article IV - Optimised ResNet

Article IV follows the same structure as Article III, and the first different versions of ResNet, ResNet-18, ResNet-34, ResNet-50, ResNet-101 and ResNet-153 are evaluated together. Then, the best of the architecture is selected for modification to achieve more accurate results. All architectures were tested with and without gradient centralisation. The results of the first experiment are shown in Table 11.

The results demonstrate high general accuracy for all architectures. With GC used, most architectures achieve higher accuracy, but with very deep ResNet-101 and ResNet-153, GC weakens accuracy. This might be an indication that GC accelerates the vanishing gradient problem if the network is very deep. Based on the findings presented in Article IV, most architectures are prone to overfitting with the target dataset. One possible reason for overfitting is small differences in images or a small dataset. ResNet-34 with gradient centralisation demonstrated the best accuracy of 96.9%.

**Table 11.** Results of first experiment (without K-Fold cross-validation) in Article IV, colour difference $<= 10\%$.

| Architecture | Centralised gradient used | avg. train time / epoch (s.) | Accuracy |
|---|---|---|---|
| Resnet-18 | Yes | 323 s. | 0.934 |
| Resnet-18 | No | 316 s. | 0.952 |
| Resnet-34 | Yes | 236 s. | 0.969 |
| Resnet-34 | No | 233 s. | 0.966 |
| Resnet-50 | Yes | 785 s. | 0.958 |
| Resnet-50 | No | 781 s. | 0.901 |
| Resnet-101 | Yes | 1414 s. | 0.941 |
| Resnet-101 | No | 1350 s. | 0.948 |
| Resnet-153 | Yes | 2079 s. | 0.682 |
| Resnet-153 | No | 1094 s. | 0.935 |

Article IV also presents the modified ResNet-34 architecture, where modifications are based on the findings of past research. The modifications were first tested without K-Fold cross-validation, and later final architecture accuracy was tested with K-Fold cross-validation.

Adding extra dropouts can be used to mitigate overfitting. The first dropout was added after the first convolutional layer, and this approach achieved an accuracy of 98.9%. Then, dropout was added after each residual group, which led to a 94.34% accuracy. The results of these modification demonstrate that dropout in the correct location in the architecture can improve model performance in the target dataset. Another way to make the network more resistant to the degradation of the between-class distance to the within-class distance ratio is to include batch normalisation. In the proposed ResNet-34 batch normalisation was added at the end of the residual block. In this approach, the accuracy of the model was reduced by 1.0% from the standard ResNet-34 implementation. Article IV also modified the residual block of ResNet by adding a max-pooling layer after each convolutional operation. This change led to a 95.1% accuracy. The maximum pooling was also tested after both convolutional operations were performed, achieving an accuracy of 99.7%. Then, as the final change of average pooling was changed to global max pooling, this change achieved an accuracy of 99.6%

Some previously mentioned changes improved the accuracy of ResNet34. However, when all or some of them were used together, the accuracy did not improve. This led to the following result: that the best options to improve the accuracy of ResNet-34 are:

- changing average pooling to maximum pooling before the fully connected layer; Architecture A

- using maximum pooling at the end of the residual block; Architecture B

Architectures A and B (Figure 30) showed continuous progress during the training phase, and even with an early stopping, training completed 30 epochs. Architecture A encountered some challenges, with low-level green (10% CY) and yellow (10% Y) images being confused with very low yellow images (5% Y). Architecture B on the other-hand confused different green images (5% CY and 10% CY) and low-intensity yellow to paper white.

| Layer name | output size | custom Resnet - version a |
|---|---|---|
| conv1 | 112 x 112 | 7x7, 64 stride 2 |
| conv2 | 56 x 56 | 3x3 max pool, stride 2<br>[3 x 3, 64<br>3 x 3, 64<br>3x3 max pool, stride 2] x 3<br>[3 x 3, 64<br>3 x 3, 64] x 1 |
| conv3 | 28 x 28 | [3 x 3, 128<br>3 x 3, 128<br>3x3 max pool, stride 2] x 3<br>[3 x 3, 128<br>3 x 3, 128] x 1 |
| conv4 | 14 x 14 | [3 x 3, 256<br>3 x 3, 256<br>3x3 max pool, stride 2] x 5<br>[3 x 3, 256<br>3 x 3, 256] x 1 |
| conv5 | 7x 7 | [3 x 3, 256<br>3 x 3, 256<br>3x3 max pool, stride 2] x 3 |
| | 1 x 1 | average pool, fc, softmax |

| Layer name | output size | custom Resnet - version b |
|---|---|---|
| conv1 | 112 x 112 | 7x7, 64 stride 2 |
| conv2 | 56 x 56 | 3x3 max pool, stride 2<br>[3 x 3, 64<br>3 x 3, 64] x 3<br>[3 x 3, 64<br>3 x 3, 64] x 1 |
| conv3 | 28 x 28 | [3 x 3, 128<br>3 x 3, 128] x 3<br>[3 x 3, 128<br>3 x 3, 128] x 1 |
| conv4 | 14 x 14 | [3 x 3, 256<br>3 x 3, 256] x 5<br>[3 x 3, 256<br>3 x 3, 256] x 1 |
| conv5 | 7x 7 | [3 x 3, 256<br>3 x 3, 256<br>3x3 max pool, stride 2] x 3 |
| | 1 x 1 | maximum pool, fc, softmax |

**Figure 30.** Final architectures A and B.

The final results of K-Fold cross-validation, using 5 folds and 30 epochs, are shown in the following Table 12.

**Table 12.** Final architectures, K-Fold cross-validation accuracy.

| Fold | Architecture A | ResNet-34 | Architecture B |
|---|---|---|---|
| 0 | 97.66 | 95.31 | 97.85 |
| 1 | 99.51 | 98.93 | 99.22 |
| 2 | 96.58 | 97.66 | 97.17 |
| 3 | 96.29 | 98.73 | 98.54 |
| 4 | 100.00 | 96.58 | 98.63 |
| Final | 98.00 | 97.44 | 98.28 |

# 6 DISCUSSION

This research has focused on exploring the possibilities of different methods, colour difference algorithms, and non-supervised and supervised learning in subtle colour difference recognition. The recognition of colours, or their difference, is a common challenge in computer vision research and is commonly used in agriculture, healthcare, civil engineering, and printing domains. Mostly, colour recognition is used to differentiate various colours, but as seen in Section 1.1, small differences in colour are important in the identification of plant disease, the quality control of print, and in medical imaging.

The use case that inspired this research was the recent development of functional inks. Functional inks have been used for a long time to create colour-changing effects on products such as cans, mugs, stickers, and so on. As these inks have developed and their price has come down, new use cases have been found, including consumer-related applications like printed freshness and temperature indicators. In addition, industry is using functional inks for the detection of gasses or humidity, and common for all these use cases is that functional inks indicate their state through a colour change.

The main research question that this research focused on was which of the presented methods is most accurate when subtle colour differences are observed in printed sources. The printed sources used have various characteristics that make the recognition of colours a challenge, and include paper quality, the ink and printer used, and ambient light conditions that are present. The use case presented in this research has focused on the consumer perspective, where it is difficult to control the usage environment or the devices used. Although modern mobile devices have high-range cameras integrated into them, the imaging quality between devices varies greatly.

The work done in the past has shown the possibilities of smartphone cameras in colour detection, for example, Fan et al. presented a digital image colorimetry on smartphone (Fan et al., 2021). As smartphones are not typically used in calibrated or controlled environments, solutions developed for them need the ability to adapt to various situations regarding camera accuracy, ambient light, print, and paper quality (Bagherinia & Manduchi, 2011; Kim, Song, & Kang, 2018). In addition, shadows and uneven light conditions play a role in colour recognition, although these are not crucial if the surface area of the colour is small. To overcome these challenges, many previous studies have used devices that make the environment more controllable (e.g. Rateni, Dario, and Cavallo (2017)). In this research, the focus was on using devices in everyday life situations without external help.

The general findings of the research show that different methods are suitable for different purposes. To some extent, each of them is a powerful tool for colour recognition. This means that if the limits of the method are known, it can be deployed in

solving suitable use cases.

## 6.1 Mathematical methods in colour comparison

The recognition or matching of colours through mathematical methods works well in digital and controlled environments. For example, when the colour interpretation of two displays are compared, or in the colour calibration of displays and cameras. Mathematical methods are also an option in use cases where the capturing environment can be controlled. But in real-life scenarios, mathematical methods have a more limited usability due to a non-controlled environment and the variety between captured images.

The colour difference algorithm is a powerful tool as it has evolved over the years and become more and more accurate for various purposes. Furthermore, its challenges, such as discontinuity and applicability, are well known (Luo et al., 2001). CIEDE2000 can be easily integrated into software solutions and does not require special libraries or such.

CIEDE2000 has become a standard in dental colour-matching use cases, where the colours of the dental ceramics are observed in very controlled environments. (Yerliyurt & Sarıkaya, 2022). CIEDE2000 is also a good choice if the colour difference is measured against how humans perceive colours. The smallest difference that humans can recognise is typically Delta-E 2.0 or greater, but the acceptable difference depends on the use case (Farrag, Bakry, & Aly, 2022; Štruncová et al., 2020; Yılmaz, Tutus, & Sönmez, 2022). In Article I Delta-E 2.0 was used to recognise the smallest possible difference for humans.

The most important part of using the CIEDE2000 algorithm is choosing the parametric values, $K_L$, $K_C$, and $K_H$. These parameters adjust the weighting of the lightness ($K_L$), chroma ($K_C$), and hue ($K_H$) components of the colour difference, respectively, based on specific viewing conditions or application needs. A common approach is to have all of these parameters have a value of 1.0, although other values, such as $K_L = 2$, $K_C = 1$ , $K_H = 1$, and as $K_L = 2.76$, $K_C = 1.58$, $K_H = 1$ have been used in the past (e.g. del Mar Perez et al. (2011); He et al. (2022); Isohanni (2022); Mangine et al. (2005); Pereira et al. (2019)). Article I used previously mentioned parametric values in its experiments. The results show that in the case of the recognition of colour difference, using more weight on lightness and chroma provides better results. This seems reasonable as the actual colour is not important, but rather the change in colour. When these parametric values are used, the change from white to any colour other than yellow can be recognised more accurately. Using larger $K_L$ and $K_C$ values has also been considered as a solution by del Mar Perez et al. (2011) and Pecho, Ghinea, Alessandretti, Pérez, and Della Bona (2016). In

another study, Pérez et al. (2022) also showed that human vision is more sensitive to changes in lightness and chroma than hue.

When varying conditions are considered, the smallest colour difference that can be reliably recognised with the CIEDE2000 algorithm is 40% as the change in intensity of a CMYK channel(s) (Figure 31). With this change between colours, the CIEDE2000 difference is constantly over the threshold. If conditions are more controlled in the sense of printing, ambient light, and devices used, then smaller differences (20%) can be recognised, or if the use case does not require 100% accuracy at all times.

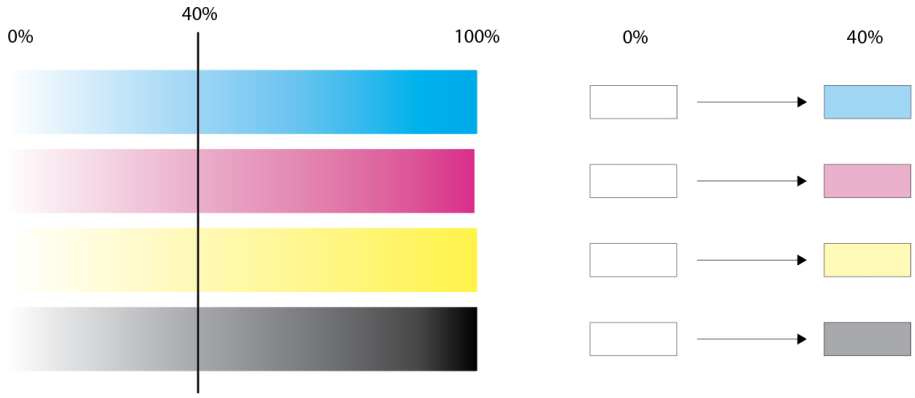



**Figure 31.** 40% colour intensity change.

If the change is observed with the CIEDE2000 algorithm, the solution should be transferable to varying conditions, and devices, if the difference between colours is high enough. This is because the absolute difference between colours is measured. However, the result depends on how well the colour information can be captured from the source, if the colour information is noise-free and how well the pre-processing works.

The CIEDE2000 algorithm can quantify how ”far apart” two colours are in perceptual terms. However, colour recognition requires identifying or naming a colour (e.g., ”red,” ”blue”), if CIEDE2000 is used for such purposes, reference colour(s) would be needed.

## 6.2 Clustering colours with unsupervised learning

Unsupervised clustering is a good tool when large datasets need to be processed. With unsupervised clustering, it is possible to get insight into the data when labels or relationships between datapoints are unknown. In computer vision/image processing, unsupervised clustering has use cases like data compression, image classi-

fication, and segmentation. As with mathematical methods, unsupervised clustering is quite easy to implement as part of software solutions, and this makes it feasible for many use cases.

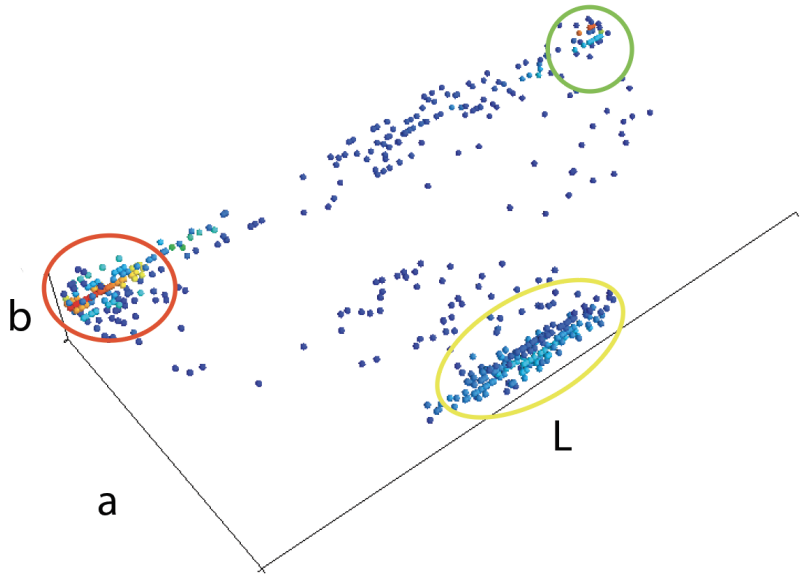



**Figure 32.** Example of the challenge of clustering colours.

When unsupervised clustering is used for colour difference recognition, the challenges can be seen in Figure 32, where the datapoints are scattered around the plot. Some dense regions (red circle = black, green circle = white, and yellow circle = CMYK(0.0, 0.6, 0.0, 0.0)) are highlighted in the figure. The black and white colours lie on different ends of the L-axis, and the yellow colour is fairly in the middle of the L-axis, with a negative A value and a positive B value. As the differences between these clusters become smaller, the algorithm cannot be sure which cluster the datapoint belongs to. From the figure, it is also possible to observe the existence of outliers: in the illustrated case outliers are between black and white colour, but also between white-yellow and black-yellow.

The research done in Article II shows that multiple different clustering approaches outperform mathematical methods in subtle colour-difference recognition. The algorithms K-means, C-means, GMM, Hierarchical clustering, and Spectral clustering all have the ability to accurately recognise colours when the difference in the CMYK intensity of the colour is 20% or greater.

In Article II, unsupervised clustering was used to identify if the colour differs from paper-white and its actual colour. And, for example, GMM was able to correctly identify the difference with 99.0 accuracy, when the intensity of the colour was equal to or greater than 20%. When the difference between colours dropped to 10%, the popular K-means algorithm and Spectral Clustering, with affinity = rbf, showed

the best performance. The smallest difference, only 5%, was too challenging for all of the algorithms used. The Delta-E difference between the 10% CMYK intensity ranges from 8.8 to 17.0 depending on the colour, which is still a very clearly visible difference for human perception. The process where colours are first clustered with unsupervised learning methods and then the difference between cluster centres is calculated with the CIEDE2000 algorithm seems to work quite well. The K-means has been used by other researchers in the past to classify colour or segment images (e.g. Abdulateef, Ahmed, and Salman (2020); Saifullah (2020); Trivedi, Shukla, and Pandey (2022); T. Wu, Gu, Shao, Zhou, and Li (2021)). Conservatively, where the accuracy of the methods (K-means, C-means, GMM, Hierarchical and Spectral clustering) is close to 100%, it can be considered that unsupervised clustering can be used if the difference in colour intensity is 20% or more (Figure 33).

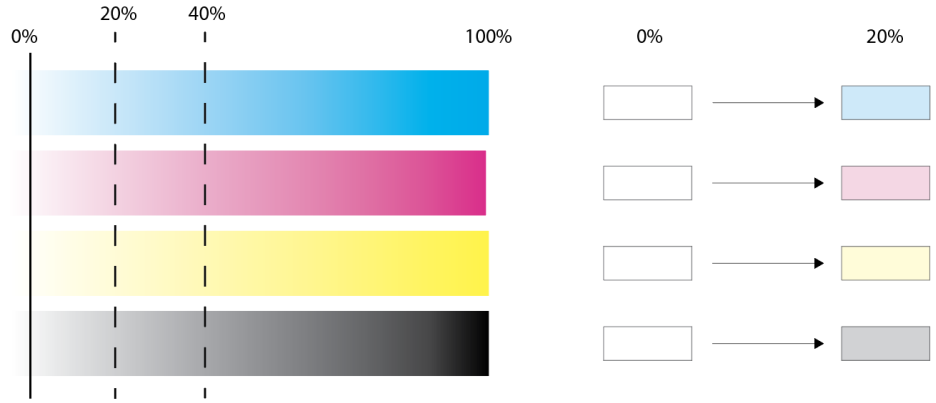



**Figure 33.** 20% colour intensity change.

As the best of the methods, K-means is very dependent on the initialisation of the cluster centres. The standard K-means algorithm randomly selects initial centroids, which can sometimes result in poor clustering results or slow convergence. The K-means++ initialisation method addresses this by carefully choosing initial centroids by spreading them out as much as possible. K-means++ has outperformed standard K-means in colour-related applications in research (e.g. Biswas, Umbaugh, Marino, and Sackman (2021)). One option would be the manual initialisation of cluster centres, as in the use case where black and white colour values are known. This could lead to better clustering results, as shown by Basar et al. (2020). As with the mathematical methods, the more advanced pre-processing might also help if it can eliminate noise and outliers. Using clustering algorithms that are resistant to outliers could also be experimented with, such as the recent innovation of Z. Wang (2020) regarding slope difference distribution (SDD) or Multi-View Clustering with Outlier Removal (MVCOR) by Chen, Wang, Hu, and Zheng (2020).

With unsupervised learning, the possibilities of transferring results into different contexts, such as different devices, printers, etc., should be feasible. As the unsupervised clustering only groups colours and the final measurement is done with the

CIEDE2000 algorithm, the results of the recognition is mostly dependent on how accurate clustering is in the used context. If clustering has reference or known values for black and white colour, then a calculation of difference to the third cluster will provide an absolute colour difference value. As with mathematical methods, the noise and paper used might again weaken the transfer of the results.

## 6.3 Neural network based colour classification

Unsupervised learning methods have already demonstrated the ability to recognise subtle colour differences with reasonable accuracy (Isohanni, 2024b). However, due to the inherent limitations in capturing non-linear relationships and contextual cues, more accurate solutions have been sought through supervised learning methods (Isohanni, 2025). In this work, the supervised approach involved classifying images into predefined categories, where each image represents a colour deviation from the paper-white average and is annotated accordingly.

Some supervised models were able to classify colours and recognise subtle differences more effectively than their unsupervised counterparts. As shown in Figure 34, the subtle colour changes become barely perceptible when the CMYK intensity varies by only 5%. As colour saturation decreases, the texture and structure of the paper surface become more pronounced, introducing additional noise into the image. It has been shown, for example by De and Pedersen, that the loss of colour information — even partial — makes the classification task more complex for CNNs (De & Pedersen, 2021).


**Figure 34.** Examples of small colour differences.

The increase in image noise highlights the importance of appropriate preprocessing. Improper preprocessing techniques can inadvertently degrade colour information or amplify noise artifacts (Maharana, Mondal, & Nemade, 2022). The noise artifacts become more significant when colour differences become smaller.

Among the tested models, only ResNet achieved classification accuracy exceeding 90% when recognising very small colour differences. The ResNet architecture has also been explored by De and Petersen, and by optimising its hyperparameters they

were able to make ResNet more robust against loss of colour information (De & Pedersen, 2021). The ResNet architectures have recently shown their robustness in recognising small differences, like example in Chen and Luo (2023); Gao, Yi, Liu, and Tan (2025); S. Wang, Wang, Yang, Li, and Fan (2022)

The core innovation of the ResNet architecture lies in its use of residual connections, which mitigate the vanishing gradient problem, a common issue in deeper convolutional neural networks. These skip connections allow gradient flow through identity mappings, which seems particularly advantageous in tasks where small colour differences must be preserved and propagated through the network. In the Article III, ResNet-34 achieved the best baseline performance among standard CNN architectures for subtle colour recognition.

The further tuning of the ResNet-34 architecture in Article IV proved that a custom architecture results in better accuracy. The customised version, where Gradient Centralization (GC) and max-pooling were used, resulted in the best accuracy even with very small colour differences (5%). This is a pretty good result, as the Delta-E difference between these colours can be as low as 4.4. In this research, ResNet-34 was used, but as the architectures evolve, other solutions might also be feasible. The Gradient Centralization has been previously used successfully in medical imagining like L. Zhang, Xia, Yang, Zhang, and Wang (2024) and Khatri and Kwon (2024). Max-pooling on the other hand can be used to identify small differences as seen in Zheng et al. (2021) and Ashtiani et al. (2023).

The Articles III and IV introduced a difference-image-based approach, where the input emphasizes the chromatic deviation from a reference white point. This approach not only improves class separability but also enhances the method's adaptability to other use cases with uniform backgrounds and subtle chromatic shifts. This solution remains sensitive to variations in paper structure and imaging conditions, which are features that CNNs may learn unless explicitly regularized.

The application of convolutional neural networks in complex colour classification tasks has recently demonstrated promising results. For instance, Wei et al. investigated the performance of CNN architectures VGG-16 and ResNet-34 in the classification of tea leaves, where colour was one of the distinguishing features despite the subtle differences in hue between samples (Wei et al., 2022). Both models achieved classification accuracies exceeding 90%, with VGG-16 outperforming ResNet-34 in overall performance. However, it is important to note that their classification approach relied not only on colour but also on additional morphological and textural features. Their findings also highlighted that incorporating all image channels contributed to improved model accuracy, underlining the value of comprehensive image data in CNN-based classification.

Similarly, Katkus, Maciulevičius, and Lipnickas (2023) explored the use of a CNN model for classifying amber gemstones based on subtle colour variations. Their cus-

tom CNN approach achieved high accuracy, demonstrating that CNNs are capable of distinguishing fine colour differences when appropriately trained and configured. The work of Katkus et al. aligns with the findings presented in this study, reinforcing the conclusion that CNNs are a powerful tool for colour classification. However, achieving high performance in such tasks typically requires tailored network architectures, careful preprocessing, and data augmentation strategies to mitigate the effects of noise and enhance generalisation.

One interesting option could be to use Multilayer Perceptrons (MLPs) instead of Convolutional Neural Networks. MLPs, by design, require the input image to be flattened into a one-dimensional vector, which removes all spatial relationships between pixels (Guo et al., 2022). This can be a limitation in image-based tasks where spatial patterns — such as edges, textures, or shapes — carry meaningful information, especially when the images are not consistently aligned in terms of position, rotation, or scale within the training set, or between the training and test sets. If the images contain more than just colour values, this loss of spatial context may degrade the performance of MLPs. In the presented use case, however, MLPs can still serve as a useful baseline for comparison with CNNs, and can help evaluate the significance of spatial information in the context of colour classification.

In a lightweight experiment, six different MLPs were trained on both datasets using ReLU activation, the Adam optimiser, and 50 training iterations. As high-resolution images lead to high-dimensional input vectors when flattened for MLPs, which significantly increases the number of model parameters. This makes the network more prone to overfitting (Liu, Starzyk, & Zhu, 2008). So in this experiment DS1 and DS2 were scaled to 64x64.

**Table 13.** Results of the different MLPs.

| MLP (layers) | Accuracy DS1 | Accuracy DS2 |
|---|---|---|
| (512, 256, 128, 64) | 0.87 | 0.88 |
| (512, 256, 128, 64, 32) | 0.86 | 0.87 |
| (256, 128, 64) | 0.76 | 0.87 |
| (256, 128, 64, 32) | 0.90 | 0.88 |
| (128, 64) | 0.72 | 0.85 |
| (128, 64, 32) | 0.80 | 0.86 |

The results (Table 13) of this comparison show that MLP networks almost reach the same level as the best CNNs (DenseNet, ResNet and VGG-16). MLPs computational cost is lower, so training can be done faster and also they are easier to implement in programming environment due their simplistic architecture. When larger colour differences are classified, MLPs can offer a good starting point.

## 6.4 Comparison of the methods

The approaches presented in Articles I-IV use different methods for colour recognition. Each of these methods has its use case and is suitable for colour difference recognition.

The challenge of the use case presented in this dissertation comes from the small data area, a subtle difference in colours, and the noise that is generated from the printing process, used paper, and imaging pipeline. Also, since the use case was aimed at consumer use, the environment of use cannot be controlled.

Based on the findings of the research, two main paths for integrating the results of the research can be identified a) unsupervised or b) supervised path. If more simple and easier to implement solutions are sought, then unsupervised clustering can be used to cluster colours into a specified number of clusters, and then the difference between colours can be calculated with the CIEDE2000 algorithm. With this approach, small differences (CMYK difference $<= 20\%$) can be reliably recognised. The results of using unsupervised learning will be better if references of white and black colour are available for use during clustering, and this solution can also be run directly on mobile devices without the need for data collection.

With supervised learning, it is possible to recognise very subtle colour differences (CMYK difference $<= 10\%$), but the solution will become more use case specific. The same model might not work if different papers and printers are used, and supervised learning will also require a lot of data collection, especially if more generalised solutions are sought. This could mean that a process to continuously train the model would need to be developed.

When printed sources are used, and the colour difference between paper-white and colour is observed with machine learning, unsupervised methods are useful if the difference between one or more CMYK channels is equal or greater than 20%. Supervised learning methods, CNNs and MLPs, can go lower and accurately recognise a 10% difference, or even 5% if the results do not need to have high accuracy all the time. There are differences between individual colours, and while the most challenging colour to be identified is yellow for all of the methods, other colours work quite equally in the presented use case.

In the tests conducted for the dissertation, both unsupervised and supervised methods were able to handle the recognition of colours that lie on the negative side of the A or B-axis on the LAB colour space, better than those on the positive side. This might mean that the used imaging pipelines give more weight to blue and green shades than to red and yellow shades.

## 6.5 Limitations and future research

Although consumer camera devices have developed over the years, the development has focused mainly on common photography, including taking selfies, vacation photos, etc. Some devices have special lenses for close-range photography, but they are not common. The presented research uses close-range photography to capture the colour data, and this approach is prone to noise that is produced in different phases of the imagining pipeline and which comes from different sources, like paper structure that appears easily. All of this has implications on the generalisation of the solution. Deviations related to camera devices can be overcome with image processing solutions that maintain colour difference information, like smart blurring or similar. But using different paper or printers can lead to situations where any generalisation of the solution is challenging. This dissertation has focused on using laser-jet printed sources with standard office paper. Different printers such as Flexo, gravure and inkjet result in different print quality. In addition, the paper that is used has an impact on the colour appearance. These limitations restrict the use of the proposed approach to specific datasets, where the image comes from the same printer with the same paper. However, these challenges can be overcome with larger datasets when supervised learning is used.

Both training and testing were conducted using images captured on the same hardware platform. This ensures consistency in image acquisition, it also limits the assessment of the model's generalisation across different devices. The used imaging scenes are relatively small and the colour areas are consistent, so variations between camera devices may have a limited impact on the results. Still, future work could explicitly evaluate the robustness of the approach by testing it on data captured with different devices exhibiting varying imaging characteristics.

More research is needed, especially in the preprocessing phase of the images. More advanced methods could adapt the images so that the differences between different environments, papers, and printers would have a less significant impact on the colour recognition, and thus make the solution more generic for various use cases. Also, this research only focused on small colour areas, so shadows and uneven light might have influence on how proposed approach works with larger areas.

Since the proposed solution can be deployed as a backend service, with results transmitted to mobile terminals or user-facing applications, inference latency is not critical at this stage. Algorithms like CIEDE2000 can be run in real-time in mobile terminals, as well as some unsupervised and supervised methods. But future research may explore how to optimise the presented models for reduced inference time, particularly if deployment in low-power or real-time environments.

The development of machine learning is so rapid at the moment that various architectures and approaches might be more usable to solve the problem in the future.

# 7 CONCLUSIONS

In this research, colour classification and recognition were studied by using various methods. The objective was to determine how colour, particularly subtle colour differences, could be classified by using colour difference algorithms, unsupervised and supervised methods. This objective was closely related to functional inks, which can serve as indicators if their colour or colour difference can be accurately identified.

The dissertation began with colour difference algorithms and demonstrated that the colour difference algorithm could identify colour differences in printed sources. But this was only if the colour difference is equal or greater than 20% in some or many CMYK channels, considering 0% as no printed colour and 100% as full intensity of colour. Although colour difference algorithms cannot be used for precise real-life use cases, they can indicate if the colour difference exceeds a certain threshold.

Unsupervised methods can classify colours when the colour difference is equal or less than 20% on some CMYK channel but greater than 10%. One of the best unsupervised algorithms is K-means with K-means++ initialisation, which is a hard clustering method that can achieve good accuracy even with a colour intensity difference as low as 10%, if the accuracy of 100% is not considered. Other unsupervised methods such as C-mean, GMM, and hierarchical clustering can classify the colour when the colour difference is over 10%.

In supervised learning, CNN architectures were shown to be more accurate than unsupervised algorithms. Architectures like AlexNet, DenseNet, and ResNet were able to correctly classify the colour, with ResNet standing out as the most accurate. ResNet addresses the problem of vanishing gradients by introducing ”skip connections” or ”residual connections,” allowing the network to bypass one or more layers by adding the input of a layer directly to the output of a subsequent layer, forming a residual block. These residual connections enable the model to retain critical features even as the network deepens.

The research modified the best CNN candidate, ResNet, and its ResNet-34 architecture to achieve more accurate classification results. When ResNet-34 was used with Gradient Centralisation and the last average pooling layer was changed to the max-pooling layer, the proposed architecture reached 98.28% accuracy with the very subtle colour differences dataset. This result was validated using K-Fold cross-validation, making it the best solution among all the experiments.

This dissertation shows that the classification of colour and the recognition of subtle colour differences is possible with colour difference algorithms, and unsupervised and supervised learning. Choosing optimal methods is use-case dependent. If the colour differences are small, CNNs are the best option. Unsupervised learning can

be quite easily taken into use without a training process and can recognise differences when they are at least 20% in CMYK intensity. If the difference is larger, colour difference algorithms can be used even under varying conditions.

# BIBLIOGRAPHY

Abdalla, A., Cen, H., El-manawy, A., & He, Y. (2019). Infield oilseed rape images segmentation via improved unsupervised learning models combined with supreme color features. *Computers and Electronics in Agriculture*, *162*, 1057–1068.

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(10), 1533-1545.

Abdulateef, S. K., Ahmed, S. R. A., & Salman, M. D. (2020). A novel food image segmentation based on homogeneity test of K-means clustering. In *IOP Conference Series: Materials Science and Engineering* (Vol. 928, p. 032059).

Adiwijaya, N. O., Romadhon, H. I., Putra, J. A., & Kuswanto, D. P. (2022). The quality of coffee bean classification system based on color by using k-nearest neighbor method. In *Journal of Physics: Conference Series* (Vol. 2157, p. 012034).

Ali, G. N., Mikkilineni, A. K., Delp, E. J., Allebach, J. P., Chiang, P.-J., & Chiu, G. T. (2004). Application of principal components analysis and gaussian mixture models to printer identification. In *NIP & Digital Fabrication Conference* (Vol. 20, pp. 301–305).

Al-Shakarji, N. M., Kassim, Y. M., & Palaniappan, K. (2017). Unsupervised learning method for plant and leaf segmentation. In *2017 IEEE applied imagery pattern recognition workshop (AIPR)* (pp. 1–4).

Amakdouf, H., Zouhri, A., El Mallahi, M., Tahiri, A., Chenouni, D., & Qjidaa, H. (2021). Artificial intelligent classification of biomedical color image using quaternion discrete radial tchebichef moments. *Multimedia Tools and Applications*, *80*, 3173–3192.

Anandhakrishnan, T., & Jaisakthi, S. (2022). Deep convolutional neural networks for image based tomato leaf disease detection. *Sustainable Chemistry and Pharmacy*, *30*, 100793.

Anderson, M., Motta, R., Chandrasekar, S., & Stokes, M. (1996). Proposal for a standard default color space for the internet—srgb. In *Color and imaging conference* (Vol. 4, pp. 238–245).

Angelico, R., Liccardo, D., Paoletti, M., Pietrobattista, A., Basso, M. S., Mosca, A., ... others (2021). A novel mobile phone application for infant stool color recogni-

tion: An easy and effective tool to identify acholic stools in newborns. *Journal of Medical Screening*, *28*(3), 230–237.

Anilkumar, KK and Manoj, VJ and Sagi, TM. (2018). Colour based image segmentation for automated detection of leukaemia: a comparison between CIELAB and CMYK colour spaces. In *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)* (pp. 1–6).

Apriyanti, D. H., Spreeuwers, L. J., Lucas, P. J., & Veldhuis, R. N. (2021). Automated color detection in orchids using color labels and deep learning. *PloS one*, *16*(10), e0259036.

Arora, J., Khatter, K., & Tushir, M. (2019). Fuzzy c-means clustering strategies: A review of distance measures. *Software Engineering: Proceedings of CSI 2015*, 153–162.

Arsenovic, M., Karanovic, M., Sladojevic, S., Anderla, A., & Stefanovic, D. (2019). Solving current limitations of deep learning based approaches for plant disease detection. *Symmetry*, *11*(7), 939.

Arunachalam, D., & Kumar, N. (2018). Benefit-based consumer segmentation and performance evaluation of clustering approaches: An evidence of data-driven decision-making. *Expert Systems with Applications*, *111*, 11–34.

Ashtiani, F., On, M. B., Sanchez-Jacome, D., Perez-Lopez, D., Yoo, S. B., & Blanco-Redondo, A. (2023). Photonic max-pooling for deep neural networks using a programmable photonic platform. In *Optical fiber communication conference* (pp. M1J–6).

Atha, D. J., & Jahanshahi, M. R. (2018). Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection. *Structural Health Monitoring*, *17*(5), 1110-1128.

Bagherinia, H., & Manduchi, R. (2011). A theory of color barcodes. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (p. 806-813).

Balaji, V., Suganthi, S., Rajadevi, R., Kumar, V. K., Balaji, B. S., & Pandiyan, S. (2020). Skin disease detection and segmentation using dynamic graph cut algorithm and classification through naive bayes classifier. *Measurement*, *163*, 107922.

Barbe, D. F. (1975). Imaging devices using the charge-coupled concept. *Proceedings of the IEEE*, *63*(1), 38–67.

Barton, S., Alakkari, S., O'Dwyer, K., Ward, T., & Hennelly, B. (2021). Convolution network with custom loss function for the denoising of low snr raman spectra. *Sensors*, *21*(14), 4623.

Basar, S., Ali, M., Ochoa-Ruiz, G., Zareei, M., Waheed, A., & Adnan, A. (2020). Unsupervised color image segmentation: A case of rgb histogram based k-means clustering initialization. *Plos one*, *15*(10), e0240015.

Bayer, B. E. (U.S. Patent 3971065A, Jul. 1976). *Color imaging array.*

Beaugnon, A., & Chifflier, P. (2018). Machine learning for computer security detection systems: practical feedback and solutions. *Proceedings of the 2018 Intelligence Artificielle et Cybersécurité/Artificial Intelligence and Cybersecurity (C&ESAR), Rennes, France*, 19–21.

Bejani, M. M., & Ghatee, M. (2021). A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, *54*(8), 6391–6438.

Belasco, R., Edwards, T., Munoz, A., Rayo, V., & Buono, M. J. (2020). The effect of hydration on urine color objectively evaluated in CIE L* a* b* color space. *Frontiers in Nutrition*, *7*, 576974.

Berns, R. S. (2019). *Billmeyer and saltzman's principles of color technology*. John Wiley & Sons.

Bigas, M., Cabruja, E., Forest, J., & Salvi, J. (2006). Review of cmos image sensors. *Microelectronics journal*, *37*(5), 433–451.

Bilgin, M., & Backhaus, J. (2017). Intelligent codes by controlled responsiveness to external stimuli. In *Printing future days 2017 7th international scientific conference on print and media technology* (pp. 85–90).

Bilius, L. B., & Pentiuc, S. G. (2020). Unsupervised clustering for hyperspectral images. *Symmetry*, *12*(2), 277.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4) (No. 4). Springer.

Biswas, H., Umbaugh, S. E., Marino, D., & Sackman, J. (2021). Comparison of K-means and K-means++ for image compression with thermographic images. In

*Thermosense: Thermal Infrared Applications XLIII* (Vol. 11743, pp. 209–214).

Bloss, R. (2009). Making better "eyes" for cameras, mobile (cell) phones and cars. *Assembly Automation*, *29*(1), 14–18.

Botalb, A., Moinuddin, M., Al-Saggaf, U., & Ali, S. S. (2018). Contrasting convolutional neural network (cnn) with multi-layer perceptron (mlp) for big data analysis. In *2018 international conference on intelligent and advanced system (icias)* (pp. 1–5).

Boulent, J., Foucher, S., Théau, J., & St-Charles, P.-L. (2019). Convolutional neural networks for the automatic identification of plant diseases. *Frontiers in plant science*, *10*, 941.

Brophy, E., Hennelly, B., De Vos, M., Boylan, G., & Ward, T. (2022). Improved electrode motion artefact denoising in ECG using convolutional neural networks and a custom loss function. *IEEE Access*, *10*, 54891–54898.

Büyükarıkan, B., & Ülker, E. (2022). Using convolutional neural network models illumination estimation according to light colors. *Optik*, *271*, 170058.

Cabello, J., Bailey, A., Kitchen, I., Prydderch, M., Clark, A., Turchetta, R., & Wells, K. (2007). Digital autoradiography using room temperature ccd and cmos imaging technology. *Physics in medicine & biology*, *52*(16), 4993.

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, *3*(1), 1–27.

Caruana, R. (1997). Multitask learning. *Machine learning*, *28*, 41–75.

Cauchy, A., et al. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, *25*(1847), 536–538.

Chen, C., & Luo, D. (2023). Enhanced resnet network for food image security recognition. In *Third international conference on optics and image processing (icoip 2023)* (Vol. 12747, pp. 490–493).

Chen, C., Wang, Y., Hu, W., & Zheng, Z. (2020). Robust multi-view k-means clustering with outlier removal. *Knowledge-Based Systems*, *210*, 106518.

Cho, J. D., Jeong, J., Kim, J. H., & Lee, H. (2020). Sound coding color to improve artwork appreciation by people with visual impairments. *Electronics*, *9*(11), 1981.

Craven, P., & Wahba, G. (1978). Smoothing noisy data with spline functions:

estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische mathematik*, *31*(4), 377–403.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*(2), 224–227.

De, K., & Pedersen, M. (2021). Impact of colour on robustness of deep neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 21–30).

de Brito Silva, G. V., & Flores, F. C. (2021). Rot corn grain classication by color and texture analysis. *IEEE Latin America Transactions*, *20*(2), 208–214.

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for machine learning*. Cambridge University Press.

del Mar Perez, M., Ghinea, R., Herrera, L. J., Ionescu, A. M., Pomares, H., Pulgar, R., & Paravina, R. D. (2011). Dental ceramics: a ciede2000 acceptability thresholds for lightness, chroma and hue differences. *Journal of Dentistry*, *39*, e37–e44.

Di Gennaro, S. F., Toscano, P., Cinat, P., Berton, A., & Matese, A. (2019). A low-cost and unsupervised image recognition methodology for yield estimation in a vineyard. *Frontiers in plant science*, *10*, 559.

Doolittle, M. H., Doolittle, K. W., Winkelman, Z., & Weinberg, D. S. (1997, January). Color images in telepathology: how many colors do we need? *Human Pathology*, *28*(1), 36–41.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(7).

Dunn, J. C. (1973). *A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters*. Taylor & Francis.

Emmert-Streib, F., & Dehmer, M. (2019). Evaluation of regression models: Model assessment, model selection and generalization error. *Machine learning and knowledge extraction*, *1*(1), 521–551.

Engilberge, M., Collins, E., & Süsstrunk, S. (2017). Color representation in deep neural networks. In *2017 IEEE International Conference on Image Processing (ICIP)* (pp. 2786–2790).

Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., & Akinyelu, A. A. (2022). A comprehensive survey of clustering algo-

rithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, *110*, 104743.

Fan, Y., Li, J., Guo, Y., Xie, L., & Zhang, G. (2021). Digital image colorimetry on smartphone for chemical analysis: A review. *Measurement*, *171*, 108829.

Farrag, K. M., Bakry, S. I., & Aly, Y. M. (2022). Effect of yellow anodization of titanium on the shade of lithium disilicate ceramic with different thicknesses. *The Journal of Prosthetic Dentistry*, *128*(4), 793–e1.

Fiesler, E., & Beale, R. (2020). *Handbook of neural computation*. CRC Press.

Ford, A., & Roberts, A. (1998). *Colour space conversions* (Tech. Rep.). Westminster University, London.

Fowlkes, E. B., & Mallows, C. L. (1983). A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, *78*(383), 553–569.

Fu, C., & Yang, J. (2021). Granular classification for imbalanced datasets: a minkowski distance-based method. *Algorithms*, *14*(2), 54.

Fuentes-Peñailillo, F., Ortega-Farias, S., Rivera, M., Bardeen, M., & Moreno, M. (2018). Using clustering algorithms to segment UAV-based RGB images. In *2018 IEEE international conference on automation/XXIII congress of the Chilean association of automatic control (ICA-ACCA)* (pp. 1–5).

Gao, X., Yi, J., Liu, L., & Tan, L. (2025). A generic image steganography recognition scheme with big data matching and an improved ResNet50 deep learning network. *Electronics*, *14*(8), 1610.

Gençtav, A., Aksoy, S., & Önder, S. (2012). Unsupervised segmentation and classification of cervical cell images. *Pattern recognition*, *45*(12), 4151–4168.

Gere, A. (2023). Recommendations for validating hierarchical clustering in consumer sensory projects. *Current Research in Food Science*, *6*, 100522.

Ghinea, R., Herrera, L., Ionescu, A., Pomares, H., Pulgar, R., Paravina, R., et al. (2011). Dental ceramics: a ciede2000 acceptability thresholds for lightness, chroma and hue differences. *Journal of Dentistry*, *39*, e37–44.

Ghinea, R., Pérez, M. M., Herrera, L. J., Rivas, M. J., Yebra, A., & Paravina, R. D. (2010). Color difference thresholds in dental ceramics. *Journal of dentistry*, *38*, e57–e64.

Gligoric, N., Krco, S., Hakola, L., Vehmas, K., De, S., Moessner, K., ... Van Kranenburg, R. (2019). Smarttags: Iot product passport for circular economy based on printed sensors and unique item-level identifiers. *Sensors*, *19*(3), 586.

Golan, E. H., Krissoff, B., Kuchler, F., Calvin, L., Nelson, K. E., & Price, G. K. (2004). Traceability in the us food supply: economic theory and industry studies.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Govender, P., & Sivakumar, V. (2020). Application of k-means and hierarchical clustering techniques for analysis of air pollution: A review (1980–2019). *Atmospheric pollution research*, *11*(1), 40–56.

*Graphic technology and photography — colour characterisation of digital still cameras (dscs)* (Vol. 2; Standard). (2012, November). Geneva, CH: International Organization for Standardization.

*Graphic technology — colour and transparency of printing ink sets for four-colour printing* (Vol. 3; Standard). (2017, August). Geneva, CH: International Organization for Standardization.

*Graphic technology — process control for the production of half-tone colour separations, proof and production prints* (Vol. 3; Standard). (2013, December). Geneva, CH: International Organization for Standardization.

Green, P., & MacDonald, L. (2011). *Colour engineering: achieving device independent colour*. John Wiley & Sons.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... others (2018). Recent advances in convolutional neural networks. *Pattern recognition*, *77*, 354–377.

Gu, Q., Zhu, L., & Cai, Z. (2009). Evaluation measures of the classification performance of imbalanced data sets. In *Computational Intelligence and Intelligent Systems: 4th International Symposium, ISICA 2009, Huangshi, China, October 23-25, 2009. Proceedings 4* (pp. 461–471).

Gunturk, B. K., Glotzbach, J., Altunbasak, Y., Schafer, R. W., & Mersereau, R. M. (2005). Demosaicking: color filter array interpolation. *IEEE Signal processing magazine*, *22*(1), 44–54.

Guo, J., Tang, Y., Han, K., Chen, X., Wu, H., Xu, C., ... Wang, Y. (2022). Hiremlp: Vision mlp via hierarchical rearrangement. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 826–836).

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell system technical journal*, *29*(2), 147–160.

Han, J., Kamber, M., & Pei, J. (2012a). 10 - cluster analysis: Basic concepts and methods. In J. Han, M. Kamber, & J. Pei (Eds.), *Data mining (third edition)* (Third Edition ed., p. 443-495). Boston: Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-12-381479-1.00010-1

Han, J., Kamber, M., & Pei, J. (2012b). 2 - getting to know your data. In J. Han, M. Kamber, & J. Pei (Eds.), *Data Mining (Third Edition)* (Third Edition ed., p. 39-82). Boston: Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-12-381479-1.00002-2

Hartigan, J. A., & Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, *28*(1), 100–108.

Harvey, J. (2007, 11). Mechanical engineers' handbook: Materials and mechanical design, volume 1, third edition. In (p. 1423 - 1436). John Wiley & Sons.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2). Springer.

Haykin, S. (2009). *Neural networks and learning machines, 3/e*. Pearson Education India.

He, R., Xiao, K., Pointer, M., Melgosa, M., & Bressler, Y. (2022). Optimizing parametric factors in cielab and ciede2000 color-difference formulas for 3d-printed spherical objects. *Materials*, *15*(12), 4055.

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception* (pp. 65–93). Elsevier.

Hensel, M., Scheiermann, M., Fahrer, J., & Durner, D. (2023). New insights into wine color analysis: A comparison of analytical methods to sensory perception for red and white varietal wines. *Journal of Agricultural and Food Chemistry*, *72*(4), 2008–2017.

Hinton, G. E., & Roweis, S. (2002). Stochastic neighbor embedding. *Advances in neural information processing systems*, *15*.

Hirsch, R. (2022). *Light and lens: Thinking about photography in the digital age*. Routledge.

Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, *22*, 85–126.

Höge, M., Wöhling, T., & Nowak, W. (2018). A primer for model selection: The decisive role of model complexity. *Water Resources Research*, *54*(3), 1688–1715.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, *24*(6), 417.

Hristev, R. (1998). *The ANN book*.

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution* (pp. 492–518). Springer.

Isohanni, J. (2022). Use of functional ink in a smart tag for fast-moving consumer goods industry. *Journal of Packaging Technology and Research*, *6*(3), 187–198.

Isohanni, J. (2023, March). *Qr-code dataset, with colour embed insed.* Zenodo. https://doi.org/10.5281/zenodo.7749912

Isohanni, J. (2024a, April). *Qr-codes with colour embed inside.* Zenodo. https://doi.org/10.5281/zenodo.11079897

Isohanni, J. (2024b). Recognising small colour changes with unsupervised learning, comparison of methods. *Advances in Computational Intelligence*, *4*(2), 6.

Isohanni, J. (2025). Customised resnet architecture for subtle color classification. *International Journal of Computers and Applications*, *47*(4), 341–355.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, *31*(8), 651–666.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, *31*(3), 264–323.

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *Statistical learning*. Springer.

Japkowicz, N., & Shah, M. (2011). *Evaluating learning algorithms: a classification perspective*. Cambridge University Press.

Jeon, Y., Yoo, J., Lee, J., & Yoon, S. (2017). Nc-link: A new linkage method for efficient hierarchical clustering of large-scale data. *IEEE Access*, *5*, 5594-5608.

Jiang, H., Tian, Q., Farrell, J., & Wandell, B. A. (2017). Learning the image processing pipeline. *IEEE Transactions on Image Processing*, *26*(10), 5032–5042.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, *32*(3), 241–254.

Katkus, D., Maciulevičius, P., & Lipnickas, A. (2023). Amber gemstones colour classification by cnn. In *2023 ieee 12th international conference on intelligent data acquisition and advanced computing systems: Technology and applications (idaacs)* (Vol. 1, pp. 531–536).

Keivani, M., Mazloum, J., Sedaghatfar, E., & Tavakoli, M. B. (2020). Automated analysis of leaf shape, texture, and color features for plant classification. *TRAITEMENT du Signal*, *37*(1), 17–28.

Khan, A. R., Khan, S., Harouni, M., Abbasi, R., Iqbal, S., & Mehmood, Z. (2021). Brain tumor segmentation using k-means clustering and deep learning with synthetic data augmentation for classification. *Microscopy Research and Technique*, *84*(7), 1389–1399.

Khatri, U., & Kwon, G.-R. (2024). Diagnosis of alzheimer's disease via optimized lightweight convolution-attention and structural mri. *Computers in Biology and Medicine*, *171*, 108116.

Kim, M., Song, K., & Kang, M. (2018, 04). No-reference contrast measurement for color images based on visual stimulus. *IEEE Access*, *PP*, 1-1.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kreer, J. (1957). A question of terminology. *IRE Transactions on Information Theory*, *3*(3), 208-208.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, *22*(1), 79–86.

Kumar, A., Singh, C., & Sachan, M. K. (2024). A novel cross correlation-based color texture descriptor for the classification of breast cancer histopathology images. *Biomedical Signal Processing and Control*, *93*, 106157.

Lai, P., & Westland, S. (2020). Machine learning for colour palette extraction from

fashion runway images. *International Journal of Fashion Design, Technology and Education*, *13*(3), 334-340.

Lance, G. N., & Williams, W. T. (1967). A general theory of classificatory sorting strategies: 1. hierarchical systems. *The computer journal*, *9*(4), 373–380.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, *2*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y., Touresky, D., Hinton, G., & Sejnowski, T. (1988). A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school* (Vol. 1, pp. 21–28).

Lee, C.-H., Lee, E.-J., Ahn, S.-C., & Ha, Y.-H. (2001). Color space conversion via gamut-based color samples of printer. *Journal of Imaging Science and Technology*, *45*(5), 427–435.

Lee, M.-K., Golzarian, M. R., & Kim, I. (2021). A new color index for vegetation segmentation and classification. *Precision Agriculture*, *22*, 179–204.

Limare, N., Lisani, J.-L., Morel, J.-M., Petro, A. B., & Sbert, C. (2011). Simplest color balance. *Image Processing On Line*, *1*, 297–315.

Litwiller, D. (2001). Ccd vs. cmos. *Photonics spectra*, *35*(1), 154–158.

Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J., & Wu, S. (2013). Understanding and enhancement of internal clustering validation measures. *IEEE transactions on cybernetics*, *43*(3), 982–994.

Liu, Y., Starzyk, J. A., & Zhu, Z. (2008). Optimized approximation algorithm in neural networks without overfitting. *IEEE transactions on neural networks*, *19*(6), 983–995.

López, A., Guzmán, G. A., & Di Sarli, A. R. (2016). Color stability in mortars and concretes. Part 1: Study on architectural mortars. *Construction and Building Materials*, *120*, 617–622. https://doi.org/https://doi.org/10.1016/j.conbuildmat.2016.05.133

Lukas, J., Fridrich, J., & Goljan, M. (2006). Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security*,

*1*(2), 205–214.

Luo, M. R., Cui, G., & Rigg, B. (2001). The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application*, *26*(5), 340–350.

Magnan, P. (2003). Detection of visible photons in ccd and cmos: A comparative view. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *504*(1-3), 199–212.

Mahalanobis, P. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, *2*(1), 49–55.

Maharana, K., Mondal, S., & Nemade, B. (2022). A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, *3*(1), 91–99.

Maiti, A., Chatterjee, B., & Santosh, K. (2021). Skin cancer classification through quantized color features and generative adversarial network. *International Journal of Ambient Computing and Intelligence (IJACI)*, *12*(3), 75–97.

Mangin, P., & Silvy, J. (1997). Fundamental studies of linting: Understanding ink-press-paper interactions non-linearity. In *Taga* (pp. 884–905).

Mangine, H., Jakes, K., & Noel, C. (2005). A preliminary comparison of cie color differences to textile color acceptability using average observers. *Color Research & Application*, *30*(4), 288–294.

Martínez-Domingo, M. Á., López-Baldomero, A. B., Tejada-Casado, M., Melgosa, M., & Collado-Montero, F. J. (2024). Colorimetric evaluation of a reintegration via spectral imaging—case study: Nasrid tiling panel from the alhambra of granada (spain). *Sensors*, *24*(12), 3872.

Mimmack, G. M., Mason, S. J., & Galpin, J. S. (2001). Choice of distance matrices in cluster analysis: Defining regions. *Journal of climate*, *14*(12), 2790–2797.

Minkowski, H. (1910). *Geometrie der zahlen* (Vol. 1). BG Teubner.

Mokrzycki, W. S., & Tatol, M. (2011, April). Colour difference delta-e - a survey. *MG&V*, *20*(4), 383–411.

Moreira, G., Magalhães, S. A., Pinho, T., dos Santos, F. N., & Cunha, M. (2022). Benchmark of deep learning and a proposed hsv colour space models for the detection and classification of greenhouse tomato. *Agronomy*, *12*(2), 356.

Morovic, J., & Luo, M. R. (2001). The fundamentals of gamut mapping: A survey.

*Journal of Imaging Science and Technology*, *45*(3), 283–290.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 807–814).

Nakamura, J. (2017). *Image sensors and signal processing for digital still cameras*. CRC press.

Nalhiati, G., Borges, G. G., Sperança, M. A., & Pereira, F. M. V. (2023). Color classification for red alcohol vinegar to control the quality of the end-product. *Food Analytical Methods*, *16*(7), 1283–1290.

Naveed, K., Ehsan, S., McDonald-Maier, K. D., & Ur Rehman, N. (2019). A multiscale denoising framework using detection theory with application to images from cmos/ccd sensors. *Sensors*, *19*(1), 206.

Nguyen, C.-N., Vo, V.-T., & Ha, N. C. (2022). Developing a computer vision system for real-time color measurement–a case study with color characterization of roasted rice. *Journal of Food Engineering*, *316*, 110821.

Ni, J., Yan, Z., & Jiang, J. (2022). Tonguecaps: An improved capsule network model for multi-classification of tongue color. *Diagnostics*, *12*(3), 653.

Niu, S., Liu, Y., Wang, J., & Song, H. (2020). A decade survey of transfer learning (2010–2020). *IEEE Transactions on Artificial Intelligence*, *1*(2), 151–166.

Nixon, M., Outlaw, F., & Leung, T. S. (2020). Accurate device-independent colorimetric measurements using smartphones. *PLoS One*, *15*(3), e0230561.

*Nokia n9 product page.* (n.d.). https://www.hmd.com/en_int/nokia-9-pureview ?sku=11AOPLW1A08. (Accessed: 2024-05-05)

Nugroho, H. A., Goratama, R. D., & Frannita, E. L. (2021). Face recognition in four types of colour space: a performance analysis. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1088, p. 012010).

Othman, N., Zain, M. Z. M., Ishak, I. S., Bakar, A. R. A., Ab Wahid, M., & Mohamad, M. (2020). A colour recognition device for the visually disabled people. *Indonesian Journal of Electrical Engineering and Computer Science*, *17*(3), 1322–1329.

Pak, A., Reichel, S., & Burke, J. (2022). Machine-learning-inspired workflow for camera calibration. *Sensors*, *22*(18), 6804.

Palus, H. (1998). Representations of colour images in different colour spaces. In *The colour image processing handbook* (pp. 67–90). Springer.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, *2*(11), 559–572.

Pecho, O. E., Ghinea, R., Alessandretti, R., Pérez, M., & Della Bona, A. (2016). Visual and instrumental shade matching using CIELAB and CIEDE2000 color difference formulas. *Dental Materials*, *32*(1), 82–92. https://doi.org/https://doi.org/10.1016/j.dental.2015.10.015

Pecho, O. E., Ghinea, R., Alessandretti, R., Pérez, M. M., & Della Bona, A. (2016). Visual and instrumental shade matching using CIELAB and CIEDE2000 color difference formulas. *Dental materials*, *32*(1), 82–92.

Pennebaker, W. B., & Mitchell, J. L. (1992). *Jpeg: Still image data compression standard*. Springer Science & Business Media.

Pereira, A., Carvalho, P., Coelho, G., & Côrte-Real, L. (2019). Efficient CIEDE2000-based color similarity decision for computer vision. *IEEE Transactions on Circuits and Systems for Video Technology*, *30*(7), 2141–2154.

Pérez, M. M., Carrillo-Perez, F., Tejada-Casado, M., Ruiz-López, J., Benavides-Reyes, C., & Herrera, L. J. (2022). CIEDE2000 lightness, chroma and hue human gingiva thresholds. *Journal of Dentistry*, *124*, 104213.

Plataniotis, K. N. (2001). Color image processing and applications. *Measurement Science and Technology*, *12*(2), 222–222.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, *4*(5), 1–17.

Pramoditha, R. (n.d.). *Overview of a neural network's learning process.* https://medium.com/data-science-365/overview-of-a-neural-networks-learning-process-61690a502fa. (Accessed: 2024-07-01)

Prechelt, L. (2002). Early stopping-but when? In *Neural networks: Tricks of the trade* (pp. 55–69). Springer.

Przybyło, J., & Jabłoński, M. (2019). Using deep convolutional neural network for oak acorn viability recognition based on color images of their sections. *Computers and Electronics in Agriculture*, *156*, 490-499.

Rafiq, M., Bugmann, G., & Easterbrook, D. (2001). Neural network design for engineering applications. *Computers & Structures*, *79*(17), 1541-1552. https://doi.org/https://doi.org/10.1016/S0045-7949(01)00039-6

Ramanath, R., Snyder, W. E., Yoo, Y., & Drew, M. S. (2005). Color image processing pipeline. *IEEE Signal processing magazine*, *22*(1), 34–43.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, *66*(336), 846–850.

Rao, K. R., & Yip, P. C. (2018). *The transform and data compression handbook*. CRC press.

Rateni, G., Dario, P., & Cavallo, F. (2017). Smartphone-based food diagnostic technologies: A review. *Sensors*, *17*(6), 1453.

Rendón, E., Abundez, I. M., Gutierrez, C., Zagal, S. D., Arizmendi, A., Quiroz, E. M., & Arzate, H. E. (2011). A comparison of internal and external cluster validation indexes. In *Proceedings of the 2011 American Conference, San Francisco, CA, USA* (Vol. 29, pp. 1–10).

Rennie, J. D., & Srebro, N. (2005). Loss functions for preference levels: Regression with discrete ordered labels. In *Proceedings of the IJCAI multidisciplinary workshop on advances in preference handling* (Vol. 1).

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.

Rubinstein, R. Y., & Kroese, D. P. (2004). *The cross-entropy method: a unified approach to combinatorial optimization, monte-carlo simulation, and machine learning* (Vol. 133). Springer.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Rundo, L., Beer, L., Ursprung, S., Martin-Gonzalez, P., Markowetz, F., Brenton, J. D., ... Woitek, R. (2020). Tissue-specific and interpretable sub-segmentation of whole tumour burden on CT images by unsupervised fuzzy clustering. *Computers in biology and medicine*, *120*, 103751.

Saifullah, S. (2020). Segmentation for embryonated egg images detection using the k-means algorithm in image processing. In *2020 Fifth International Conference on Informatics and Computing (ICIC)* (pp. 1–7).

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210–229.

Samuel, A. L. (1967). Some studies in machine learning using the game of checkers. ii—recent progress. *IBM Journal of research and development*, *11*(6), 601–617.

Schäfer, E., Heiskanen, J., Heikinheimo, V., & Pellikka, P. (2016). Mapping tree species diversity of a tropical montane forest by unsupervised clustering of airborne imaging spectroscopy data. *Ecological indicators*, *64*, 49–58.

Schöberl, M., Senel, C., Fößel, S., Bloss, H., & Kaup, A. (2009). Non-linear dark current fixed pattern noise compensation for variable frame rate moving picture cameras. In *2009 17th European Signal Processing Conference* (pp. 268–272).

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Sharma, S. (1995). *Applied multivariate techniques*. John Wiley & Sons, Inc.

Shrivastava, V. K., & Pradhan, M. K. (2021). Rice plant disease classification using color features: a machine learning paradigm. *Journal of Plant Pathology*, *103*(1), 17–26.

Sietsma, & Dow. (1988). Neural net pruning-why and how. In *IEEE 1988 international conference on neural networks* (pp. 325–333).

Sinaga, K. P., & Yang, M.-S. (2020). Unsupervised k-means clustering algorithm. *IEEE access*, *8*, 80716–80727.

Skandarajah, A., Reber, C. D., Switz, N. A., & Fletcher, D. A. (2014). Quantitative imaging with a mobile phone microscope. *PloS one*, *9*(5), e96906.

Sokal, R., Michener, C., & of Kansas, U. (1958). *A statistical method for evaluating systematic relationships*. University of Kansas.

Souchleris, K., Sidiropoulos, G. K., & Papakostas, G. A. (2023). Reinforcement learning in game industry—review, prospects and challenges. *Applied Sciences*, *13*(4), 2443.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The*

*journal of machine learning research*, *15*(1), 1929–1958.

Steinbach, M., Ertöz, L., & Kumar, V. (2004). The challenges of clustering high dimensional data. In *New directions in statistical physics: econophysics, bioinformatics, and pattern recognition* (pp. 273–309). Springer.

Štruncová, M., Toma, S. H., Araki, K., Bresciani, E., Rodrigues, F. P., Medeiros, I. S., & Dutra-Correa, M. (2020). Silver nanoparticles added to a commercial adhesive primer: Colour change and resin colour stability with ageing. *International Journal of Adhesion and Adhesives*, *102*, 102694.

Su, R., Guo, Y., Wu, C., Jin, Q., & Zeng, T. (2024). Kernel correlation–dissimilarity for Multiple Kernel k-Means clustering. *Pattern Recognition*, *150*, 110307.

Su, X., Yue, X., Kong, M., Xie, Z., Yan, J., Ma, W., ... Liu, M. (2023). Leaf color classification and expression analysis of photosynthesis-related genes in inbred lines of chinese cabbage displaying minor variations in dark-green leaves. *Plants*, *12*(11), 2124.

Su, Z., Yang, J., Li, P., Jing, J., & Zhang, H. (2022). A precise method of color space conversion in the digital printing process based on pso-dbn. *Textile Research Journal*, *92*(9-10), 1673–1681.

Sun, X., Panda, R., Feris, R., & Saenko, K. (2020). Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, *33*, 8728–8740.

Szeliski, R. (2022). *Computer vision: algorithms and applications*. Springer Nature.

Takase, T., Oyama, S., & Kurihara, M. (2018). Effective neural network training with adaptive learning rate based on training loss. *Neural Networks*, *101*, 68–78.

Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105–6114).

Tang, K., Astola, J., & Neuvo, Y. (1994). Multichannel edge enhancement in color image processing. *IEEE Transactions on circuits and systems for video technology*, *4*(5), 468–479.

Tejada-Casado, M., Pérez, M. M., Della Bona, A., Lübbe, H., Ghinea, R., & Herrera, L. J. (2024). Chroma-dependence of CIEDE2000 acceptability thresholds for dentistry. *Journal of Esthetic and Restorative Dentistry*, *36*(3), 469–476.

Terensan, S., Salgadoe, A. S. A., Kottearachchi, N. S., & Weerasena, O. J. (2024). Proximally sensed rgb images and colour indices for distinguishing rice blast and brown spot diseases by k-means clustering: Towards a mobile application solution. *Smart Agricultural Technology*, 100532.

Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika*, *18*(4), 267–276.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *58*(1), 267–288.

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(2), 411–423.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, *4*(2), 26–31.

Tooms, M. S. (2015). *Colour reproduction in electronic imaging systems: photography, television, cinematography*. John Wiley & Sons.

Trivedi, V. K., Shukla, P. K., & Pandey, A. (2022). Automatic segmentation of plant leaves disease using min-max hue histogram and k-mean clustering. *Multimedia Tools and Applications*, *81*(14), 20201–20228.

Ultsch, A., & Lötsch, J. (2022). Euclidean distance-optimized data transformation for cluster analysis in biomedical data (edotrans). *BMC bioinformatics*, *23*(1), 233.

Valkenborg, D., Rousseau, A.-J., Geubbelmans, M., & Burzykowski, T. (2023). Unsupervised learning. *American Journal of Orthodontics and Dentofacial Orthopedics*, *163*(6), 877–882.

Wang, D., Wang, X., Chen, Y., Wu, Y., & Zhang, X. (2023). Strawberry ripeness classification method in facility environment based on red color ratio of fruit rind. *Computers and Electronics in Agriculture*, *214*, 108313.

Wang, H., Yu, W., You, J., Ma, R., Wang, W., & Li, B. (2021). A unified framework for anomaly detection of satellite images based on well-designed features and an artificial neural network. *Remote Sensing*, *13*(8), 1506.

Wang, S., Wang, K., Yang, T., Li, Y., & Fan, D. (2022). Improved 3D-ResNet sign language recognition algorithm with enhanced hand features. *Scientific Reports*, *12*(1), 17812.

Wang, X., & Zhang, D. (2010). An optimized tongue image color correction scheme. *IEEE Transactions on information technology in biomedicine*, *14*(6), 1355–1364.

Wang, X., Zhang, J., Jiang, Y., Du, J., Miao, D., & Xu, C. (2024). Color difference of yarn-dyed fabrics woven from warp and weft yarns in different color depths. *Pigment & Resin Technology*, *53*(1), 28–35.

Wang, Z. (2020). Robust segmentation of the colour image by fusing the SDD clustering results from different colour spaces. *IET Image Processing*, *14*(13), 3273–3281.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, *58*(301), 236–244.

Wei, K., Chen, B., Li, Z., Chen, D., Liu, G., Lin, H., & Zhang, B. (2022). Classification of tea leaves based on fluorescence imaging and convolutional neural networks. *Sensors*, *22*(20), 7764.

Wilkes, T. C., McGonigle, A. J., Pering, T. D., Taggart, A. J., White, B. S., Bryant, R. G., & Willmott, J. R. (2016). Ultraviolet imaging with low cost smartphone sensors: development and application of a raspberry pi-based uv camera. *Sensors*, *16*(10), 1649.

Wong, T.-T., & Yeh, P.-Y. (2019). Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, *32*(8), 1586–1594.

Wu, D., & Sun, D.-W. (2013). Colour measurements by computer vision for food quality control–a review. *Trends in food science & technology*, *29*(1), 5–20.

Wu, J., Chen, J., Xiong, H., & Xie, M. (2009). External validation measures for k-means clustering: A data distribution perspective. *Expert Systems with Applications*, *36*(3), 6050–6061.

Wu, J., Cui, Y., Sun, X., Cao, G., Li, B., Ikeda, D. M., ... Li, R. (2017). Unsupervised clustering of quantitative image phenotypes reveals breast cancer subtypes with distinct prognoses and molecular pathways. *Clinical Cancer Research*, *23*(13), 3334–3342.

Wu, J., Zhang, B., Zhou, J., Xiong, Y., Gu, B., & Yang, X. (2019). Automatic recognition of ripening tomatoes by combining multi-feature fusion with a bi-layer classification strategy for harvesting robots. *Sensors*, *19*(3).

Wu, T., Gu, X., Shao, J., Zhou, R., & Li, Z. (2021). Colour image segmentation based on a convex k-means approach. *IET Image Processing*, *15*(8), 1596–1606.

Xu, B., Zhang, B., Kang, Y., Wang, Y., & Li, Q. (2012). Applicability of CIELAB/CIEDE2000 formula in visual color assessments of metal ceramic restorations. *Journal of Dentistry*, *40*, e3–e9.

Xu, D., & Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of data science*, *2*, 165–193.

Yang, J., Shen, F., Wang, T., Luo, M., Li, N., & Que, S. (2021). Effect of smart phone cameras on color-based prediction of soil organic matter content. *Geoderma*, *402*, 115365.

Yerliyurt, K., & Sarıkaya, I. (2022). Color stability of hybrid ceramics exposed to beverages in different combinations. *BMC Oral Health*, *22*(1), 180.

Yılmaz, U., Tutus, A., & Sönmez, S. (2022). Effects of using recycled paper in inkjet printing system on colour difference. *Pigment & Resin Technology*, *51*(3), 336–343.

Yong, H., Huang, J., Hua, X., & Zhang, L. (2020). Gradient centralization: A new optimization technique for deep neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16* (pp. 635–652).

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, *27*.

Yulita, I. N., Amri, N. A., & Hidayat, A. (2023). Mobile application for tomato plant leaf disease detection using a dense convolutional network architecture. *Computation*, *11*(2), 20.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, L., Xia, R., Yang, B., Zhang, J., & Wang, J. (2024). MSFNet-2SE: A multi-scale fusion convolutional network for Alzheimer's disease classification on magnetic resonance images. *International Journal of Imaging Systems and Technology*, *34*(4), e23112. https://doi.org/https://doi.org/10.1002/ima.23112

Zhang, Q., Zhuo, L., Li, J., Zhang, J., Zhang, H., & Li, X. (2018). Vehicle color recognition using multiple-layer feature representations of lightweight con-

volutional neural network. *Signal Processing*, *147*, 146–153.

Zhang, W., Li, X., & Ding, Q. (2019). Deep residual learning-based fault diagnosis method for rotating machinery. *ISA transactions*, *95*, 295–305.

Zhang, X., Li, X., Feng, Y., & Liu, Z. (2015). The use of ROC and AUC in the validation of objective image fusion evaluation metrics. *Signal processing*, *115*, 38–48.

Zhbanova, V. L. (2020). Evaluation and selection of colour spaces for digital systems. *Light & Engineering*, *28*(6).

Zheng, Y., Iwana, B. K., Malik, M. I., Ahmed, S., Ohyama, W., & Uchida, S. (2021). Learning the micro deformations by max-pooling for offline signature verification. *Pattern Recognition*, *118*, 108008.

Zhong, X., & Ban, H. (2022). Pre-trained network-based transfer learning: A small-sample machine learning approach to nuclear power plant classification problem. *Annals of Nuclear Energy*, *175*, 109201.

Zhou, J., & Glotzbach, J. (2007). Image pipeline tuning for digital cameras. In *2007 IEEE International Symposium on Consumer Electronics* (pp. 1–4).

Zhu, D., & Qiu, D. (2021). Residual dense network for medical magnetic resonance images super-resolution. *Computer Methods and Programs in Biomedicine*, *209*, 106330.

Zou, F., Shen, L., Jie, Z., Zhang, W., & Liu, W. (2019). A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition* (pp. 11127–11135).

Journal of Packaging Technology and Research (2022) 6:187–198
https://doi.org/10.1007/s41783-022-00137-4

RESEARCH ARTICLE

# Use of Functional Ink in a Smart Tag for Fast-Moving Consumer Goods Industry

**Jari Isohanni**[1]

Received: 10 October 2021 / Accepted: 21 June 2022 / Published online: 25 July 2022


**Abstract**
In the fast-moving consumer goods (FMCG) industry, current labelling solutions have challenges to meet the track & trace requirements. Currently, FMCG items use mainly paper-based self-adhesive labels with traditional barcodes. These labels are low priced and technically easy to produce and deploy. The shift towards advanced solutions, like radio frequency identification (RFID) or near field communication (NFC) tags, still does not offer a good enough cost/benefit ratio. These advanced solutions have a high unit price or require costly changes in production lines. Still, the industry recognizes the possibilities of smart tags. Recent research has shown that functional inks can operate as cheap sensors. However, more research is needed to take functional inks into the operational FMCG environment. This paper presents one technical solution for an FMCG smart tag. The proposed smart tag builds on traditional QR-Code and Datamatrix markers, printed with standard inks. However, it also has functional ink embedded inside the marker as a sensor. This research experiments how embedding impacts the overall performance of the smart tag decoding. And if the CIEDE2000 color difference algorithm can calculate the state of the sensor. Three different parameter combinations, CIEDE2000(1, 1, 1), CIEDE2000(2, 1, 1), CIEDE2000(2.76, 1.58, 1), and their accuracy are compared. Experiments show that the proposed approach does not negatively affect the decoding performance. And that a color comparison can detect sensor states, especially when the functional ink has high enough color intensity. Between different parameters, CIEDE2000(2.76, 1.58, 1) performed best, especially in the low-intensity test. However, some future research needs to address absolute color value detection and the accuracy of color recognition; especially when if the color has low intensity.

**Keywords** Functional ink · Smart tag · Fast-moving consumer goods · Color difference · Machine vision

## Introduction

From the day mobile devices have had cameras, they have been used to decode markers. The most popular markers today are QR-Code and Datamatrix. These markers can be found in many items starting from fast-moving consumer goods (FMCG) and going all the way to industrial and medical usage. According to Tiwari (2016), the first one-dimensional (1D) marker was invented in the 1960s. Since markers have evolved from 1D barcodes to multi-color two-dimensional (2D) markers (Fig. 1) [56]. Marker development has followed other technologies that relate, like cameras and image vision. Currently, there are multiple marker types, some more general, some more use-case specific.

This research extends standard QR-Code and Datamatrix markers to smart tags. Smart tags primary function is to provide information to its user regarding the status of item [38]. Printed non-electronic smart tags function by communicating through the physical senses (human vision) or machine vision.

Functional inks are used to extend standard QR-Code and Datamatrix markers. These inks form the part of the marker which acts as an active sensor by reacting to environmental variables like humidity, temperature, or light. Functional inks are suitable to the normal label print process of markers. The chemical structure of the functional ink determines how and on which environmental variables ink reacts.

As summary, this research looks into solving following research questions:

✉ Jari Isohanni
jari.isohanni@abo.fi

[1] Faculty of Science and Engineering, Åbo Akademi University, Tuomiokirkontori 3, 20500 Turku, Finland

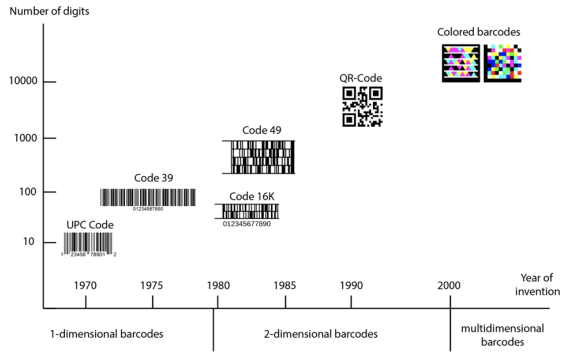



**Fig. 1** Evolution of barcodes, adapted from [56]

- How to embed functional ink inside the QR-Code or Datamatrix?
- How does the embedding impact the decoding performance of QR-Code or Datamatrix?
- Can CIEDE2000 color difference algorithms accurately recognize state of the functional ink?

The rest of the paper is organized as follows. Section "Related Work" goes through related work. Section "Materials and Methods defines what smart tags are, how label printing works, introduces use-cases that relate to smart tags, and looks into the proposed technical approach. Section "Experiments" discusses details of experiments, and in Section "Results and Discussion", results of the experiments are discussed. Finally, Section "Conclusions" concludes the work, followed by future research topics in Section Future Work.

## Related Work

Relevant past research work has focused on exploring possibilities of intelligent packaging in the FMCG industry. As part of intelligent packaging smart tags have been seen as a possibility to track items and their status. For example, this research has been done in the context of food items by Yam [60], Mohebi and Marquez [34], Realini and Marcos. [43], Kalpana et al. [21], and Fuertes et al. [8].

Related research has developed new advanced markers. These markers have one difference from traditional markers. Where traditional markers use black and white values to present binary values, advanced markers use a limited spectrum of colors. Parikh and Jancke proposed an approach to recognize multiple colors in 2D color barcodes [36]. Bulan and Sharma proposed using dot orientation, with colors, to encode data into high-capacity barcode [4]. Grillo et al. proposed a new barcode that uses colors to include more data [13]. Bulan and Sharma proposed a barcode that uses the cyan, magenta, and yellow colorant separations to enable high-capacity barcode [5]. Subpratatsavee and Kuacharoen proposed a new barcode that reduces the physical space needed for the barcode. In their research, Subpratatsavee and Kuacharoen present the design and the implementation of a high-capacity two-dimensional barcode [50]. Taveerad and Vongpradhip proposed color QR-Code to hold more data through a novel encoding concept [54]. Common to all these approaches is that in theory, black & white marker data can only be translated into 1s and 0s when decoded. However, colors, or shades, can be translated to as many values as possible to identify [41]. Most of these past studies relate to using colors to encode more data into barcodes. Or how to change data that barcode contains through colors. John and Raahemifar provided an overview of color barcodes. They also proposed a binarization and grouping algorithm to encode data to form a color barcode [20]. Wasule and Metkar took into account the intensity variation which occurs while decoding colored barcodes and proposed an approach that will increase the capacity of barcodes beyond threefold; in their work, they used quantization of grey levels [59]. Ramalho et al. used the concept of super-modules to encode more data into QR-Code but also to make QR-Code more secure [42].

Functional inks and mainly their properties have been studied during recent years. Gao et al., in their research, focused on time-temperature indicators (TTIs) data modelling [9]. Li and Chen used two different functional inks in their proposal as a new Dynamic and Sensitive Barcode (D&S), D&S can react to environmental state change [27]. Chen et al. used a vice-versa approach and made the paper reactive to environmental changes, and they also used smart devices to decode color information [6]. Kulčar et al. focused their research on the properties of functional inks regarding state changes [24].

One of the most relevant research around functional inks and smart tags is a study by Gligoric et al. In this research, authors defined possible approaches for smart tags [12]. Quite similar work was done by Hakola and Vehmas [14]. However, these studies mainly focus on uses-cases and properties of functional inks, and only briefly define technical frameworks for ink-based smart tags

This research uses CIEDE2000 (KL, KC, KH) algorithm for color difference calculations. CIEDE2000 has been used in many studies where color difference is matched against human perception or in general color comparison. This work has been conducted for example by Taoa et al. [52], where authors used the algorithm to recognize leaf color differences. By Nguyen et al. [35] in the context of rice color recognition and by López et al. [30] for mortar color differences. Usage of KL = 1, KC = 1, and KH = 1 parametric values is the most common approach when using

CIEDE2000 [11, 31, 37]. Past research has also achieved good results with KL = 2, KC = 1, and KH = 1 parametric values [10, 32]. The third set of parametric values used in this research is KL = 2.76, KC = 1.58, and KH = 1, which has been used in the past, especially in digital and printed images [29].

The past research approaches smart tags from different point-of-view; they rarely consider the actual FMCG label printing process. And previous color difference research has focused on color difference calculations; especially, in use-cases where color comes from a physical object, or colors have a different hue.

Contributions of this research work are:

(a) Technical approach to extend standard QR-Code and Datamatrix markers to smart tags, without significant impact on marker decoding performance.
(b) Verify the suitability of embedding functional inks into QR-Code and Datamatrix markers.
(c) CIEDE2000 algorithms' parameter comparison with printed colors.

## Materials and Methods

### Labels

Smart tags have been defined in the past research in various ways. They might refer to radio-enabled electronic devices, e.g., [62], radio frequency identification (RFID), or near field communication (NFC) tags, e.g., [1]. Smart tags have also been considered as printed electronics, e.g., [45].

In this research, smart tags are standard two-dimensional markers with added intelligence from functional inks. Smart tags are printed on self-adhesive labels, and finally glued to FMCG products. This paper does not take into account labelling/printing that is done directly on items, like laser marking. According to Kirwan (2012), the invention of self-adhesive labels was in the mid-1930s. Labels were back then used to apply price and decoration on store items. Currently, the FMCG industry uses labels to add value to a product item during its life-cycle. The latest development of labels has made them smart, smart-active, or smart-intelligent (Fig. 2). Smart labels have various usages like tracking products, monitoring their temperature, and indicating food freshness [23].

Smart tags have the functionality to react to environmental changes and identify individual items, rather than presenting only static content. This categorizes smart tags as customized labels [14]. Smart tags defined in such a way are part of intelligent packaging, an emerging technology in the FMCG. Intelligent packaging uses smart tags as

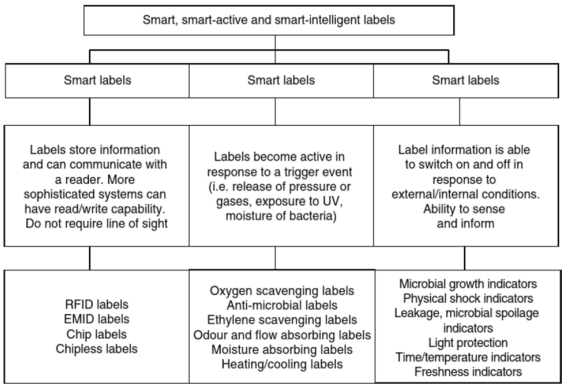



**Fig. 2** Categories of labels [23]

the communication function between the package and the user. Smart tags and their information help in the decision-making related to the item, to achieve the added value, like enhanced food quality, user experience, and safety [60].

### Label Printing Process

Labels with smart tags are either (a) pre-printed in print houses or (b) printing occurs just before they are applied (in-house) [49].

Depending on the chosen label printing facility, pre-print or in-house, multiple technologies can be used to print labels.

- Rotary and semi-rotary letterpress.
- Flexographic printing.
- UV-flexographic printing.
- Screen printing.
- Offset printing.
- Rotogravure printing.
- Thermal printing.
- Laser and inkjet printing [23].

All listed technologies are suitable for the pre-printing of labels. For labels printed in the production line, thermal, laser, and inkjet printing are plausible. These non-conventional technologies can also print customized information that varies per label. It is also possible to mix conventional and non-conventional ways of printing; this is called hybrid printing [23].

Electronic/non-electronic labels can occupy as much as 50% of the package costs. However, in most low-priced

products, like food, the price of the package should not exceed 10% of the total costs of the product [58].

## Functional Inks

Functional inks can report exposure to environmental influences by switching between two states of optical properties. The state of the ink and its absolute value depends on the current properties of the used ink. Functional inks can ultimately appear either in an active (1) or non-active (0) binary state. Switch between states is relative to physical influences. For example, water/humidity (hydrochromism), temperature (thermochromism), and the intensity of light (photochromism) can change the state of the functional ink [15].

Depending on the environmental influence, and properties of the ink, change between two binary states occurs in a specified time. During this period, functional ink can also have values between 0 and 1. Optically visible change between states can either be a change between colors or color-changing its intensity. Also, depending on the chemical compound of the ink, change can be reversible or irreversible [3].

According to Zabala et al., the change in functional ink can represent accumulated exposure (total), or if the ink has exceeded its activation point [61]. Therefore, functional inks can track continuous exposure during the whole life-cycle, or exposure to an environment lower/higher than the threshold set.

Functional inks in this research are compatible with the high-speed printing process (conventional, non-conventional, and hybrid). And to meet labelling cost requirements, the price of the functional ink is close to the price range of standard color inks. The advantage of the smart tag with functional ink is that there are no electronics in it. This makes the manufacturing technically less complex than RFID/NFC-based smart tags. The advantage is also in recycling, as some regulations define RFID/NFC-based tags as electronics. Using functional ink raises the possibility to design innovative products for markets not yet addressed by the electronic tags [12]. Functional inks are suitable for most printing methods presented in the previous chapter, including flexographic, gravure, screen, and inkjet [28].

When smart tags are developed for the needs of the food industry, they should not impose thread to items inside packages or when in direct contact with food items. There are three main ways of food safety can be compromised by the ink used in labels/packages [46]:

- Migration, components of ink pass through the substrate
- Invisible set-off—transfer of components of ink from the printed side to the food-facing side, for example when packages are stacked.
- Gas-phase transfer of ink components via the air in the packaging to food.

Some functional inks are suitable for short- or long-term direct food contact (DFC). These inks meet regulations set by authorities like EuPIA (European Printing Ink Association) and U.S. Food and Drug Administration (FDA). However, some functional inks can only be used in non-food contact. Past research has also developed options for this like impermeable barriers/papers, which can be used when printing smart tags [61]. As this paper only focuses on proposing a general approach, research of these options is left for future research.

## Use-Cases for Smart Tags in FMCG Industry

Fast-Moving-Consumer-Goods are perishable packaged products purchased and consumed by all members of society. FMCG cover items like food and beverages; often, these goods are also considered as Consumer Packaged Goods (CPG). One of the main purposes of the product packaging is to communicate important information about item(s) in the package to the user. Information user gets through labels, which might relate to the ingredients/material of the product. However, the label can also contain other information, like nutritional facts. Packages, and their labels, look to meet customer experience expectations. However, some of their functionality is also responsible for fostering a circular economy. Part of this is cradle-to-grave tracking and actions that relate. Typical life-cycle (cradle-to-grave) of an product is illustrated in Fig. 3.

As shown in Fig. 3, intelligent packaging links to all phases of the product life-cycle. In this context, smart tags

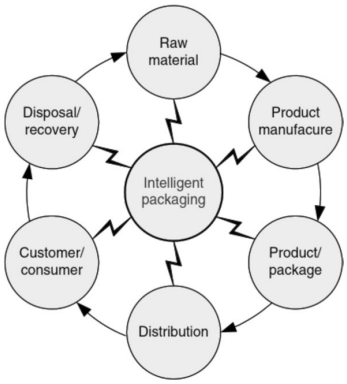



**Fig. 3** Product life-cycle and its relation to intelligent packaging [60]

should deliver various information to the user. This information contains either data about the current status of the product or information that relates to the product's historical status. Depending on the life-cycle phase, user might be interested in a lifetime, temperature history, freshness status, package condition, authenticity, or something else [23].

Smart tag-related research has recognized suitable use-cases which directly or indirectly contribute to meeting the needs of consumers, producers, and manufacturers [7]. These uses-cases are classified as follows:

- Manufacturing (Raw material, Product manufacturing, Packaging).
- Distribution & Consume.
- Recycle [38].

Listed use-cases are related to item identification and providing information about the item to the current stakeholder or user. When the packaged item has a smart tag on its label, the item has intelligent packaging. The intelligent package is "a packaging that contains an external or internal indicator to provide information about aspects of the history of the package and/or the quality" [44]. Some use-cases are more suitable for intelligent packaging than others.

As printed smart tags belong to the same category as RFID tags' most use-cases can be derived from RFID/NFC use-cases [12]. Printed smart tags lack a radio interface, so the decoder must have a line of sight and relatively close distance. However, printed smart tags support a visual interface that low-cost RFID/NFC do not currently have. Generally, the costs of the smart tags are part of the packaging costs.

## Technical Approach for the Smart Tag

Smart tags developed for the FMCG, must be able to: (a) avoid any false negatives (samples that seem safe but are dangerous) and (b) they should have as few false positives (samples that seem unsafe but are healthy) as possible.

If a functional marker detects false negatives, it provides information that the item is safe, although the reality is vice versa. False positives are not that crucial, but providing false positives weakens the trust towards the system. False positives provide information that the item is unsafe even if they are not. The price of the marker is also a meaningful factor and how the user can decode the marker [39].

Intelligent packaging use-cases are dependent on two features. First is a unique identifier, which provides a way to trace items' history and link life-cycle events of the item into databases or other services [38]. The second feature is a use-case-specific sensor printed with functional ink. Therefore, markers must be printed with two different inks, one that is static (unique identifier) and one that can react to environmental changes (sensor) [51].

The following three methods can be when extending markers with functional ink [51]. These are:

- (a) Functional ink placed outside of the marker, a solution used by many of the existing smart tags. This increases the required physical print area and might make decoding of the functional ink state complex.
- (b) Functional ink changes markers' data, and this solution is more advanced and requires specific and matching functional inks and compatible marker contents.
- (c) Functional ink placed inside of the marker, a solution presented in this paper.

In the proposed approach, functional ink is placed inside the marker without disturbing the reserved cells, like a timing pattern. This does not exceed markers' error correction capability. However, due to ink misplacement or dye growth, the approach uses functional ink intensity satisfying requirements for symbols set by GS1 standard (bar code symbol print quality test specification—two-dimensional symbols) [16]. In these requirements, the contrast between the symbols black and white cells should not be lower than 40%. Placing functional ink inside marker also affects modulation criteria, which compares local contrast to global contrast. As functional ink area disturbs the global threshold, it might affect the probability of incorrect cell color identification. In such a case, base markers' capability to recover from error/destroyed data is used while decoding. When using QR-Code or Datamatrix as the base marker, around 30% of destroyed data can be recovered [17, 18].

The proposed approach has the following advantages: (a) it does not occupy more physical space than adding standard QR-Code, (b) QR-Code can work as call-to-action, and (c) it is technically suitable for the current label printing processes. Technically approach might be suitable for use-cases with more than one functional ink (sensor), but more research is needed.

Figure 4 shows proposed approach in QR-Code. In this approach, the sensor area would be defined by rectangle

**Fig. 4** Approach to embed functional ink into QR-Code

$x = 8$

$y = n - 3$

$w = n - 9$

$h = 2,$

where $n$ is the count of cells marker has in vertical/horizontal direction and indexes of the cells start from top-left corner where index is (0,0) and bottom-right corner has index $(n-1, n-1)$. When sensor data are occupying such a rectangle, it does not occupy more space than 24 cells ($\tilde{1}3\%$) of 189 data cells in the smallest possible Qr-Code (21×21) [17]. Some research needs to be done in the future to explore if this amount of sensor cells is enough for accurate color recognition. Theoretically, the sensor could occupy more area within QR-Code. However, this might affect the decode performance of the QR-Code. Especially performance could be lower in challenging environments if more errors occur in other parts of the QR-Code [26]. Using only one row for the sensor might affect sensor data decoding, especially in small QR-Codes. This is because the QR-Code generator algorithm optimizes the ratio between white and black cells, but might sometimes generate rows where black/white cells have the majority [17].

Functional ink inside the marker is straightforward to locate. When decoding a standard QR-Code, an algorithm finds coordinates of each cell. As the algorithm knows the location of each cell, it can also define the coordinates of the sensor area. In the proposed approach, standard QR-Code scanner applications can decode contents of the QR-Code, but cannot decode sensor area information.

Decoding of the sensor value happens with a special algorithm. This algorithm calculates the difference between white and black cells' color values within the sensor area. Then, the difference is compared to the maximum difference,

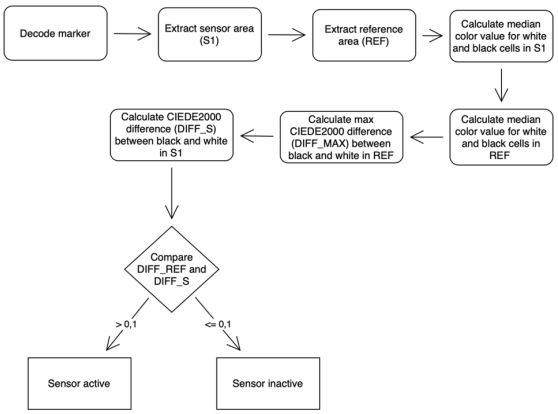



**Fig. 5** Sensor data decoding process of the proposed approach

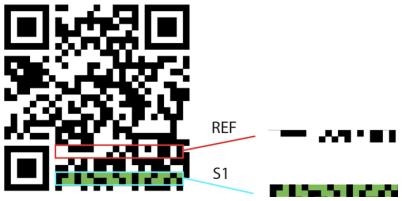



**Fig. 6** Reference area (REF) and sensor area (S1) extraction

calculated in the same way but from the reference area. The color difference is calculated by CIEDE2000(KL, KC, KH) formula [31]. The performance of different parameters is compared later in the experiments section. Figure 5 shows the process chart of the algorithm. The sensor is considered active if its value is over 10% of the reference maximum difference.

Figure 6 shows the reference area (REF) and sensor area (S1). These areas work as sources for the calculation of the white-black or sensor-black difference. The location of the reference area is two rows above the sensor area because of possible functional ink spreading or misplacement. Placing the reference area further from the sensor are might alter it to ambient lighting changes like shadows, different to ones that the sensor has.

Even though in Fig. 4, it shows sensor in the QR-Code marker, the approach should apply to other markers with the error correction capability, like Datamatrix. As such, the addition of the sensor affects the general performance of the marker, discussed in the next chapter. Depending on how much data are recoverable from damaged markers (Datamatrix or QR-Code), height of the sensor area can vary in different sized markers. In this research, sensor area is always two rows in height.

## Experiments

This section focuses on the practical experiments with the proposed approach. The focus of the first experiment is to validate how sensor embedding impacts the decoding performance of the marker. The focus of the second experiment is to test the calculation of the sensor value in simulated use-cases. Finally, the third test experiments with three real-life markers using actual functional ink.

Past research has shown that the size of the marker should be larger than 10×10mm and have at least 4 pixels (px) per cell [40, 53]. Based on this information, for the first two experiments, three different markers in size 20 mm × 20 mm were printed with an office laser-jet (Canon ImageRunner C5535i). Printing was done in 300dpi to standard office A4-paper (Canon Black Label Plus 80 g/m$^2$).

Markers had different data to represent different item-level tracing options:

(a) EAN/UPC-13 data [47], 13 digits, illustrated as 25×25 sized QR-Code
(b) EPC data, 26 digits [55], illustrated as 29×29 sized QR-Code
(c) UUID data [57], 36 digits, illustrated as 37×37 sized QR-Code.

Using different sizes is to experiment with how the proposed approach performs when there is a different amount of pixels per cell.

The third experiment uses real-life markers generated with the proposed approach. These markers were printed with the following functional inks:

(1) LCR Hallcrest cold activated thermochromic ink, clear to green CMYK (22,7,17,0) at 7 °C, 38 $cm^3/m^2$, 1 printed layer [25]. Printed using flexographic printing.
2) Datamatrix with SICPA heat activated thermochromic ink, clear to magenta CMYK(0,100,0,0) at 26 °C, 38 $cm^3/m^2$, 1 printed layer [48]. Printed using screen printing and flexographic printing.
3) Datamatrix with Sunlase TDS sulfuric acid reactive ink, clear to black CMYK(0,0,0,100), 38 $cm^3/m^2$, 1 printed layer Printed using screen printing.

The third experiment has also the difference that it uses Datamatrix codes where functional ink is embedded in the same way as in QR-Code used previously.

## QR-Code Integrity Experiment

The first test experimented with two versions of each marker (a) version where sensor area was transparent (sensor off), in Fig. 7 top row. And (b) version where sensor was totally black (sensor on), in Fig. 7 bottom row.

Latter markers simulate a situation where the sensor destroys all white cells within the specified data region, and error correction recovers these data. This experiment looks to validate that adding a sensor does not prevent the decoding of the marker.

**Fig. 7** Markers for the first experiment

In this experiment, iPhone 7 and iPhone 11 Pro were used, with a standard camera application, to decode QR-Code markers from different distances. Controlled environment, 25 cm × 20 cm × 40 cm sized white box, was used for the test. The test environment had a moving sledge for distance control and adjustable LED lights for ambient lighting control. Decoding distance was measured by moving the sledge to the furthest distance where decoding of the marker happens continuously. In this experiment, decoding distance was tested in two ambient light levels, 400lx (office lighting) (Table 1) & 15lx (dark environment) (Table 2).

## Sensor Value Calculation Experiment

The second experiment focuses on testing if the algorithm defined in the previous section can identify sensor states when different sensor colors are used. Second experiment uses same marker sizes and contents as the first experiment. However, this time with different sensor colors and their intensities. Four pure colors magenta ($C = 0.0, M = 1.0, Y = 0.0, K = 0.0$), cyan ($C = 1.0, M = 0.0, Y = 0.0, K = 0.0$), yellow ($C = 0.0, M = 0.0, Y = 1.0, K = 0.0$) and black ($C = 0.0, M = 0.0, Y = 0.0, K = 1.0$). And three mixed colors red ($C = 1.0, M = 1.0, Y = 0.0, K = 0.0$), green ($C = 1.0, M = 0.0, Y = 1.0, K = 0.0$) and purple ($C = 1.0, M = 1.0, Y = 0.0, K = 0.0$) were used.

For each color, three marker sizes were printed with different intensity (100%, 80%, 60%, 40%, and 20%) levels. Markers were captured, with iPhone 7, Nokia TA-1032, and iPhone 11 Pro to images from two distances 10 cm and 20 cm, in the same ambient light levels as in experiment one. Aside from capturing markers in a controlled environment, images were captured in random contexts, including daily environments like fridges, office tables, and store shelves.

**Table 1** Experiment results of the 400lx experiment

| Size | No sensor (cm) | Sensor (cm) | Difference (%) |
|---|---|---|---|
| 25 × 25 | 26.3 | 24.7 | 6 |
| 29 × 29 | 24.7 | 23.7 | 4 |
| 37 × 37 | 24.0 | 24.0 | 0 |

**Table 2** Experiment results of the 15lx experiment

| Size | No sensor (cm) | Sensor(cm) | Difference (%) |
|---|---|---|---|
| 25 × 25 | 25.5 | 23.8 | 7% |
| 29 × 29 | 24.0 | 23.0 | 4% |
| 37 × 37 | 23.5 | 22.6 | 4% |

**Fig. 8** Examples of photos used for the second experiment

This totalled 1260 images for the second experiment, and examples of photos used in this experiment are shown in Fig. 8.

In the second experiment, QR-Codes were decoded with a custom Python application [19]. This application used algorithm shown in Fig. 5 to recognize sensor state. In the experiment, three CIEDE formulas with different parametric values for *KL*, *KC*, and *KH* were used to calculate color difference. These formulas were CIEDE2000(1,1,1), CIEDE2000(2.76,1.58,1), and CIEDE2000(2,1,1).

Tables 3, 4, and 5 show the results of the second experiment. Results are separated into three tables depending on parametric values. The first column shows sensor color, and the following columns show the lowest and highest value of the sensor. These results can be used to estimate the usability and accuracy of the algorithm and its parameters.

The third and final test experimented with smart tags printed with real functional inks. These smart tags were based on Datamatrix and photographed in a real-life environment with iPhone 7, Nokia TA-1032, and iPhone 11 Pro.

**Table 3** Results from the second experiment, CIEDE2000 (1,1,1) algorithm

| CMYK color | 20% | 40% | 60% | 80% | 100% |
|---|---|---|---|---|---|
| No sensor | [0,00 ... 0,05] | | | | |
| (0,0,0,1) | [0,16 ... 0,27] | [0,31 ... 0,60] | [0,58 ... 0,87] | [0,82 ... 0,95] | [0,60 ... 1,00] |
| (1,0,0,0) | [0,07 ... 0,17] | [0,17 ... 0,36] | [0,28 ... 0,36] | [0,37 ... 0,49] | [0,47 ... 0,61] |
| (0,1,0,0) | [0,06 ... 0,24] | [0,16 ... 0,31] | [0,29 ... 0,46] | [0,29 ... 0,61] | [0,43 ... 0,80] |
| (0,0,1,0) | [0,04 ... 0,12] | [0,08 ... 0,17] | [0,08 ... 0,16] | [0,12 ... 0,14] | [0,14 ... 0,23] |
| (1,1,0,0) | [0,15 ... 0,28] | [0,41 ... 0,51] | [0,64 ... 0,69] | [0,81 ... 0,88] | [0,75 ... 0,95] |
| (0,1,1,0) | [0,08 ... 0,18] | [0,17 ... 0,31] | [0,22 ... 0,47] | [0,35 ... 0,61] | [0,44 ... 0,61] |
| (1,0,1,0) | [0,10 ... 0,25] | [0,22 ... 0,31] | [0,31 ... 0,55] | [0,39 ... 0,42] | [0,53 ... 0,75] |

**Table 4** Results from the second experiment, CIEDE2000 (2.76,1.58,1) algorithm

| CMYK color | 20% | 40% | 60% | 80% | 100% |
|---|---|---|---|---|---|
| No sensor | [0,00 ... 0,05] | | | | |
| (0,0,0,1) | [0,16 ... 0,27] | [0,31 ... 0,60] | [0,58 ... 0,87] | [0,82 ... 0,95] | [0,60 ... 1,00] |
| (1,0,0,0) | [0,08 ... 0,16] | [0,19 ... 0,36] | [0,30 ... 0,37] | [0,41 ... 0,50] | [0,50 ... 0,63] |
| (0,1,0,0) | [0,10 ... 0,24] | [0,23 ... 0,34] | [0,36 ... 0,49] | [0,36 ... 0,63] | [0,48 ... 0,81] |
| (0,0,1,0) | [0,11 ... 0,15] | [0,16 ... 0,21] | [0,21 ... 0,39] | [0,30 ... 0,33] | [0,27 ... 0,32] |
| (1,1,0,0) | [0,19 ... 0,29] | [0,28 ... 0,38] | [0,65 ... 0,72] | [0,82 ... 0,90] | [0,75 ... 0,97] |
| (0,1,1,0) | [0,09 ... 0,18] | [0,17 ... 0,31] | [0,23 ... 0,47] | [0,36 ... 0,61] | [0,46 ... 0,63] |
| (1,0,1,0) | [0,14 ... 0,29] | [0,28 ... 0,38] | [0,36 ... 0,66] | [0,45 ... 0,53] | [0,56 ... 0,76] |

**Table 5** Results from the second experiment, CIEDE2000 (2,1,1) algorithm

| CMYK color | 20% | 40% | 60% | 90% | 100% |
|---|---|---|---|---|---|
| No sensor | [0,00 ... 0,05] | | | | |
| (0,0,0,1) | [0,16 ... 0,27] | [0,31 ... 0,60] | [0,58 ... 0,87] | [0,82 ... 0,95] | [0,60 ... 1,00] |
| (1,0,0,0) | [0,07 ... 0,16] | [0,18 ... 0,36] | [0,30 ... 0,37] | [0,39 ... 0,49] | [0,48 ... 0,62] |
| (0,1,0,0) | [0,08 ... 0,24] | [0,20 ... 0,32] | [0,33 ... 0,48] | [0,33 ... 0,62] | [0,45 ... 0,80] |
| (0,0,1,0) | [0,08 ... 0,13] | [0,13 ... 0,19] | [0,15 ... 0,29] | [0,23 ... 0,25] | [0,22 ... 0,26] |
| (1,1,0,0) | [0,17 ... 0,29] | [0,43 ... 0,52] | [0,64 ... 0,70] | [0,81 ... 0,89] | [0,75 ... 0,95] |
| (0,1,1,0) | [0,09 ... 0,18] | [0,18 ... 0,31] | [0,23 ... 0,48] | [0,36 ... 0,61] | [0,46 ... 0,63] |
| (1,0,1,0) | [0,12 ... 0,27] | [0,25 ... 0,34] | [0,34 ... 0,60] | [0,42 ... 0,47] | [0,54 ... 0,75] |

**Fig. 9** Examples of photos used for the third experiment

**Table 6** Results from the third experiment

| Marker | (1,1,1) | (2.76,1.58,1) | (2,1,1) |
|---|---|---|---|
| Sensor off | [0,01 ... 0,03] | [0,01 ... 0,03] | [0,01 ... 0,03] |
| 100% black | [0,65 ... 0,72] | [0,65 ... 0,72] | [0,65 ... 0,72] |
| 100% magenta | [0,36 ... 0,39] | [0,52 ... 0,54] | [0,45 ... 0,46] |
| 40% green | [0,12 ... 0,31] | [0,15 ... 0,31] | [0,13 ... 0,31] |

First before exposing them to a reactive environment. After exposure, markers were photographed again. Fifty images were captured for each marker and sensor state, totalling 300 images. Examples of photos are seen in Fig. 9.

The same Python application as in experiment 2 was used to decode markers. Table 6 shows the results of the third experiment. Columns show different CIEDE200 parameters, and values range from decoding. On the rows are different marker states

## Results and Discussion

Based on the results from the first experiment, adding a sensor inside the marker reduces the decoding distance only slightly. Distance is decreased more in a dark environment, but the absolute change in the decoding distance is small. In everyday use, this does not have a significant impact. Markers with more data are affected less as the sensor occupies a relatively smaller area from the data area.

In the second experiment, simulated sensors with different intensities [0%, 20%, 40%, 60%, 80%, and 100%] were decoded with the proposed approach. The second experiment included sensor value calculations with the CIEDE2000 algorithm. Three different parametric (*KL*, *KC*, and *KH*) values were tested, (1,1,1),(2.76,1.58,1) and (2,1,1). Results show that when sensor intensity is 40% or more, all parametric combinations can recognize the state of the sensor. When intensity is lower (20%), calculations are not working accurately, and incorrect states are sometimes recognized. Especially, algorithm has challenges with the yellow ink. With higher intensities, the algorithm gives values over 0.1. This can be considered as a threshold value. When the sensor has a value over 0.1, color exists, and the sensor is ON. Values vary highly, especially with higher intensities. Therefore, the algorithm cannot be used for accurate color value identification.

The third experiment showed that the proposed approach works well with actual functional inks. However, the algorithm provides sometimes lower values for the sensor as in experiment two. This might be from the label background color differences or calibration of the ink intensity. In the third experiment, all CIEDE2000 parameter combinations provided values that ranged quite widely. In practice, this means that the approach is suitable for sensor state recognition. However, it fails when used to recognize the absolute value of the sensor.

Past research has shown that it is possible to detect colors with smartphone cameras [6]. However, due to restrictions like camera accuracy, ambient light, print, and the paper quality, limited amount of color shades can be recognized [2, 22]. In the past, multiple approaches for color identification in barcodes have been proposed, for example, by John and Raahemifar [20], Bagherinia and Manduchi [2]. In these approaches, different colors represent different data. The same restrictions apply when functional ink operates as a sensor, and a limited amount of shades can be recognized. The most reliable way to use functional ink is to use high enough color intensity. It is challenging to recognize the actual shade of the functional ink. However, recognition of functional inks states, ON or OFF, is quite straightforward. The proposed approach cannot recognize shades accurately, as there is variation in results. One possible approach to overcome this problem might be calibration. In traditional calibration, color charts are used for camera calibration [2]. However, in the case of smart tags, calibration has a small amount of data in its use. These data include only labels background color (paper color), black cells of the marker, and possible color reference points. Together with calibration, reference points can be used. With reference points, it is possible to compare the state of functional ink into specified reference values [6]. However, if reference points are used, they must be printed with color as close as possible to functional ink's value in a specific state.

Actual color shade recognition with functional ink is a challenging topic as the shade is depending on multiple factors. Ink intensity is the most important factor. However, the direction of state change that occurred last has its impact. In other words, color is slightly different if it has been reached during a negative or positive change. Also, change between states has different speeds in different environments and colors. In some cases, when changing from color to transparent, some color can remain [24].

## Conclusions

To support the data-driven applications in the FMCG sector, more data need to be collected. Without this data collection, it is not possible to implement optimal decision-making

processes. Currently, the obstacle is not on cloud services or data processing. But on how we collect data from low-cost items and track & trace them during their life-cycle. One approach is to integrate low-cost smart tags with functional ink into items. Functional ink would be able to react to environmental changes and work as a sensor.

Implementation of such markers can be done in various ways, and this paper presented a low-cost way to embed functional ink inside a standard marker. With this approach, the marker has two features: a unique identifier and sensor. Unique identifiers are used to track the item, and the sensor can react to environmental changes. Different color and ink combinations can react to different environmental variables like temperature, time, and humidity. Information from the sensor can be decoded by observing the color changes. One approach to recognize the color and state of the sensor is to use a simple color comparison. CIEDE2000 algorithm with (2.76,1.58,1) parametric values fits for this purpose, and it works reliably when the sensor's color intensity is 40% or higher. On lower intensities, proposed approach sometimes provides incorrect results.

Yet, more work is needed in the field of color recognition. If color can be recognized accurately, it would allow more use-cases around functional inks and make functional inks usable for use in challenging environments.

## Future Work

More research is needed on the computer vision side, especially on how to decode sensor color and its value better accuracy. Also wrapping and challenging conditions might impact proposed approach. Some options, which might achieve better results, could be advanced mathematical calculations, usage of reference colors, or using calibration methods. Usage of machine learning (ML) or artificial (AI) intelligence could also be one option. However, this might make the solution more complex, unless ML/AI solution can be run in a decoding device, rather than in the cloud. When ML/AI is considered, it could be possible to build a rich database containing information about data to color conversion or artificial intelligence models can be trained and used [33].

**Funding** Open access funding provided by Abo Akademi University (ABO). This work was supported by Finnish Cultural Foundation's Central Ostrobothnia Regional Fund (Grant Number 25211242).

## Declarations

**Competing interests** The authors have not disclosed any competing interests.



## References

1. Abad E, Palacio F, Nuin M, De Zarate AG, Juarros A, Gómez JM, Marco S (2009) Rfid smart tag for traceability and cold chain monitoring of foods: demonstration in an intercontinental fresh fish logistic chain. J Food Eng 93(4):394–399
2. Bagherinia H, Manduchi R (2011) A theory of color barcodes. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp 806–813. https://doi.org/10.1109/ICCVW.2011.6130335
3. Bilgin M, Backhaus J (2017) Intelligent codes by controlled responsiveness to external stimuli. In: Printing future days 2017 7th international scientific conference on print and media technology, pp 85–90
4. Bulan O, Monga V, Sharma G (2009) High capacity color barcodes using dot orientation and color separability. In: Delp EJ III, Dittmann J, Memon ND, Wong PW (eds) Media forensics and security, vol 7254. International Society for Optics and Photonics, SPIE, Bellingham, pp 397–403
5. Bulan O, Sharma G (2011) High capacity color barcodes: per channel data encoding via orientation modulation in elliptical dot arrays. IEEE Trans Image Process 20(5):1337–1350
6. Chen Y, Fu G, Zilberman Y, Ruan W, Ameri SK, Zhang YS, Miller E, Sonkusale SR (2017) Low cost smart phone diagnostics for food using paper-based colorimetric sensor arrays. Food Control 82:227–232
7. Farizal, Abhirama P (2019) Feasibility analysis of implementing logistic information system with internet of things technology on a fmcg company. In: 2019 IEEE 6th international conference on engineering technologies and applied sciences (ICETAS), pp 1–6. https://doi.org/10.1109/ICETAS48360.2019.9117563
8. Fuertes G, Soto I, Carrasco R, Vargas M, Sabattin J, Lagos C (2016) Intelligent packaging systems: sensors and nanosensors to monitor food quality and safety. J Sens 2016:4046061
9. Gao T, Tian Y, Zhu Z, Sun D (2020) Modelling, responses and applications of time-temperature indicators (TTIs) in monitoring fresh food quality. Trends Food Sci Technol 99:311–322
10. Ghinea R, Herrera L, Ionescu A, Pomares H, Pulgar R, Paravina R et al (2011) Dental ceramics: a ciede2000 acceptability thresholds for lightness, chroma and hue differences. J Dent 39:e37-44
11. Ghinea R, Pérez MM, Herrera LJ, Rivas MJ, Yebra A, Paravina RD (2010) Color difference thresholds in dental ceramics. J Dent 38:e57–e64
12. Gligoric N, Krco S, Hakola L, Vehmas K, De S, Moessner K, Jansson K, Polenz I, van Kranenburg R (2019) Smarttags: Iot product passport for circular economy based on printed sensors and unique item-level identifiers. Sensors 19(3):586
13. Grillo A, Lentini A, Querini M, Italiano GF (2010) High capacity colored two dimensional codes. In: Proceedings of the

international multiconference on computer science and information technology, pp 709–716. https://doi.org/10.1109/IMCSIT.2010.5679869
14. Hakola L, Vehmas K (2018) Functional ink formulation for individualized smart tags. In: NIP & digital fabrication conference, number 1. society for imaging science and technology, pp 211–214. https://doi.org/10.2352/ISSN.2169-4451.2018.34.211
15. Harvey J (2007) Mechanical engineers' handbook: materials and mechanical design, volume 1, vol 1. Wiley, Hoboken, pp 1423–1436
16. International Organization for Standardization (2011) Automatic identification and data capture techniques—bar code symbol print quality test specification - two-dimensional symbols. Standard, International Organization for Standardization, Geneva
17. International Organization for Standardization (2015) Automatic identification and data capture techniques—qr-code bar code symbology specification. Standard, International Organization for Standardization, Geneva
18. International Organization for Standardization (2018) Information technology—international symbology specification—data matrix. Technical report
19. Isohanni J (2021) Qr-code sensor decoder. https://github.com/jariisohanni/QR-code-sensor-decoder. Accessed 5 Aug 2021
20. John RA, Raahemifar K (2015) Designing a 2d color barcode. In 2015 IEEE 28th Canadian Conference on electrical and computer engineering (CCECE), pp 297–301. https://doi.org/10.1109/CCECE.2015.7129203
21. Kalpana S, Priyadarshini SR, Leena MM, Moses JA, Anandharamakrishnan C (2019) Intelligent packaging: trends and applications in food systems. Trends Food Sci Technol 93:145–157
22. Kim M, Song K, Kang M (2018) No-reference contrast measurement for color images based on visual stimulus. IEEE Access 6:1
23. Kirwan M (2012) Handbook of paper and paperboard packaging technology, 2nd edn. Wiley, Hoboken
24. Kulčar R, Friškovec M, Knešaurek N, Sušin B, Klanjšek Gunde M (2009) Colour changes of uv-curable thermochromic inks. In: Proceedings of the 36th international research conference of iarigai advanced in printing and media technology, pp 429–434
25. Hallcrest LCR (2021) Thermochromic free flowing powder technical data. Whitepaper, LCR Hallcrest
26. Li L, Qiu J, Lu J, Chang C (2016) An aesthetic qr-code solution based on error correction mechanism. In Journal of Systems and Software. 116:85–94. https://doi.org/10.1016/j.jss.2015.07.009
27. Li Z, Chen W (2014) D&S Barcode: A Dynamic and Sensitive Barcode for Intelligent Environment Monitoring. In: 2014 International Conference on Intelligent Environments, pp 47–51. https://doi.org/10.1109/IE.2014.14
28. Lindqvist U, Eiroma K, Hakola L, Jussila S, Kaljunen T, Moilanen P, Rusko E, Siivonen T, Välkkynen P (2008) Technical innovations and business from printed functionality. Number 2436 in VTT Tiedotteita—Meddelanden—Research Notes. VTT Technical Research Centre of Finland, Finland. Project code: 4312
29. Liu H, Huang M, Liu Y, Wu B, Xu Y (2012) Color difference evaluation and calculation for digital and printed images. In: NIP & Digital Fabrication Conference 2012(1):140–143
30. López A, Guzmán GA, Di Sarli AR (2016) Color stability in mortars and concretes. Part 1: study on architectural mortars. Constr Build Mater 120:617–622
31. Luo MR, Cui G, Rigg B (2001) The development of the cie 2000 colour-difference formula: Ciede 2000. Color Res Appl 26(5):340–350
32. Mangine H, Jakes K, Noel C (2005) A preliminary comparison of cie color differences to textile color acceptability using average observers. Color Res Appl 30(4):288–294
33. Mercan ÖB, Kılıç V, Şen M (2021) Machine learning-based colorimetric determination of glucose in artificial saliva with different reagents using a smartphone coupled $\mu$PAD. Sens Actuators B Chem 329:129037
34. Mohebi E, Marquez L (2015) Intelligent packaging in meat industry: an overview of existing solutions. J Food Sci Technol 52(7):3947–3964
35. Nguyen C, Vo V, Cong Ha N (2022) Developing a computer vision system for real-time color measurement—A case study with color characterization of roasted rice. J Food Eng 316:110821
36. Parikh D, Jancke G (2008) Localization and segmentation of a 2d high capacity color barcode. In: 2008 IEEE workshop on applications of computer vision, pp 1–6. https://doi.org/10.1109/WACV.2008.4544033
37. Pecho OE, Ghinea R, Alessandretti R, Pérez M, Bona AD (2016) Visual and instrumental shade matching using CIELAB and CIEDE2000 color difference formulas. Dent Mater 32(1):82–92
38. Plimmer J (2013) Augmenting and securing the consumer brand experience through smart and intelligent packaging for food, beverages and other fast-moving consumer goods, In: Trends in packaging of food, beverages and other fast-moving consumer goods (FMCG), pp 35–57. https://doi.org/10.1533/9780857098979.35
39. Poyatos-Racionero E, Ros-Lis JV, Vivancos J, Martínez-Máñez R (2018) Recent advances on intelligent packaging as tools to reduce food waste. J Clean Prod 172:3398–3409
40. Qian J, Du X, Zhang B, Fan B, Yang X (2017) Optimization of QR-Code readability in movement state using response surface methodology for implementing continuous chain traceability. Comput Electron Agric 139:56–64
41. Querini M, Italiano G (2014) Reliability and data density in high capacity color barcodes. Comput Sci Inf Syst 11:1595–1615
42. Ramalho J, Correia S, Fu L, Dias L, Adão P, Mateus P, Ferreira R, André PS (2020) Super modules-based active qr-codes for smart trackability and iot: a responsive-banknotes case study. npj Flex Electron 4(1):1–9
43. Realini CE, Marcos B (2014) Active and intelligent packaging systems for a modern society. Meat Sci 98(3):404–419
44. Robertson G (2016) Food packaging: principles and practice, 3rd edn. CRC Press, Boca Roton
45. Salmerón JF, Rivadeneyra A, Agudo-Acemel M, Capitán-Vallvey LF, Banqueri J, Carvajal MA, Palma AJ (2014) Printed single-chip uhf passive radio frequency identification tags with sensing capability. Sens Actuators A Phys 220:281–289
46. Schmid P, Welle F (2020) Chemical migration from beverage packaging materials—a review. Beverages 6(2):37–55
47. Schmidt L, Mitton N, Simplot-Ryl D (2009) Towards unified tag data translation for the internet of things. In: 2009 1st International Conference on wireless communication, vehicular technology, information theory and aerospace electronic systems technology, pp 332–335. https://doi.org/10.1109/WIRELESSVITAE.2009.5172469
48. SICPA Securink Corp (2016) Thermochromic inks. Whitepaper, SICPA Securink Corp
49. Stanislav B, Igor M, Kristijan G (2015) Packaging printing today. In: Faculty of Graphic Arts, University of Zagreb, Croatia Packaging Printing Today, acta graphical, vol 26(4). pp 27–33, SSN: 1848-3828
50. Subpratatsavee P, Kuacharoen P (2012) An implementation of a high capacity 2D barcode. In: Papasratorn B, Charoenkitkarn N, Lavangnananda K, Chutimaskul W, Vanijja V (eds) Advances in information technology. Springer Berlin Heidelberg, Berlin, pp 159–169
51. TagItSmart (2017) Initial enablers for smarttags. Technical report
52. Tao M, Ma X, Huang X, Liu C, Deng R, Liang K, Qi L (2020) Smartphone-based detection of leaf color levels in rice plants. Comput Electron Agric 173:105431
53. Tarjan L, Šenk I, Tegeltija S, Stankovski S, Ostojic S (2014) A readability analysis for QR-Code application in a traceability system. Comput Electron Agric 109:1–11

54. Taveerad N, Vongpradhip S (2015) Development of color qr-code for increasing capacity. In: 2015 11th International Conference on signal-image technology internet-based systems (SITIS), pp 645–648. https://doi.org/10.1109/SITIS.2015.42
55. Thiesse F, Michahelles F (2006) An overview of epc technology. Sens Rev 26(2):101–105
56. Tiwari S (2016) An Introduction to QR-Code Technology. 2016 International Conference on Information Technology (ICIT) 322(10):39–44
57. Triebel D, Reichert W, Bosert S, Feulner M, Okach DO, Slimani A, Rambold G (2018) A generic workflow for effective sampling of environmental vouchers with UUID assignment and image processing. Database J Biol Databases Curation 2018
58. Đurđević S, Novaković D, Zeljković Ž, Avramović D (2016) Using augmented reality technology for controlling state of smart packaging products. Preliminary Report, pp 427–437
59. Wasule S, Metkar S (2017) Improvement in two-dimensional barcode. Sādhanā 42(7):1025–1035
60. Yam KL (2012) 8—Intelligent packaging to enhance food safety and quality. In: Yam KL, Lee DS (eds) Emerging food packaging technologies. Woodhead Publishing Series in Food Science, Technology and Nutrition. Woodhead Publishing, Sawston, pp 137–152
61. Zabala S, Castán J, Martínez C (2015) Development of a time-temperature indicator (TTI) label by rotary printing technologies. Food Control 50:57–64
62. Zhu D, Beeby SP, Tudor MJ, Harris NR (2011) A credit card sized self powered smart sensor node. Sens Actuators A Phys 169(2):317–325

Advances in Computational Intelligence (2024) 4:6
https://doi.org/10.1007/s43674-024-00073-7

ORIGINAL ARTICLE

# Recognising small colour changes with unsupervised learning, comparison of methods

**Jari Isohanni**[1]

Received: 11 April 2023 / Revised: 16 March 2024 / Accepted: 19 March 2024 / Published online: 16 April 2024


**Abstract**
Colour differentiation is crucial in machine learning and computer vision. It is often used when identifying items and objects based on distinct colours. While common colours like blue, red, green, and yellow are easily distinguishable, some applications require recognising subtle colour variations. Such demands arise in sectors like agriculture, printing, healthcare, and packaging. This research employs prevalent unsupervised learning techniques to detect printed colours on paper, focusing on CMYK ink (saturation) levels necessary for recognition against a white background. The aim is to assess whether unsupervised clustering can identify colours within QR-Codes. One use-case for this research is usage of functional inks, ones that change colour based on environmental factors. Within QR-Codes they serve as low-cost IoT sensors. Results of this research indicate that K-means, C-means, Gaussian Mixture Model (GMM), Hierarchical clustering, and Spectral clustering perform well in recognising colour differences when CMYK saturation is 20% or higher in at least one channel. K-means stands out when saturation drops below 10%, although its accuracy diminishes significantly, especially for yellow or magenta channels. A saturation of at least 10% in one CMYK channel is needed for reliable colour detection using unsupervised learning. To handle ink densities below 5%, further research or alternative unsupervised methods may be necessary.

**Keyword** Machine vision, Colour difference, Printed colours, Unsupervised learning

## 1 Introduction

The human visual system (HVS) can separate colours, even under challenging ambient light conditions. A healthy human eye can recognise approximately 100 colour shades in each of its three different types of cone cells. In total HSV can roughly recognise around one million different colours Hurlbert and Ling (2012).

Computer vision (CV) research develops solutions that are as accurate as or more accurate than HSV. Colour imaging research, as part of computer vision research, focuses on colour inspection, sorting, detection, and matching. This research addresses colour inspection/matching, particularly in colour differentiation.

All colours we see are combinations of hue, saturation, and brightness values. In digital imaging, light reflected from an object is captured as a digital representation using a digital camera. The entire process is complex and involves electronics, signal processing, and algorithms. This process is unique to each device and depends on the imaging conditions. Therefore, the resulting digital images always vary slightly, more in non-controlled environments and less in controlled environments. The same colour can appear as many different digital representations, or the same digital presentation can appear as two separate colours.

Computer vision research has identified many solutions for colour differentiation. These approaches are based on mathematical algorithms or artificial intelligence (AI). Mathematical algorithms work well in conditions where colours are clearly different, their location is known, and the data is high quality (Isohanni 2022). Artificial intelligence (supervised and unsupervised) is primarily used to recognise objects of different colours and patterns.

However, the current solutions face challenges when dealing with small colour differences in unknown locations. Previous studies have used unsupervised learning to segment/cluster colours. Although the reported performance of algorithms has improved, past research has found out that unsupervised colour segmentation doesn't work in all use-

Jari Isohanni
x2603813@student.uwasa.fi ; jari.isohanni@gmail.com

[1] Digital Economy, University of Vaasa, Wolffintie 34, 65200 Vaasa, Finland

cases (Xu et al. 2018). Improving the quality of the clustering process or finding the best clustering method can have an impact for example on healthcare (Vishnuvarthanan et al. 2016), smart city (Mao and Li 2019) and agriculture (Abdalla et al. 2019) applications. Challenges unsupervised methods face are a) some algorithms require the setting of clusters before running the algorithm, and b) some algorithms are sensitive to the initial cluster centre guess and might stick in the local optima during the process (Abdalla et al. 2019).

In this study, the recognition of small colour differences with unsupervised learning was investigated by using printed inks. One direct use-case of this research is the colour recognition of functional inks. The development of novel printing methods and inks has enabled the labelling and packaging industry to create innovative labelling methods. Some of these innovations use functional inks. Functional inks change their colour depending on environmental values and it is important to detect this change reliably (Isohanni 2022). This research focuses on recognising colour differences using unsupervised clustering methods and compares different methods, their accuracy, and running time. Contributions of the research are:

- Comparison of unsupervised learning methods in colour difference recognition
- Approach to detect small colour changes in printed colours with unsupervised learning

This research is structured as follows: Sect. 2. contains relevant previous research done in the past. Section 3. defines the methods and materials used in this study. The results of this study are presented in Sect. 4. Finally, Sect. 5. discusses conclusions and future research needs.

## 2 Related work

Colour recognition has its role in object detection, object recognition, image segmentation and many other applications. Most research done around colour recognition focuses on high-level use cases, for example, colour recognition is used in animal/plant recognition (Koubaroulis et al. 2002; Jhawar 2016), in dental applications (Bretzner et al. 2002; Bar-Haim et al. 2009; Riri et al. 2016; Kang and Ji 2010), in face / skin recognition (Yang et al. 2010), in robotics (Rabie 2017; Bazeille et al. 2012) and in intelligent traffic (Gao et al. 2006; Gong et al. 2010; De la Escalera et al. 2003; Zhu and Liu 2006). However there many other use-cases where colour recognition is useful.

Colour recognition can be done by mathematical algorithms, which is the most dominant approach, but during the last decade, artificial intelligence has been applied successfully in many use-cases. In the artificial intelligence context, both supervised and unsupervised learning have been proved as possible approaches. Unsupervised learning is usually used when the clustering of colours is done, or when dominant colours are looked at from the source image (Du et al. 2004; Kuo et al. 2005; Bo et al. 2013; Basar et al. 2020). Supervised learning has been found more suitable in higher level use-cases like object colour recognition (Zhang et al. 2019; Aarathi and Abraham 2017; Feng et al. 2019).

As seen from Table 1, most relevant past research around unsupervised learning has focused on agriculture and healthcare use-case.

Banic et al. used unsupervised colour clustering for image colour calibration, they researched a custom clustering approach and finally achieved results where the median angular error was almost always below 2° (Banic and Loncaric 2018).

Gerke and Xiao studied the usage of two classification strategies a supervised method (Random Trees) and unsupervised approach. They also used graph-cuts for energy optimization. Their results achieved 97.74% accuracy in the context of recognition of urban objects. however, methods had challenges with shadows (Gerke and Xiao 2014).

Dresp-Langley and Wandeto used in their research quantization error from Self-Organising Map (SOM). With using of the quantization error their purpose was to recognise increase of amounts of red or green pixels (Dresp and Wandeto 2020). Results of their research were good but only colour amounts where used, not intensity.

Yavuz and Köse achieved good results in colour clustering in the blood vessel extraction use-case. Even with small colour differences. Authors used combination of K-means and Fuzzy C-means. They also used postprocessing to remove falsely segmented isolated regions (Yavuz and Köse 2017).

Abdallaa et al. used subsequent combination of various unsupervised learning methods (GMM, SOM, FCM and K-Means). They proposed methods to overcome illumination and weather condition challenges in segmentation of infield oilseed rape images. They achieved segmentation accuracy of 96% even in challenging conditions (Abdalla et al. 2019).

Basar et al. developed a novel approach to overcome challenges in the initialisation of the clustering algorithm. Their research focuses on the challenge of defining number of clusters and the initial central points of clusters. Their results improved segmentation quality and reduced the classification error (Basar et al. 2020).

Wang et al. achieved good results, even with small colour contrast changes, when they used first-order colour moments, second-order colour moments, and colour histogram peaks. Their objective was to extract feature vectors from the image. And to realise data dimension reduction. Use-case was to classify solid wood panels with K-means Wang et al. (2021).

**Table 1** Related past research

| Study title | Authors | Colour spaces | Method |
|---|---|---|---|
| Unsupervised learning for colour constancy (Banic and Loncaric 2018) | Banic,Koscevic, Loncaric | RGB | colour Tiger |
| Fusion of airborne laser scanning point clouds and images for supervised and unsupervised scene classification (Gerke and Xiao 2014) | Gerke, Xiao | RGB | Markov random field formulation |
| Unsupervised classification of cell imaging data using the quantization error in a self-organising map (Dresp and Wandeto 2020) | Dresp–Langley, Wandeto | RGB | Self-organising map |
| Blood vessel extraction in colour retinal fundus images with enhancement filtering and unsupervised classification (Yavuz and Köse 2017) | Yavuz, Köse | RGB | K-means, Fuzzy C-means |
| Infield oilseed rape images segmentation via improved unsupervised learning models combined with supreme colour features (Abdalla et al. 2019) | Abdalla, Cen, El-manawy, He | Multiple | Gaussian mixture model (GMM), self-organising map (SOM), fuzzy c-mean(FCM), and k-means algorithms |
| Unsupervised colour image segmentation: a case of RGB histogram based K-means clustering initialisation (Basar et al. 2020) | Basar, Ali, Ochoa-Ruiz, Zareei, Waheed, Adnan | RGB | K-Means |
| Colour classification and texture recognition system of solid wood panels (Wang et al. 2021) | Wang, Zhuang, Liu, Ding, Tang | RGB, HSV, LAB | K-means |

Related work shows that unsupervised learning can be used to classify colours. However there aren't many studies which have focused on recognition of small colour differences, especially in the printed colours. Some studies have also clearly pointed out that unsupervised learning approaches have challenges when foreground and background objects have only slight colour difference.

## 3 Materials and methods

The dataset analysed in this research is available in the Zenodo repository (Isohanni 2023). This study used 25 different modified QR-Codes as the original dataset. QR-Codes had three colour zones in them: black, white, colour. (Fig. 1).

The black zone (1.) was printed with pure black (100K / CMYK(1.0,1.0,1.0,1.0)), the white zone (2.) had no colour (paper white, 0K / CMYK(0.0,0.0,0.0,0.0)) and the third (3.) zone was printed with some colour. All zones had equal size. In the example Fig. 1. colour area with (20 M / CMYK(0.0,0.2,0.0,0.0)) is presented, this means that colour has 20% saturation in magenta channel. Dataset was created by printing colour areas with different ink saturation 20%, 40%, 60%, 80% and 100% and with different colours. Example of a QR-codes with colour saturation 20–100% in magenta (M) channel are illustrated in the following Fig. 2. Other colours or colour combinations used were, C (cyan), Y (yellow), K (black) and combination CY (green). Further experiments were also made with ink saturation's 10%

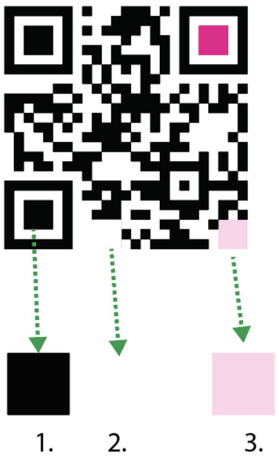



**Fig. 1** QR-code sample

and 5% with unsupervised learning methods that performed best in the first experiments. QR-Codes were printed in size 20 mm × 20 mm. Printed used in this research was an standard office laser-jet (Canon ImageRunner C5535i). And paper that was used was a standard office A4-paper (Canon Black Label Plus 80 g/m2). Printing was done in 300dpi.

All QR-Codes were captured into a image dataset which contained 25–30 images per QR-Code. Different environments were used to capture QR-codes to images, some of the images were captured in normal office ambient light level around 500 lux and colour temperature of around 5000k.


**Fig. 2** Different intensity samples

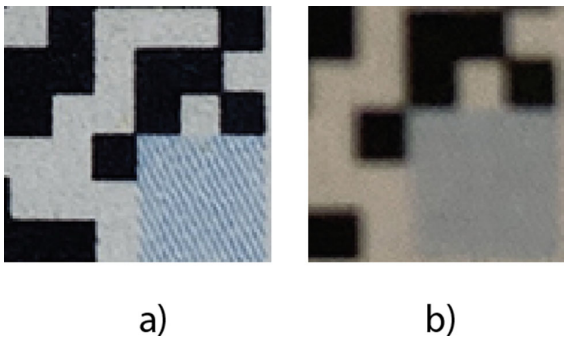



**Fig. 3** Samples from different environments

This higher ambient light environment results more separable colours, but results to more small details (noise) in images. Some of the images were captured in home environment with around 250 lux and 3000 K colour temperature. In darker environment digital cameras might not be able to capture colour information properly (Zamir et al. 2021). Example of differences in these two environments are shown in the following Fig. 3.

Image a) is captured in higher ambient light. This results in more details and clear transforms between colours. Image b) is from lower and warmer ambient light, in these images it can also be seen that camera focus makes image blurry and colours not as clearly separable, however, image has less noise.

All images were taken with iPhone 11 Pro by using standard camera application from around 30 cm distance and stored as JPG's in RGB format with 8 bits / channel. Captured images were resized to 1200×1600 resolution in Photoshop before processing them, no other processing was done. Eventually after resize QR-code occupied around 400×400 px area from the image, and each of the zone was roughly 50×50 px.

### 3.1 Process as whole

The process that was used in this research is shown in the following Fig. 4. Process starts from the RGB JPEG-image, and results into CIELAB values of three (white, black, colour)

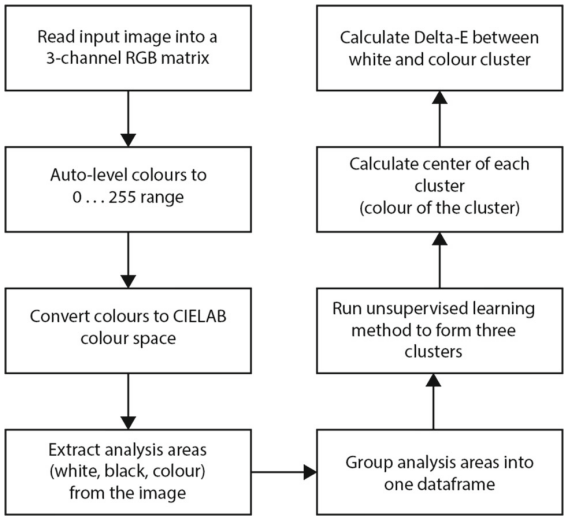



**Fig. 4** Flowchart of the process used

cluster centres (Fig. 5) and Delta-E between white and colour cluster. Delta-E is a measurement which ranges between 0 and 100, it quantifies the difference between two colours, and can be used to determine if two colours are different (Luo et al. 2001). Formula to calculate Delta-E is

$$\Delta E_{00} = \sqrt{\left(\frac{\Delta L'}{k_L \cdot S_L}\right)^2 + \left(\frac{\Delta C'}{k_C \cdot S_C}\right)^2 + \left(\frac{\Delta H'}{k_H \cdot S_H}\right)^2 + R \cdot \frac{\Delta C'}{k_C \cdot S_C} \cdot \frac{\Delta H'}{k_H \cdot S_H}},$$

where

$\Delta E_{00}$ : Delta-E 2000 colour difference
$\Delta L'$ : Difference in lightness
$\Delta C'$ : Difference in chroma
$\Delta H'$ : Difference in hue
$k_L, k_C, k_H$ : Weighting factors
$S_L, S_C, S_H$ : Adjustment factors based on standard deviations
$R$ : Rotation function

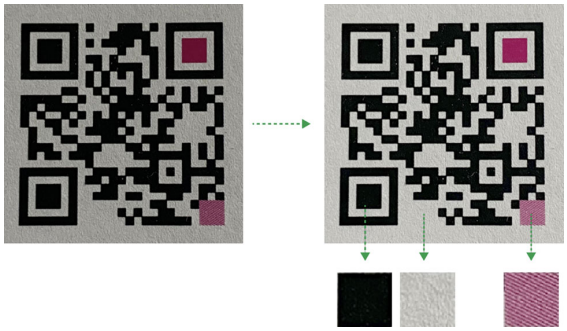


**Fig. 5** Extraction of analysis area

The CIELAB colour system represents quantitative relationship of colours on three axes, lightness (L), and chromaticity coordinates (a,b).

After reading the input image, auto-levelling of colours was performed. Auto-levelling uses following equation:

$$I' = \frac{I - I_{\min}}{I_{\max} - I_{\min}} \times 255,$$

where $I_{\max}$ is the most bright white value in the image and $I_{\min}$ is the darkest black value in the image. Auto-level does a histogram equalisation to achieve more uniform distribution of values in range [0, 255] in all (R,G,B) channels (Kao et al. 2006). For the value 0 the mean value of black area (Fig. 1 area 1.) of the image was used and for the value 255 mean value of the white area (Fig. 1 area 2.) was used. Other values of image are then stretched, and as seen in Fig. 5 this makes difference between colours in the image clearer. Histogram equalisation improves the contrast of an image, but might lead into over enhancement of the image. Some other image enhancement methods could be also used, but this is out of scope of this research and discussed in the conclusions. Result of auto-levelling is shown in Fig. 5. were leftmost image is original image, then one on the right is image after auto-levelling. Three squares that are seen at the right-bottom of the image are extracted colour areas after auto-levelling. Data from these areas is grouped into one dataframe. In the dataframe CIELAB colour format is used, so each row of the frame contained one pixels L, A and B values.

Dataframe was then processed by the unsupervised learning methods. One visual example of the dataframe is presented in the Fig. 6, dataframe LAB colour values are in this figure plotted as density chart. Density of points is shown in different colours, dark blue being less dense and red having highest dense. Circles in the figure are added for illustrative purposes, they show different colour areas (red = black, green = white, yellow = CMYK(0.0, 0.6, 0.0, 0.0)). This is also the result unsupervised learning is expected to achieve. Clusters red and green should stay in quite same location in different images, but yellow cluster moves to different locations in 3D plane. Also, from this figure it can be seen that there is lot of noise present. Noise comes from the digital image and imagining environment.

Clustering of colours was done with different approaches:

- Centroid-based algorithms organise the data in non-hierarchical clusters. These algorithms use distance measuring, like Euclidean, between points to determine if points belong to the same cluster. Usually, centroid-based algorithms run iterations and update cluster centres in each iteration. Centroid-based algorithms are efficient and fast. However, they are sensitive to initial cluster centres and outliers Gonzalez (1985); Hartigan and Wong (1979).
- Connectivity-driven clustering, often referred to as hierarchical clustering, operates under the assumption that points tend to have stronger connections with nearby points compared to those that are farther apart. Algorithms based on connectivity use the distances between points to create clusters. The goal is to minimise the maximum distance required to link these points together Reddy (2021).
- Density-based clustering defines clusters as dense regions of space. Between dense regions, there are regions where data density is lower. Low-density region can also be empty. Density-based clustering algorithms are good at finding arbitrarily shaped clusters, but they have difficulty when it comes to varying densities and if data has high dimensions (Kriegel et al. 2011).
- Distribution-based clustering assumes that the data has a specified number of distributions. Each of these distributions has it's own mean and variance (or covariance). Distribution can be based on Gaussian distribution for example. Points probability to belong to the distribution decreases when it's distance from distribution centre increases Xu et al. (1998).

In this research K-Mean, Fuzzy C-Mean, DBSCAN, MeanShift, Hierarchical clustering, Spectral clustering, Gaussian Mixture Model (GMM), BIRCH and OPTICS from scikit-learn (Pedregosa et al. 2011) were used as unsupervised learning methods. All of these methods were used to cluster dataframe's 3D points (L, A, B) into clusters, if method had option to cluster datapoints into specified amount of clusters it was set to three. Methods used are explained in the following sub-chapters in more detail. The objective of each method was to find the cluster centre or average value of points which had the same cluster.

After clustering clusters were labelled, the cluster closest to CIELAB(1.0, 0.0, 0.0) was labelled as "white", cluster closest to CIELAB(0.0, 0.0, 0.0) got label "black". Finally

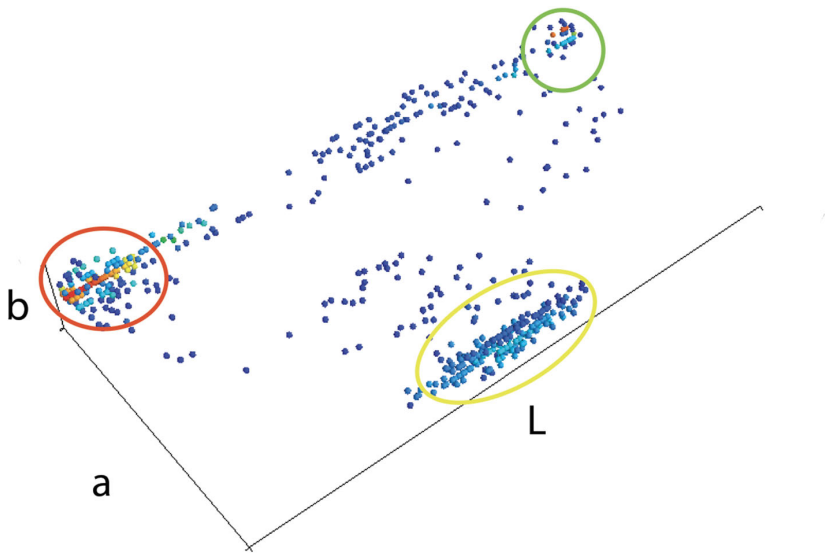



**Fig. 6** Illustration of LAB colour data points intensities

there were left one cluster that was labelled "colour". If only two cluster or more than three major clusters were found, result of the clustering process was considered as failed.

Finally CIEDE2000 Delta-E between "white" and "colour" was calculated. This value was then compared to ground-truth Delta-E calculated from mean values of white and colour areas. Result of the comparison was stored in.CSV file with other image information for later analysis discussed in results section.

### 3.2 K-means

The K-Means clustering algorithm belongs to the partition-based clustering algorithms category. K-means uses a iterative process to partition $n$ observations into $k$ clusters. K-means minimises the sum of the squared Euclidean distance of each point to its cluster centroid. The K-Means works by first choosing $k$ points as initial cluster centres, this can be done in multiple ways. Then algorithm calculates the distance between each cluster centre and each point. Individual points are assigned to the closest cluster centre. After this, the mean of all the data points in each cluster is calculated and used as the new centroid for that cluster. Then again assigning all points to the nearest centroid. This process is repeated until the centroids do not change or until a predetermined number of iterations has been reached. K-Means is so-called hard clustering where a point can belong to one cluster only Lloyd (1982); MacQueen et al. (1967).

This research uses a standard implementation of K-means where the algorithm is given a fixed number of clusters before running the clustering algorithm. "lloyd/full" Lloyd (1982) and "elkan" Elkan (2003) algorithms are experimented. The difference between these two algorithms is that "full" is an expectation-maximisation (EM) algorithm and "elkan" uses the triangular inequality.

K-means looks into minimising total intra-cluster variance, for this squared error function is used:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} \cdot \|x_i - c_j\|^2,$$

where $J$ is the distortion measure. $n$ is the number of data points. $k$ is the number of clusters. $x_i$ represents a data point. $c_i$ represents a cluster centroid. $w_i j$ is a binary indicator variable indicating whether data point $x_i$ is assigned to cluster $j$.

### 3.3 Fuzzy C-mean

The Fuzzy C-Means (FCM) is a so-called soft clustering method. In FCM points can belong to two or more clusters. Each point belongs to every cluster to a certain degree. Points that are located near the centroid of the cluster have a high degree of belonging to this cluster, and point that is located far from the centre has a low degree Bezdek et al. (1984).

Fuzzy C-means starts with an initial guess for the cluster centres for a predefined number of clusters. Then FCM assigns every data point a membership grade for each cluster. In the same way that K-means C-means works iteratively and moves the cluster centres to the right locations. FCM's iteration is based on minimising an objective function that represents the distance from any given data point to a cluster centre weighted by that data point's membership grade Bezdek et al. (1984).

With C-mean, different fuzziness parameters are 2.0–5.0 experimented. The FCM algorithms fuzziness parameter is a key parameter. Larger fuzziness parameters blur the clusters, and all points will finally belong to all clusters Zhou et al. (2014).

In C-means following membership degree function is used.

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij}^{m} \cdot \|x_i - c_j\|^2,$$

where $J$ is the objective function. $n$ is the number of data points. $k$ is the number of clusters. $x_i$ represents a data point. $c_j$ represents a cluster centroid. $u_i\, j$ is the fuzzy membership value of data point $x_i$ in cluster j. And $m$ is the fuzziness parameter (a positive constant).

### 3.4 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN), an algorithm designed for clustering data points where noise exists. When provided with a collection of points, DBSCAN organises them into clusters by assessing the density of their arrangements. Essentially, points that are closely packed together form dense regions and are grouped accordingly. However, if a point is situated significantly apart from its neighbouring points, DBSCAN identifies it as noise or an outlier Ester et al. (1996).

DBSCAN works based on two parameters:

- $\epsilon(eps)$, two points are considered neighbours if the distance between them is smaller than epsilon.
- *minPts* the minimum number of points required to form a dense region Ester et al. (1996).

DBSCAN is very sensitive when it comes to these parameters. *minPts* is easier to decide, as it can be determined from the total pixel amount and cluster count. However $\epsilon(eps)$ is more complex, and some past research has looked into ways to determine optimal $\epsilon$ value (Giri and Biswas 2020). DBSCAN doesn't have similar single objective function as K-means or C-mean, but can be described with the following algorithm.

1. Identify core points based on the density criterion.
2. Connect core points to form clusters using density-reachability.
3. Assign border points to clusters if they are density-reachable from a core point.
4. Identify noise points that are neither core points nor density-reachable from core points.

In this research $\epsilon(eps) = 3.0,\ 5.0,\ 1.0$ is used and $minPts = (T/n)^*\ 0.8$, where $T$ = total pixels and $n$ is cluster count. As result each cluster must have at least 80% of pixels, if total pixel count is divided into $n$ clusters. After running the DBSCAN algorithm density group's average colour value is considered to represent the whole density group.

### 3.5 MeanShift

MeanShift is an unsupervised learning algorithm, MeanShift works in iterations. On each iteration algorithm shifts points to the direction where region have the highest density of data points. On each iteration MeanShift algorithm updates candidates for centroids to be the mean of the points within a given region. This region is also called bandwidth, which is the only parameter given to the MeanShift algorithm. After the update MeanShift filters out near-duplicates so that finally a final set of centroids are left Wu and Yang (2007).

MeanShift's iterative optimization algorithm, which results into a vector that represents the direction in which the density increases the most at the location of the data point, can be expressed as follows:

$$\Delta x = \frac{\sum_{i=1}^{n} K(x - x_i) \cdot x_i}{\sum_{i=1}^{n} K(x - x_i)} - x,$$

where $K(x)$ is a kernel function, often a Gaussian kernel, and $\Delta x$ is the MeanShift vector for a data point $x$. $x_i$ are the other data points in the dataset. In this researched uses bandwidth which is the median of all pairwise distances. Calculation of bandwidth is slow, as it takes time at least quadratic to point count.

### 3.6 Hierarchical clustering

Hierarchical clustering algorithms build a nested cluster by merging or splitting them successively. The final hierarchy of a dataset is represented as a tree. The root of the hierarchy tree gathers all the samples together, and finally, leaves are clusters with only one sample. Hierarchical clustering depends on the so-called linkage function which defines the distance between any two subsets. Linkage functions developed in the past are single linkage, average linkage, complete linkage, Ward linkage, etc. Nielsen (2016).

This research uses AgglomerativeClustering, which is a version of hierarchical clustering that uses a bottom-up approach (Zhao and Qi 2010). The clustering algorithm starts from the situation where each point is in its cluster (leaf). The algorithm starts to merge clusters using Ward's linkage criteria. Ward linkage analyses the variance of clusters and minimises the sum of squared differences within all clusters (Miyamoto et al. 2015). The variance, also known as the sum of squares, is calculated based on the squared Euclidean

distance between data points and the centroid of the cluster.

$$D(X, Y) = \frac{N_X N_Y}{N_X + N_Y} \cdot \|C_X - C_Y\|^2$$

where $D(X, Y)$ is the distance between clusters $X$ and $Y$. $N_X$ and $N_Y$ are the numbers of elements in clusters $X$ and $Y$ respectively. $\|C_X - C_Y\|^2$ is the Euclidean distance between the centroids of clusters $X$ and $Y$.

### 3.7 Spectral clustering

Spectral clustering is very useful when the shape of the cluster is non-convex. This is because spectral clustering focuses on connectivity rather than the compactness of the cluster. This can be the case for example when the cluster has a shape of an arch, or if clusters are nested circles. Spectral clustering performs measurements for given data points by calculating their pairwise similarities with chosen similarity function. The similarity function is symmetric and non-negative. This research uses Euclidean distance. This results in a similarity matrix which is used in an unnormalized or a normalised spectral clustering Ng et al. (2001).

Euclidean distance as a similarity function is expressed

$$\text{Similarity}\ (x_i, x_j) = \frac{1}{1 + \|x_i - x_j\|}$$

where $x_i$ and $x_j$ are data points.

### 3.8 OPTICS

Optics Clustering (Ordering Points To Identify Cluster Structure) was developed to address DBSCAN's weakness when data has varying density. OPTICS does this by linearly ordering dataset points, and points which are spatially closest become neighbours in a density-based representation called the reachability plot. In this plot, every point has a reachability distance. This reachability distance defines how easily a point can be reached from other points. Clusters are then formed based on reachability distances Ankerst et al. (1999).

Reachability distance is calculated with the following function

$$r_{\text{dist}}(p, q) = \max(\text{dist}(p, q), \text{core-distance}(q))$$

where $p$ and $q$ are data points. $\text{dist}(p, q)$ is the Euclidean distance between points $p$ and $q$. core-distance is the radius within which a certain density threshold $\epsilon$ or $MinPts$ is satisfied. OPTICS in this research uses the same parameters as DBSCAN.

**Table 2** Results of K-means algorithm

| Method (parameters) | Success rate (%) | Runtime (s.) |
|---|---|---|
| K-means (algorithm = elkan) | 98.1 | 0.052 |
| K-means (algorithm = full) | 98.1 | 0.051 |

### 3.9 GMM

The GMM (Gaussian mixture model) is a finite mixture probability distribution model. GMM assumes that all data points are generated from a mixture of a finite number of Gaussian distributions. The parameters of these distributions are however unknown. Each Gaussian distribution has mean and covariance which defines its parameters, the whole GMM is built of mean vectors ($\mu$) and covariance matrices ($\sigma$). GMM uses an iterative expectation-maximisation method to estimate these parameters for distributions Rasmussen (2000). A Gaussian Mixture Model is represented by the following probability density function:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

### 3.10 Birch

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) is a hierarchical clustering algorithm. BIRCH incrementally builds a tree-like data structure (Clustering Feature tree). Tree summarises the information about the dataset and is built top-down. Algorithm recursively splits the data into subclusters. Algorithm does not use traditional distance-based split criteria like other clustering algorithms. The split conditions are based on factors like the number of points in a subcluster or the sum of squared feature values. The actual split logic is more intricate due to the CF tree structure and the desire to maintain balance in the tree. BIRCH algorithm has two main parameters: the maximum number of subclusters that can be generated from a single cluster and the threshold distance. These parameters determine the size and depth of the Clustering Feature tree Zhang et al. (1997).

## 4 Results

The results of this research were obtained by running the process mentioned in the previous chapter. Clustering was performed with a 2,3 GHz Quad-Core Intel Core i5 processor. Clustering process was considered successful, if it was able to recognise three different clusters, and clusters were formed correctly. Correct formulation of clusters was defined as follows, if Delta-E value between white and colour was

**Table 3** Results of C-means algorithm

| Method (parameters) | Success rate (%) | Runtime (s.) |
|---|---|---|
| C-means ($m = 1.0$) | 96.1 | 0.034 |
| C-means ($m = 2.0$) | 96.6 | 0.027 |
| C-means ($m = 3.0$) | 97.1 | 0.039 |
| C-means ($m = 4.0$) | 97.6 | 0.051 |
| C-means ($m = 5.0$) | 97.6 | 0.050 |
| C-means ($m = 6.0$) | 97.6 | 0.068 |

**Table 4** Results of DBSCAN algorithm

| Method (parameters) | Success rate (%) | Runtime (s.) |
|---|---|---|
| DBSCAN (eps = 2.5, min_samples = 25%) | 19.5 | 0.172 |
| DBSCAN (eps = 5.0, min_samples = 25%) | 70.1 | 0.176 |
| DBSCAN (eps = 10.0, min_samples = 25%) | 78.1 | 0.191 |
| DBSCAN (eps = 10.0, min_samples = 16.7%) | 81.6 | 0.177 |
| DBSCAN (eps = 10.0, min_samples = 33.3%) | 25.2 | 0.167 |
| DBSCAN (eps = 15.0, min_samples = 25%) | 70.1 | 0.181 |

equal or smaller than 2.0 when compared to ground-truth white - colour Delta-E (calculated in the process step 4 Fig. 4). Value 2.0 for Delta-E was selected as past research has identified it as the smallest colour difference an inexperienced human observer can notice (Han et al. 2022). This can be expressed as the following equation:

$$\Delta E_{00} = \begin{cases} \text{success}, & \text{if } \Delta E_{00} <= 2.0 \\ \text{failed}, & \text{otherwise} \end{cases}$$

Success rate for each method was calculated using the standard formula:

$$\text{success rate } = \left( \frac{\text{correctly clustered images}}{\text{total images}} \right).$$

Results of running all unsupervised clustering methods for the whole 620 images dataset are presented in the following tables. In these tables success rate describes how large share of the images were successfully clustered into three cluster were white and colour had Delta-E equal or smaller than 2.0, when compared to ground-truth. In the method column, method and its possible parameters are described, some methods were tested with multiple parameters so that best parameters and their combination was found. Runtime is the average runtime per image.

K-Means (Table 2) achieved very high success rate. Challenges it had were when density of ink colour was low (20%), specially in M & C CMYK-components. As being a hard clustering method has it advantages when it comes to recognition of colour cluster, this is because each data point must belong to only one cluster. Problems arise from datapoints which are outliers and pull centres away from their ground-truth. Two different algorithms were experimented, "full" and "elkan", there seems to be no difference between these two algorithms.

**Table 5** Results of GMM algorithm

| Method (parameters) | Success rate (%) | Runtime (s.) |
|---|---|---|
| GMM (covariance_type = full) | 99.0 | 0.085 |
| GMM (covariance_type = tied) | 97.6 | 0.071 |
| GMM (covariance_type = diag) | 99.0 | 0.071 |

C-Means (Table 3) seems to fail when colour tried to be identified had only K CMYK-component, or if density of ink is low (20%). Otherwise, it works well. Close cluster seem to make it harder for C-mean algorithm to differentiate cluster, this comes from C-mean nature of being soft clustering method. In close cluster datapoints have some grade of belonging to other than its main cluster. Outliers also impact negatively when C-mean is used. For the best results fuzziness parameters must be 4.0 or larger. Execution time although increases if fuzziness parameters is large.

DBSCAN (Table 4) usually recognises three clusters, but sometimes only two, which makes the whole clustering process to fail. In these cases DBSCAN mistakenly identifies large amount of datapoints as noise, even though they belong to a cluster. As seen from the table, DBSCAN was experimented with different esp values. DBSCAN fails throughout the dataset, but especially when intensity of colour is between 40 and 80%.

GMM (Table 5) has very good performance, failures happen in same images as with the K-means, but not in so many test cases. Challenge with GMM is that it does not explicitly model outliers, and noisy data points can influence the cluster parameters. Different covariance matrixes, which define the gaussian distributions, were tested. Best options were diagonal and full. Full covariance gives components a possibility to adopt any position and shape individually. When diagonal is used the In the diagonal covariance, the contour axes align with the coordinate axes. However, for other orientations, the eccentricities of components may differ.

BIRCH (Table 6) algorithm manages to solve clustering better than expect, as dataset is not generally considered hierarchical. As with other clustering algorithms BIRCH is also vulnerable when it comes to outliers. Also BIRCH is depend on data ordering of datapoints, this was not considered in this research. With the BIRCH methods two different parameters and their values were tested. Threshold value defines the radius of the which is used to merge samples to subclusters, and branching factor defines the maximum number of sub-

**Table 6** Results of BIRCH algorithm

| Method (parameters) | Success rate (%) | Runtime (s.) |
|---|---|---|
| BIRCH (threshold = 1.0, branching_factor = 50) | 78.5 | 0.285 |
| BIRCH (threshold = 0.5, branching_factor = 50) | 81.9 | 0.344 |
| BIRCH (threshold = 0.25, branching_factor = 50) | 85.3 | 0.351 |
| BIRCH (threshold = 0.25, branching_factor = 100) | 84.0 | 0.354 |
| BIRCH (threshold = 0.25, branching_factor = 25) | 83.5 | 0.41 |
| BIRCH (threshold = 0.1, branching_factor = 50) | 85.0 | 0.38 |

**Table 7** Results of hierarchical clustering algorithm

| Method (parameters) | Success rate (%) | Runtime (s.) |
|---|---|---|
| Hierarchical (affinity = euclidean, linkage = ward) | 97.7 | 0.537 |
| Hierarchical (affinity = euclidean, linkage = complete) | 82.9 | 0.414 |
| Hierarchical (affinity = euclidean, linkage = average) | 92.7 | 0.468 |
| Hierarchical (affinity = euclidean, linkage = single) | 84.2 | 0.417 |

**Table 8** Results of spectral clustering algorithm

| Method (parameters) | Success rate (%) | Runtime (s.) |
|---|---|---|
| Spectral (affinity = rbf) | 98.1 | 2.87 |
| Spectral (affinity = nearest_neighbour) | – | – |

**Table 9** Results of Meanshift algorithm

| Method (parameters) | Success rate (%) | Runtime (s.) |
|---|---|---|
| Meashift (quantile = 0.5) | 5.2 | 18.3 |
| Meashift (quantile = 0.75) | 0 | 17.5 |
| MeanShift (quantile = 0.4) | 5.0 | 15.6 |

clusters in each node. Even with the best parameters BIRCH fails throughout the dataset, especially when colour is green, or its intensity is 20%

Hierarchical clustering (Table 7) algorithms are sensitive to noise and outliers, as they can impact the linkage and structure of clusters, still hierarchical clustering works well. Distances between datapoints in LAB colour space can easily be calculated using euclidean distance. Usage of euclidean distance is used to allocate points into clusters, depend on their distance. In this research the hierarchical clustering affinity matrix was constructed using euclidean distance, and four different options for linkage were used. Ward linkage, which minimises the variance of the clusters when merged, performed best. With the hierarchical clustering most problematic cases are 20% M and 20% C images, and it also fail sometimes with the green colour.

Nature of the problem does not especially require spectral cluster (Table 8), sill spectral clustering achieves quite high success rate. Spectral clustering does not explicitly handle noisy data points or outliers. Outliers can affect the affinity matrix and potentially lead to the formation of unwanted clusters. When affinity matrix is computed using nearest neighbours algorithm fails to run. However, when affinity matrix is constructed using a radial basis function (RBF) kernel algorithm.

Meanshift (Table 9) algorithm is very slow on given problem. Larger problem was that Meanshift was not able to cluster data points correctly. Problems of the Meanshift might come from incorrect bandwidth value or noisy data, but even if Meanshift would work correctly it would be very slow in the problem given. Different quantile sizes were also tested, but Meanshift still was not able to recognise clusters.

As OPTICS (Table 10) is based on same base core idea as DBSCAN, it was expected to work in quite similar way. However, OPTICS was able to achieve only 70% success rate with best parameters, being also slower than DBSCAN. OPTICS hierarchical approach still seems to work better than DBSCAN.

Also, one other algorithm was experimented, Affinity Propagation, but it failed to perform in the given dataset, all runs ended to results that no clusters were detected because affinity propagation did not converge.

The following Fig. 7. shows some of the images where clustering failed in all best algorithms. In the image a) colour intensity is CMYK(0,0,0,0.6) gray, b) & c) (1.0,1.0,0,0) green and d) (0,0,0.2,0) magenta. In the image colour is dark in clustering results two main clusters instead of three. In b) & c) ambient lighting is challenging and its too hard for clustering algorithms to recognise green colour. Finally in

**Table 10** Results of OPTICS algorithm

| Method (parameters) | Success rate (%) | Runtime (s.) |
|---|---|---|
| OPTICS (eps = 2.5, min_samples = 25%) | 17.7 | 5.12 |
| OPTICS (eps = 5.0, min_samples = 25%) | 70.0 | 24.12 |
| OPTICS (eps = 10.0, min_samples = 25%) | 17.7 | 5.12 |

**Fig. 7** Failed images in experiment one

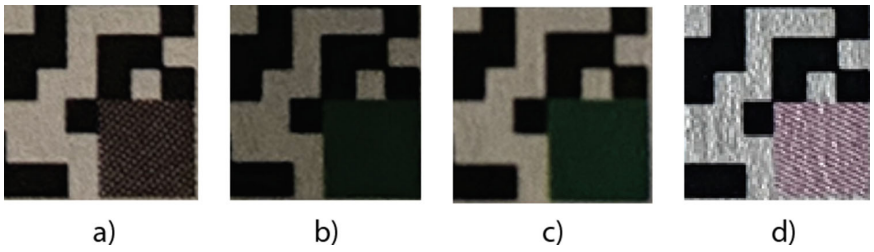



**Table 11** Results of the second experiment

| Method (parameters) | 10% intensity | 5% intensity |
|---|---|---|
| C-means (m = 3.0) | 94.8% | 83.3% |
| K-means | 96.3% | 89.3% |
| GMM (covariance = full) | 92.5% | 84.5% |
| hierarchical clustering (affinity=euclidean, linkage=ward) | 93.4% | 84.5% |
| Spectral clustering (affinity = rbf) | 97.8% | 76.2% |

the d) image there is lot of noise and print quality is low which results to incorrect results.

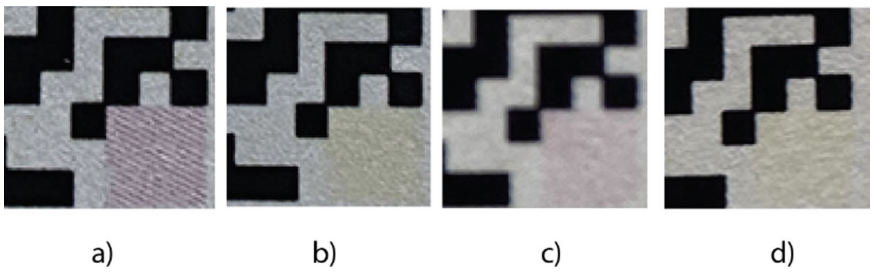
Finally second experiment was done with algorithms that had best performance, K-means, C-means ($m$ = 3.0), GMM (covariance = full), hierarchical clustering (affinity=euclidean, linkage=ward), spectral clustering (affinity = rbf) with a smaller dataset. This dataset contained colours with ink densities 0%, 5% and 10%, these images were captured in the same environments as images used in the first experiment. In total this dataset contained 230 images. Results of this experiment are shown in the Table 11.

In the second experiment success rate of all algorithms dropped, but still all of them performed well with over 90% success rate when colour intensity was 10%. But when intensity dropped to 5% only K-means achieved almost 90% success rate. Mostly challenges were with magenta and yellow colours, some of the images that failed are show in Fig. 8. In this figure images a) and b) have intensity of 10%, c) and d) have intensity of 5%

## 5 Conclusion and discussion

Results show that unsupervised clustering methods, K-means, C-means, GMM, Hierarchical clustering, Spectral clustering can be used to recognise colour differences in printed CMYK colours. Especially when difference in ink density on at least one CMYK channel is 20% or more. In these cases GMM achieves 99,0 % success rate followed by Spectral clustering and K-means. These results show that if high success rate is wanted using ink densities in 20% intervals is good way to go. Best parameter options for methods are K-means with K-means++ initialisation, C-means with fuzziness parameter 3.0, GMM with covariance type full,

**Fig. 8** Failed images in experiment two



hierarchical clustering with affinity=euclidean and Ward's linkage and Spectral clustering with RBF affinity. When ink levels drop to 10% all algorithms still have success rate over 90%. But when ink levels drop down to only 5% none of the algorithms achieves over 90% success rate.

The best algorithm based on the results of the both experiments is K-means which achieves better results than quite similar C-mean, especially in the second low ink density experiment. This is because fuzziness of the C-mean algorithm clusters some datapoints incorrectly when cluster centres are close to each other. Hard clustering which K-means uses seems to work well in lower ink densities. Used K-means algorithm uses K-means++ initialisation method. K-means++ assigns the first centroid randomly, then selects the rest of the centroids based on the maximum squared distance. K-means++ initialisation seems to work well. Incorrect clustering happens also with hierarchical and spectral clustering, some datapoints are linked into wrong cluster as they are close outliers or noise datapoints of other clusters. In this use-case datapoints form dense regions, which have spherical or elliptical shape, this plays K-means and GMM's advantage, but K-means seems to manage outliers better than GMM. While spectral clustering also works well it has challenges when ink density is low, and tries to form two clusters instead of three.

To get better results from unsupervised clustering some methods like filtering out datapoints which are outliers could be used. This would need to be done in a way which doesn't destroy datapoints which belong to some clusters, as there is limited amount of datapoints in images. Also using some pre-processing techniques, like noise filtering, colour enchantment and so might improve results. Problem might be that in this use-case there is very limited amount of data available for colour enchantment. In this problem it might be also possible to assign one or two initial locations of centroids (black & white) manually, this might lead into better results but is left for future research.

**Acknowledgements** This work was supported by Finnish Cultural Foundation's Central Ostrobothnia Regional Fund (Grant Number 25211242).

**Funding** Open Access funding provided by University of Vaasa.

**Data availability** The dataset used in this manuscript is available as Zenodo repository: 10.5281/zenodo.7749912.

## Declarations

**Conflict of interest** There are no conflicts of interest.



## References

Aarathi KS, Abraham A (2017) Vehicle color recognition using deep learning for hazy images. In: 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), pp 335–339

Abdalla A, Cen H, El-manawy A et al (2019) Infield oilseed rape images segmentation via improved unsupervised learning models combined with supreme color features. Comput Electron Agric 162:1057–1068. https://doi.org/10.1016/j.compag.2019.05.051

Ankerst M, Breunig MM, Kriegel HP et al (1999) Optics: ordering points to identify the clustering structure. ACM Sigmod Record 28(2):49–60

Banic N, Loncaric S (2018) Unsupervised learning for color constancy. pp 181–188

Bar-Haim Y, Saidel T, Yovel G (2009) The role of skin colour in face recognition. Perception 38(1):145–148

Basar S, Ali M, Ochoa-Ruiz G et al (2020) Unsupervised color image segmentation: a case of rgb histogram based k-means clustering initialization. PLoS ONE 15(10):e0240015

Bazeille S, Quidu I, Jaulin L (2012) Color-based underwater object recognition using water light attenuation. Intell Serv Robot 5(2):109–118

Bezdek JC, Ehrlich R, Full W (1984) Fcm: The fuzzy c-means clustering algorithm. Comput Geosci 10(2–3):191–203

Bo L, Ren X, Fox D (2013) Unsupervised feature learning for rgb-d based object recognition. In: Experimental robotics, Springer, pp 387–402

Bretzner L, Laptev I, Lindeberg T (2002) Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In: Proceedings of fifth IEEE international conference on automatic face gesture recognition, IEEE, pp 423–428

De la Escalera A, Armingol JM, Mata M (2003) Traffic sign recognition and analysis for intelligent vehicles. Image Vis Comput 21(3):247–258

Dresp B, Wandeto JM (2020) Unsupervised classification of cell imaging data using the quantization error in a self-organizing map. In: on Science AC, ASCE E (eds) 22nd International Conference on Artificial Intelligence ICAI 2020, American Council on Science and Education, Las Vegas, United States, CSCI 2020 Book of Abstracts, https://hal.archives-ouvertes.fr/hal-02913378

Du EY, Chang CI, Thouin PD (2004) Unsupervised approach to color video thresholding. Opt Eng 43(2):282–289

Elkan C (2003) Using the triangle inequality to accelerate k-means. In: Proceedings of the 20th international conference on Machine Learning (ICML-03), pp 147–153

Ester M, Kriegel HP, Sander J, et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd, pp 226–231

Feng L, Jiang D, Zhang A, et al (2019) Color recognition for rubik's cube robot. In: 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), IEEE, pp 269–274

Gao XW, Podladchikova L, Shaposhnikov D et al (2006) Recognition of traffic signs based on their colour and shape features extracted

using human vision models. J Vis Commun Image Represent 17(4):675–685
Gerke M, Xiao E (2014) Fusion of airborne laserscanning point clouds and images for supervised and unsupervised scene classification. ISPRS J Photogr Remote Sens 87:78–92. https://doi.org/10.1016/j.isprsjprs.2013.10.011
Giri K, Biswas TK (2020) Determining optimal epsilon (eps) on dbscan using empty circles. In: International Conference on Artificial Intelligence and Sustainable Engineering: Select Proceedings of AISE 2020, Vol 1, Springer Nature, p 265
Gong J, Jiang Y, Xiong G, et al (2010) The recognition and tracking of traffic lights based on color segmentation and camshift for intelligent vehicles. In: 2010 IEEE Intelligent Vehicles Symposium, IEEE, pp 431–435
Gonzalez TF (1985) Clustering to minimize the maximum intercluster distance. Theor Comput Sci 38:293–306
Han A, Kim J, Ahn J (2022) Color trend analysis using machine learning with fashion collection images. Clothing Textiles Res J 40(4):308–324
Hartigan JA, Wong MA (1979) Algorithm as 136: A k-means clustering algorithm. J R Stat Soc Ser C (Appl Stat) 28(1):100–108
Hurlbert A, Ling Y (2012) Understanding colour perception and preference. In: Colour design. Elsevier, p 129–157
Isohanni J (2022) Use of functional ink in a smart tag for fast-moving consumer goods industry. J Pack Technol Res 6(3):187–198
Isohanni J (2023) Qr-code dataset, with colour embed inside
Jhawar J (2016) Orange sorting by applying pattern recognition on colour image. Proc Comput Sci 78:691–697
Kang J, Ji Z (2010) Dental plaque quantification using mean-shift-based image segmentation. In: 2010 International Symposium on Computer, Communication, Control and Automation (3CA), IEEE, pp 470–473
Kao WC, Wang SH, Che WH, et al (2006) Designing image processing pipeline for color imaging systems. In: 2006 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE
Koubaroulis D, Matas J, Kittler J, et al (2002) Evaluating colour-based object recognition algorithms using the soil-47 database. In: Asian Conference on Computer Vision
Kriegel HP, Kröger P, Sander J et al (2011) Density-based clustering. Wiley Interdiscip Rev Data Min Knowl Discov 1(3):231–240
Kuo CFJ, Shih CY, Kao CY et al (2005) Color and pattern analysis of printed fabric by an unsupervised clustering method. Textile Res J 75(1):9–12
Lloyd S (1982) Least squares quantization in pcm. IEEE Trans Inf Theory 28(2):129–137. https://doi.org/10.1109/TIT.1982.1056489
Luo MR, Cui G, Rigg B (2001) The development of the cie 2000 colour-difference formula: Ciede 2000. Color Res Appl 26(5):340–350. https://doi.org/10.1002/col.1049
MacQueen J, et al (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, pp 281–297
Mao B, Li B (2019) Building façade semantic segmentation based on k-means classification and graph analysis. Arab J Geosci 12(7):1–9
Miyamoto S, Abe R, Endo Y, et al (2015) Ward method of hierarchical clustering for non-Euclidean similarity measures. In: 2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR), IEEE, pp 60–63
Ng A, Jordan M, Weiss Y (2001) On spectral clustering: analysis and an algorithm. Adv Neural Inf Process Syst 14
Nielsen F (2016) Hierarchical clustering. Springer International Publishing, Cham, pp 195–211
Pedregosa F, Varoquaux G, Gramfort A et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
Rabie T (2017) Training-less color object recognition for autonomous robotics. Inf Sci 418:218–241
Rasmussen C (2000) The infinite gaussian mixture model. Adv Neural Inf Process Syst
Reddy EK (2021) Clustering techniques in data mining: A comparative analysis. Research issues on datamining, pp 95–101
Riri H, Elmoutaouakkil A, Beni-Hssane A et al (2016) Classification and recognition of dental images using a decisional tree. In: 2016 13th International Conference on Computer Graphics. Imaging and Visualization (CGiV), IEEE, pp 390–393
Vishnuvarthanan G, Rajasekaran MP, Subbaraj P et al (2016) An unsupervised learning method with a clustering approach for tumor identification and tissue segmentation in magnetic resonance brain images. Appl Soft Comput 38:190–212
Wang Z, Zhuang Z, Liu Y et al (2021) Color classification and texture recognition system of solid wood panels. Forests 12(9):1154. https://doi.org/10.3390/f12091154
Wu KL, Yang MS (2007) Mean shift-based clustering. Pattern Recogn 40(11):3035–3052
Xu G, Li X, Lei B et al (2018) Unsupervised color image segmentation with color-alone feature using region growing pulse coupled neural network. Neurocomputing 306:1–16
Xu X, Ester M, Kriegel HP, et al (1998) A distribution-based clustering algorithm for mining in large spatial databases. In: Proceedings 14th International Conference on Data Engineering, IEEE, pp 324–331
Yang G, Li H, Zhang L, et al (2010) Research on a skin color detection algorithm based on self-adaptive skin color model. In: 2010 International Conference on Communications and Intelligence Information Security, IEEE, pp 266–270
Yavuz Z, Köse C (2017) Blood vessel extraction in color retinal fundus images with enhancement filtering and unsupervised classification. J Healthc Eng 2017:4897258. https://doi.org/10.1155/2017/4897258
Zamir SW, Arora A, Khan S, et al (2021) Learning digital camera pipeline for extreme low-light imaging
Zhang S, Huang W, Zhang C (2019) Three-channel convolutional neural networks for vegetable leaf disease recognition. Cogn Syst Res 53:31–41
Zhang T, Ramakrishnan R, Livny M (1997) Birch: a new data clustering algorithm and its applications. Data Min Knowl Discov 1:141–182
Zhao H, Qi Z (2010) Hierarchical agglomerative clustering with ordering constraints. In: 2010 Third International Conference on Knowledge Discovery and Data Mining, IEEE, pp 195–199
Zhou K, Fu C, Yang S (2014) Fuzziness parameter selection in fuzzy c-means: the perspective of cluster validation. Sci China Inf Sci 57:1–8
Zhu S, Liu L (2006) Traffic sign recognition based on color standardization. In: 2006 IEEE International Conference on Information Acquisition, IEEE, pp 951–955

# Using convolutional neural networks to classify subtle colour differences

Jari Isohanni

University of Vaasa, Digital Economy, Wolffintie 34, Vaasa, 65200, Finland.

Contributing authors: x2603813@student.uwasa.fi;

**Abstract**

Convolutional neural networks (CNNs) have proven to have high accuracy in image classification tasks. In recent years, their image classification performance has significantly improved. This research compares several of the most popular CNNs for color classification tasks. The problem presented is similar but different than the image classification task. Image classification involves a large amount of high-level feature information, while color classification involves only low-level information. This research applies standard versions of each architecture, trains the model and evaluates its accuracy. The purpose of the CNN is to identify which color has been printed on the paper. Mobile phone-captured images are first preprocessed by applying autoleveling, and then the difference between the color and paper white images is calculated. These different images are stored as a dataset. The colors used in this research are CMYK colors, and their intensity differs in the first dataset in 20 % of the steps. The second dataset uses smaller differences of 10 % and 5 %. Most architectures achieved high accuracy when the colour difference was 10 % or greater, and DenseNet was able to correctly classify 100 % of the images. When the colour difference decreased to less than 10 %, the accuracy of most models decreased significantly. ResNet, as the best architecture, achieved 95 % accuracy. ResNet only had problems with very low-intensity magenta and yellow colours. The achieved accuracy is very high, as differences in images are very subtle. The residual connections of ResNet, which help the model learn in an incremental way, and skip connections, which facilitate the reuse of features, allow ResNet to outperform other architectures. However, ResNet, like many other architectures, is prone to overfitting. This research provides a baseline for using CNN colour classification, but more research is needed, especially regarding the performance of CNNs, for example, via ablation studies.

Different fine-tuning and optimization methods, as well as more advanced preprocessing methods, should also be considered in future research.

**Keywords:** machine vision, colour difference, printed colours, convolutional neural networks

# 1 Introduction

Human eyes are sensitive to subtle color differences; for instance, we can distinguish whether a wall is painted in berry or navy hues. It is commonly acknowledged that women perceive colors differently than men. This phenomenon has been widely studied; for example, Bimler et al. Bimler et al (2004) reported that colour differences can carry significant information. IIn industries such as food and cosmetics, slight color variations can indicate quality, freshness, or differences in colors, which might raise serious safety concerns. Color recognition is also a common topic in computer vision research and development. Typically, color recognition is a part of a larger solution. In these solutions, computer vision is employed to classify, identify or recognize items. In addition, colour is a feature that can be used in the process of separation. The applications of computer vision and color recognition span various fields, including agriculture (Jhawar (2016),Lamb and Chuah (2018)), object recognition (Albani et al (2017),Nalinipriya et al (2018)), environmental protection Zhao et al (2020) medical imagining Zhu et al (2017) and predictive maintenance Ahmed and Nandi (2021).

Color recognition is also an integral part of the printing industry Luo and Zhang (2003). With the development of functional inks, colour recognition has played a role in printing nonelectronic sensors Hakola et al (2021). These sensors can be applied to fast-moving consumer goods or other product labels, where sensors react to environmental changes through color alteration Isohanni (2022). The detection of this color change via computer vision enables these sensors to function as cost-efficient IoT sensors.

This research focuses on experimenting with some of the most popular convolutional neural network (CNN) models in color classification. The results provide knowledge to researchers and developers interested in employing color classification in their projects. Such a comparative analysis has not been conducted previously, as color classification is typically studied as part of a larger recognition task. Notably, there has been very little research on using CNNs to recognize colors in printed sources. However, CNNs have proven to be powerful tools for various different recognition tasks, leading to improved product quality and efficiency in various applications.

In this research, popular convolutional neural networks (CNNs) are used to recognize color differences in printed sources. The research question this research aims to answer is "How can CNNs be used to recognize small color differences in printed sources, especially when a low-cost smartphone is used

as a camera?" The key contributions of this work can be summarized as an approach to recognizing small color changes in printed sources. This helps when creating approaches that can be applied across various industries for quality control, safety, and precision.

This research is structured as follows: Chapter 2 starts with relevant past research. Chapter 3 contains an overview of the system. Chapter 4 provides an overview of the convolutional neural networks used and their comparison based on past research. Section 5 describes the experiments used, and in Section 6, the results of the experiments are shown. Finally, Section 7 discusses the conclusions of the research.

## 2 Past research

Color recognition, especially in regard to the recognition of slight color differences, is not a commonly studied topic. Past research like (Lai and Westland (2020),Senthilkumar (2010),Wu et al (2019),Vidal-Calleja et al (2014),Arsenovic et al (2019), Anandhakrishnan and Jaisakthi (2022)) relate loosely to approach presented in this research. However, they examined using colors in artificial neural networks in different contexts or as part of wider object recognition. The development of convolutional neural networks (CNNs) and their ability to be used in closely related research (Apriyanti et al (2021),Büyükarıkan and Ülker (2022), Engilberge et al (2017),Atha and Jahanshahi (2018a), Boulent et al (2019), Tiwari (2018), Muhammad et al (2018),Zhang et al (2018), Kagaya et al (2014),Przybyło and Jabłoński (2019)). This research focuses on using CCNs to identify slight color differences. This research takes input from past research when designing an architecture proposed for slight color difference recognition.

Closely related past research has developed CNN architectures, which perform well in problems related to this research. Past research shows that CNNs are well suited for color recognition tasks. This is due to the CNN's ability to automatically learn hierarchical features from images. CNNs can capture intricate patterns and variations in color, making them effective at discerning subtle differences. The capability of a CNN depends on its architecture. One of the main architectural design-related questions is how wide and/or deep architecture should be. A wide neural network is characterized by having a substantial number of neurons within each of its layers. A network is considered "wider" when there is a greater number of neurons in a given layer. Wider networks are employed for tasks demanding extensive feature engineering, such as image classification and natural language processing. A deep neural network is characterized by having numerous layers. The greater the number of layers in a network is, the deeper the network becomes. Deep networks have applications in tasks that necessitate a high degree of abstraction, such as speech recognition and predictive analytics.

Some of the closely related research has focused on the inspection of items or structures via computer vision and the use of CNNs to identify defects.

These use cases have come from agriculture and civil engineering. Detection of defects is crucial, for example, in the prediction of structural failures or the correct time for pest control.

Soukup and Huber-Mörk proposed a CNN architecture that had two convolutional and pooling layers and a final fully connected layer. They were able to recognize defects in steel rails, but future research should use deeper CNNs. The findings of Soukup's and Huber-Mörk's research are similar, as defects in rails are small, and differences in nondefect and defect images are small. However, their research used a special setup for image capture. Soukup and Huber-Mörk (2014)

Fuentes et al. researched the use of various CNNs to identify defects in tomato plants. They used the Faster Region-based Convolutional Neural Network (Faster R-CNN), Region-based Fully Convolutional Network (R-FCN), and Single Shot Multibox Detector (SSD) together with deep feature extractors such as VGG net and Residual Network (ResNet). Authors state that in their case plain networks perform better than deeper networks. Their research has quite closely the same setup as in this research. Changes in the tomatoes were slight, and they did not use any special image capture setup. Fuentes et al (2017)

Zhang et al. proposed the automatic detection of road cracks and did not use any special image capture setups, such as normal smartphones. Their problems are quite close to one in this research, as cracks have low contrast with the surrounding pavement. Their approach used four convolutional layers, five max-pooling layers, and two fully connected layers. With this approach, they achieved almost 90% precision. Zhang et al (2019a)

Mohanty et al. used the common CNN architectures AlexNet and GoogLeNet to identify plant diseases. Their research dataset was collected by using normal smartphones, and GoogLeNet consistently outperformed AlexNet. Additionally, their results show that using color images instead of grayscale images yields better results. Mohanty et al (2016)

Chen and Jahanshahi proposed a deep learning framework based on a convolutional neural network (CNN) and a naive Bayes data fusion scheme, called NB-CNN, to analyse individual video frames for crack detection. Their use also involves very small and difficult-to-detect changes in metallic surfaces. Their approach had four convolutional and pooling layers with two fully connected layers. The result of their approach was that 99% of the defects were recognized successfully. Chen and Jahanshahi (2017)

Kumar et al. used CNNs to identify defects in sewer pipes. The defects they looked for did not significantly differ from those of pipes under decent conditions. Their approach had two convolutional layers and two connected layers. Their approach achieved 86% accuracy. Kumar et al (2018)

Past research has shown that various CNN architectures can be used to identify slight differences in images. Some of them are very deep. Some of them are standard architectures, and some of them are specially developed for certain use cases. As shown in Table 1, past research has used both existing

CNNs and custom CNNs. However, there has been very little research on the use of CNNs to recognize colors in printed sources.

**Table 1** Related past research

| Article | CNN architecture |
|---|---|
| Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection Atha and Jahanshahi (2018b) | VGG, GoogLeNet, ResNet, AlexNet, ZFNet |
| Solving Current Limitations of Deep Learning Based Approaches for Plant Disease Detection Arsenovic et al (2019) | AlexNet, VGG, Inception version 3, DenseNet, ResNet152 |
| Using convolutional neural network models illumination estimation according to light colors Büyükarıkan and Ülker (2022) | VGG |
| Automated color detection in orchids using color labels and deep learning Apriyanti et al (2021) | VGG16, Inception, Resnet50, Xception, Nasnet |
| Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks Cha et al (2017) | Custom |
| Convolutional Neural Networks for the Automatic Identification of Plant Diseases Boulent et al (2019) | ResNet-101,DenseNet-121,VGG-16,Inception-V3, SqueezeNet |
| Color representation in deep neural networks Engilberge et al (2017) | VGG-19, AlexNet |
| Rock images classification by using deep convolution neural network Cheng and Guo (2017) | Custom |
| Deep Convolutional Neural Networks for image based tomato leaf disease detection Anandhakrishnan and Jaisakthi (2022) | Custom |
| Three-channel convolutional neural networks for vegetable leaf disease recognition Zhang et al (2019a) | Custom |
| Early fire detection using convolutional neural networks during surveillance for effective disaster management Muhammad et al (2018) | Custom AlexNet |
| Using Deep Convolutional Neural Network for oak acorn viability recognition based on color images of their sections Przybyło and Jabłoński (2019) | CNN-F-16, CNN-F-13, CNN-F-9 |
| Vehicle color recognition using Multiple-Layer Feature Representations of lightweight convolutional neural network Zhang et al (2018) | Custom |
| Color for object recognition: Hue and chroma sensitivity in the deep features of convolutional neural networks Flachot and Gegenfurtner (2021) | AlexNet, VGG-16 and VGG-19 |
| Color for object recognition: Hue and chroma sensitivity in the deep features of convolutional neural networks Flachot and Gegenfurtner (2021) | AlexNet, VGG-16 and VGG-19 |

# 3 System overview

The process as a whole is described in this section and its subsections. Figure 1 shows the process of creating the final dataset used in the research. Before splitting the dataset into training and validation datasets, image autoleveling, extraction of color areas, and calculation of difference images are performed. The difference image is used in this research because the objective of the research is to determine whether CNNs can correctly identify colour differences from paperwhites. The difference image is also feasible for use because the two areas that are compared can be made equal in size. Another option would be to merge two color images together. Then, the merged image could be used as a multichannel input to the convolutional neural network. However, this could lead to decision-making where individual pixel differences might play an important role. The use of multichannel images is left for future research.

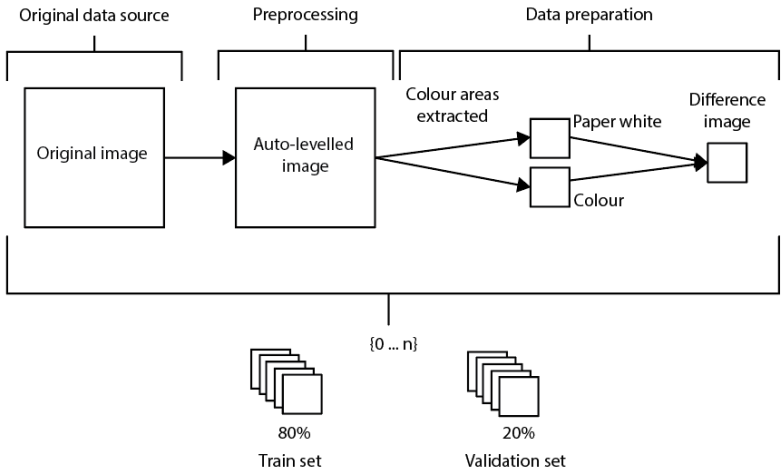



**Fig. 1** Dataset creation process

## 3.1 Colour dataset

This research uses two datasets. The first dataset is an open dataset that contains QR codes with different color areas (Figure Isohanni (2023).). The second dataset is a custom dataset created for this research with the same approach as the first dataset. Different color areas in QR codes are a) paper-white, b) 100K black, and c) color. From these are paper-white and black are used for auto-leveling and paper-white and colour to form the final dataset.

The second dataset is a custom dataset created for this research with the same approach as the first dataset. The different coloured areas in the QR codes are a) paper-white, b) 100K black, and c) coloured. Paper-white and black are used for autoleveling, and paper-white and color are used to form the final dataset. In the first dataset, colors vary in all CMYK colors, combinations,

and saturations. The ink saturation in these images varies between 20% and 100% in steps of 20%. The second dataset, which was captured only for this research, extends the first dataset by colors with lower ink saturations of 0%, 5%, and 10%. Examples of the colors in the first dataset are shown in Figure 2.

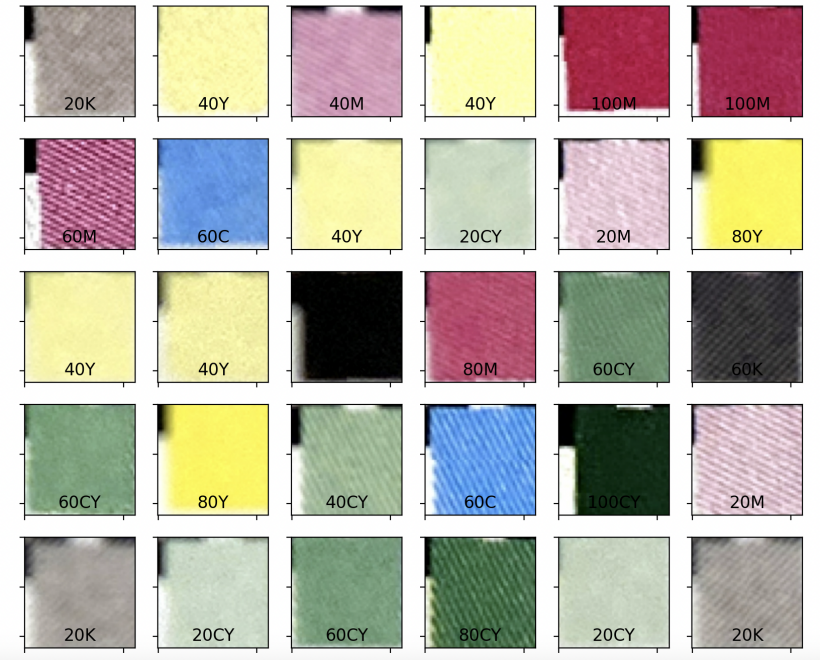



**Fig. 2** Colour examples, dataset 1.

Images in both datasets were captured with two standard mobile devices (different versions of the iPhone) in normal living/office ambient light environments. In practice, this means that in the dataset, images with different colour temperatures, ambient light and images have been captured from different distances and with different camera settings. The purpose of the dataset is to reflect various situations that might arise every day. The different colours are grouped into folders, where the folder names define the colour combination and saturation of the coloured areas.

As can be seen from the dataset one (Figure 2), all colours are quite clear. However, it can be seen that the samples were printed with a laserjet, as stripes typical of laser printing appear when camera autofocus is used in a specific setting.

In the Figure 3, colour differences are no longer clear. Camera settings, such as autobalancing and adapting to capture the environment, easily destroy some of the small colour difference information while improving the overall image. Yellow colour information (5Y and 10Y) is sometimes almost destroyed. This can be expected to make classification of images very challenging.

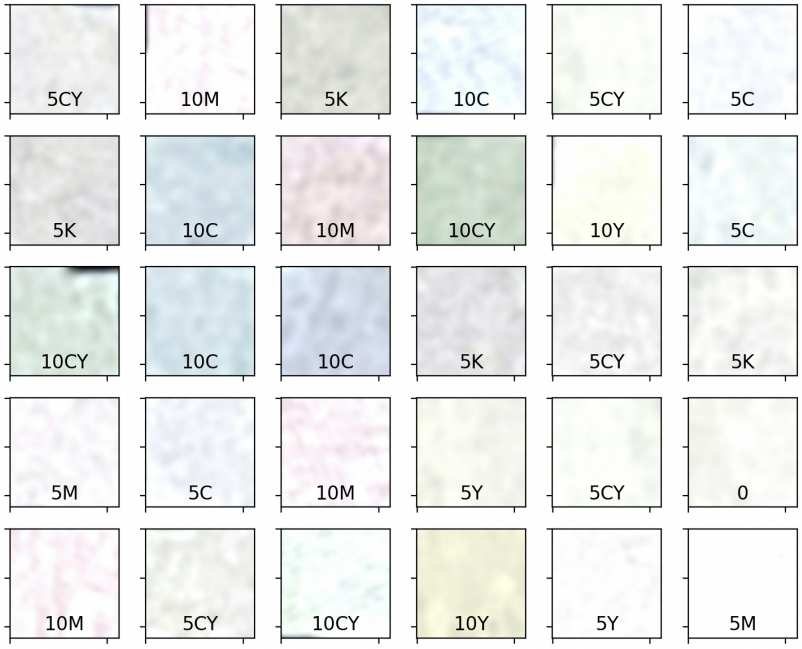



**Fig. 3** Colour examples, dataset 2.

## 3.2 Preprocessing

Preprocessing of the images in the dataset is performed by identifying the darkest and whitest areas, marked with red and green rectangles in the image (Figure 4. red rectangle), and the color area (Figure 4. yellow rectangle). These areas work as input for autocontrast. Auto-contrast is a mapping function that stretches the pixel intensities in the image to cover the entire range. In the case of RGB images, this range is [0 ... 255] on each channel. The darkest areas will be mapped to the minimum intensity value (e.g. 0). The whitest areas will be mapped to the maximum intensity value (e.g. 255). Taylor (2003) Output image b) in Figure 4, is expected to have improved contrast and color balance. Auto contrast adjustment is calculated with the following equation:

$$I' = \frac{I - I_{\min}}{I_{\max} - I_{\min}} \times 255,$$

where $I_{\max}$ is the brightness value in the image and $I_{\min}$ is the darkest value in the image.

Preprocessing of the whole image is concluded with Gaussian blurring of the image to eliminate noise. Gaussian blurring is performed by applying a Gaussian filter to the image; this is a convolution operation involving a Gaussian kernel. In this research, the kernel used is 5x5 pixels in size. This makes edges and fine details less pronounced. These edges especially appear in laser-printed images. Chauhan (2018)

After blurring two areas are extracted from the image, the white area (Figure 4. red rectangle) and the color area (Figure 4. green rectangle). These color areas are resized to 256 x 256 px so that their size is close to the size used

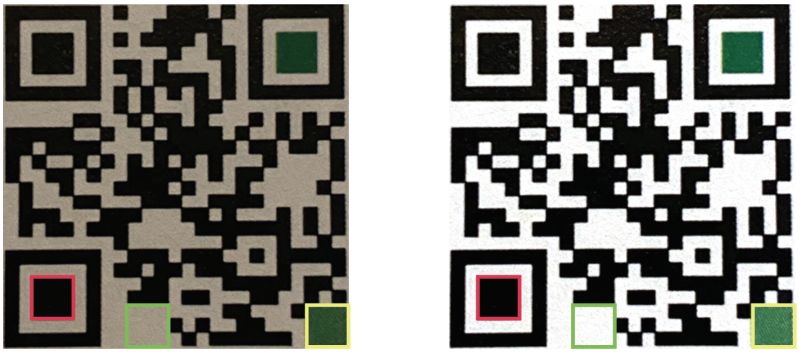



**Fig. 4** Different colour areas of the QR-Code and results of the auto-levelling.

by CNNs as input. Then, as the last step, the difference image between white and colour is calculated. The difference image is calculated pixel by pixel. The formula for calculating the difference image is:

$$D(x, y) = |I1(x, y) - I2(x, y)|$$

where $D(x, y)$ represents the difference value for the pixel at position (x, y) in the difference image. $I1(x, y)$ is the color value of the pixel at position (x, y) in the first RGB image. $I2(x, y)$ is the color value of the pixel at position (x, y) in the second RGB image. The absolute value is used to ensure that the difference is a positive value, as the actual difference could be negative if $I1(x, y)$ is smaller than $I2(x, y)$. Taking the absolute value ensures that the final image reflects the magnitude of the difference, not its direction.

This difference image is stored as an RGB image in a folder, where the folder name represents the label of the image.

## 3.3 Transfer learning

As this research has two different datasets, which are both quite similar but the classes represent different colors, the CNN is first trained with the larger dataset. Then, the knowledge is transferred to a smaller model and trained with a smaller dataset.

Transfer learning (Figure 5) is a pivotal technique that leverages preexisting knowledge to enhance the performance of deep learning models. Transfer learning involves reusing neural network architectures and their learned representations from one task and adapting them for a new, related task. This process significantly reduces the need for extensive labelled data and extensive training resources. Pan and Yang (2009)

Computer vision tasks, such as image classification and object detection, often demand large datasets and significant computational power for training deep neural networks from scratch. Transfer learning mitigates these challenges by allowing training to start from pretrained models. These models learn features from larger, usually closely related datasets. Some past studies that have successfully used transfer learning in computer vision-related tasks

include those by Ravishankar et al (2016), Wu et al (2018), Varde et al (2023) and Kentsch et al (2020).

The keys to successful transfer learning lies in choosing the appropriate pretrained model and understanding how to adapt it effectively. Challenges regarding the use of transfer learning are related to negative transfers Brodzicki et al (2020). Negative transfer can occur, for example, if the source and target domains are too dissimilar, if the source data are biased or unrepresentative of the target data, or if the representation learned in the source task may not be suitable for the target task. Li et al (2022)

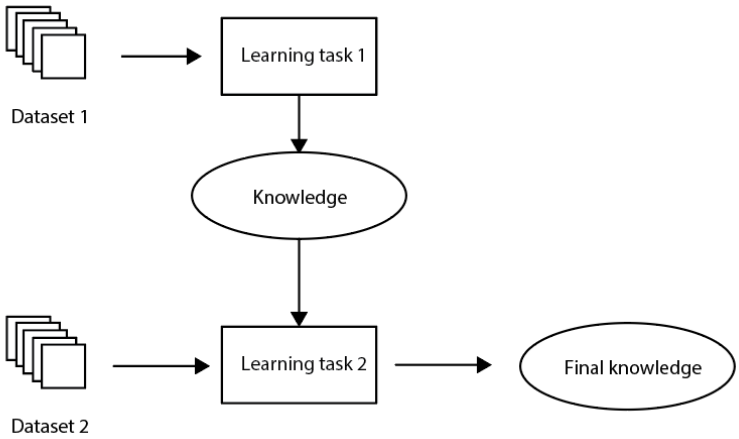



**Fig. 5** Example of transfer learning, adopted from Pan and Yang (2009)

# 4 Used CNN architectures

Convolutional neural networks (CNNs) are a class of deep learning models designed for processing structured grid data. All CNNs are built from the same elements, convolutional layers, activation functions, pooling operations, flattening.

The core building block of a CNN is the convolutional layer. Convolutional operations involve sliding a set of filters (kernels) over the input data, extracting local features, and creating feature maps. These layers capture hierarchical features through the network. Alzubaidi et al (2021)

Different activation functions have been developed as part of CNN research. Currently, nonlinear activations, such as rectified linear units (ReLUs), are the most efficient. They are applied after convolutional operations to introduce nonlinearity and enable the model to learn complex patterns. Alzubaidi et al (2021)

Pooling layers (e.g., max pooling) downsample the spatial dimensions of the feature maps, reducing computational complexity and providing a form of translation invariance. Pooling layers help retain the most important information while discarding less relevant details. Alzubaidi et al (2021)

After the convolutional and pooling layers, the spatial information is flattened into a vector, and fully connected layers are employed. These layers connect every neuron to every neuron in the subsequent layer, enabling the model to learn high-level abstractions. Alzubaidi et al (2021)

The different CNNs that have been developed in the following subsections provide an overview of the most important ones in relation to this research.

## 4.1 AlexNet and ZFNet

AlexNet (Figure 6) is a large, deep convolutional neural network. It has 60 million parameters and 500,000 neurons. The architecture of AlexNet consists of five convolutional layers. Some of the convolutional layers are followed by maxpooling layers, and finally, the architecture ends with a 1000-way softmax layer. AlexNet uses a regularization method to reduce overfitting. ZFNet is an improved version of AlexNet. ZFNet adjusts the filter sizes using smaller filters (7x7 for the first layer and 3x3 for the subsequent layers). This modification aims to capture finer details in the images. The larger stride of ZFNet in the first layer helps reduce the spatial dimensions more quickly. Krizhevsky et al (2017); Zeiler and Fergus (2014)

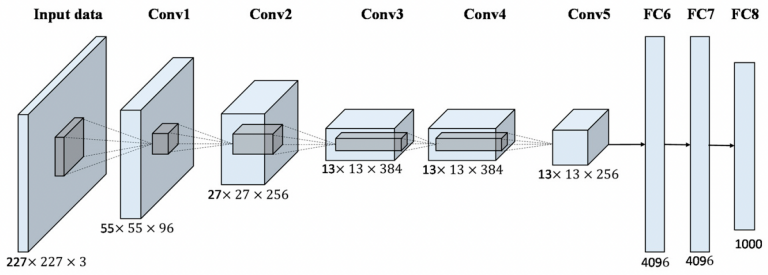



**Fig. 6** AlexNet architecture Han et al (2017).

## 4.2 VGG

VGG (Visual Geometry Group), presented in Figure 7 has 16-19 layers. VGG is a convolutional neural network (CNN) architecture designed for image classification tasks. It was developed by researchers at the University of Oxford's Visual Geometry Group. VGGs come in different versions with varying depths. For example, VGG16 has 16 layers, and VGG19 has 19 layers. The number represents how deep the network is. More layers can capture more complex features but require more computations. Simonyan and Zisserman (2014) One of the key features of VGG is its simplicity and depth. Unlike some other CNN architectures, VGG uses a straightforward structure.

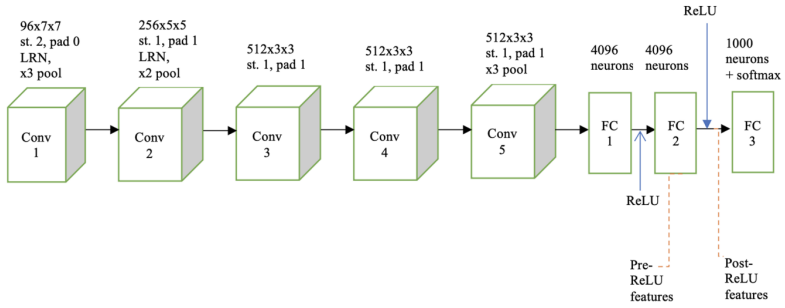



**Fig. 7** VGG architecture Paul et al (2020).

## 4.3 ResNet

ResNet (Residual Network (Figure 8)) iis a 152-layer architecture that has been developed to maximize network depth while limiting complexity. ResNet introduces "residual blocks," which allow information to bypass one or more layers, creating shortcut connections. These connections help gradient flow more effectively during training, enabling the training of extremely deep networks. He et al (2016). Being a very deep network, ResNet outperforms many networks, but it might be complex to train and use considerable amounts of memory.

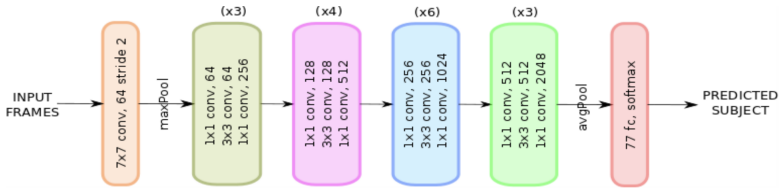



**Fig. 8** ResNet architecture He et al (2016)

## 4.4 GoogLeNet

GoogLeNet (Figure 9) is a deep convolutional neural network (CNN) architecture with 22 layers. It is known for its efficiency and performance in image recognition tasks. The key innovation in GoogLeNet is the use of "Inception modules," which are blocks of layers that perform multiple types of convolutions simultaneously, allowing the network to capture diverse features at different scales. This architecture helps reduce computational complexity while maintaining high accuracy. Szegedy et al (2015) The strength of the GoogLeNet is that it has increased depth and width without sacrificing computational efficiency.

## 4.5 DenseNet

DenseNet (Figure 10), or Densely Connected Convolutional Network, is a deep learning architecture that maximizes information flow between layers. Unlike

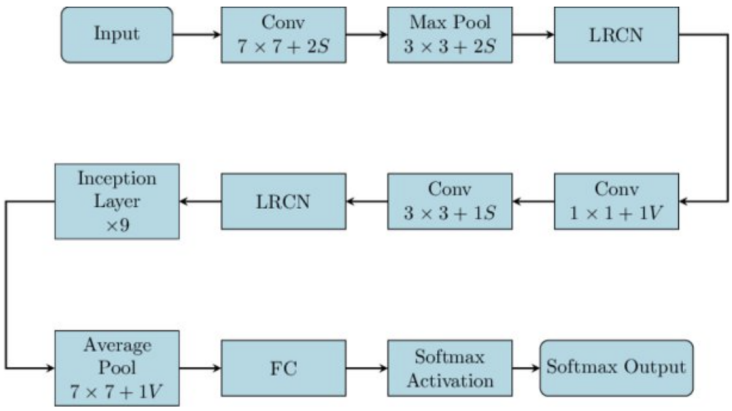



**Fig. 9** Simplified GoogLeNet Kishore and Singh (2015)

traditional networks, DenseNet connects each layer to every other layer in a dense manner. This dense connectivity enables efficient feature reuse and gradient propagation, resulting in highly efficient and accurate models with fewer parameters. used in a comparison experiment to evaluate different training strategies. Zhu and Newsam (2017)

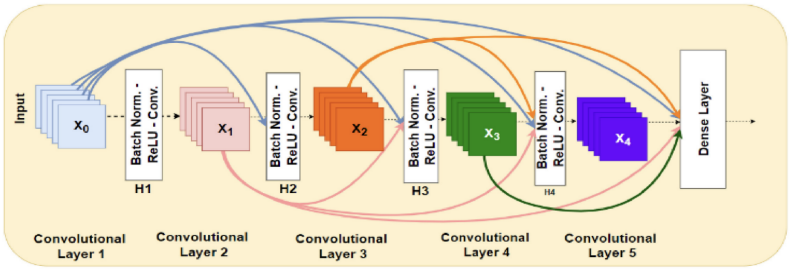



**Fig. 10** DenseNet architecture Kim and Jang (2023)

## 4.6 EfficientNet

The EfficientNet (Figure 11) models are designed to achieve state-of-the-art performance in terms of accuracy while being computationally efficient, making them suitable for a wide range of applications, including devices with limited resources. EfficientNetB1, which is used in this research, specifically represents the baseline model in the EfficientNet series, with a balance between model size and performance. The "B1" indicates its relative scale within the EfficientNet family; larger variants, denoted by B2, B3, etc.,etc., have more parameters and are computationally more demanding but may offer improved accuracy. EfficientNet architectures are characterized by a compound scaling approach, where the model's depth, width, and resolution are scaled together to find an optimal balance. This scaling strategy allows EfficientNet models to achieve

better performance than traditional architectures with similar computational costs. Tan and Le (2019)

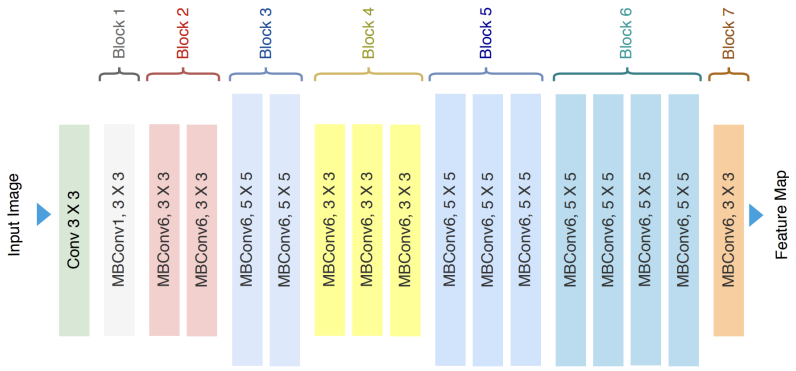



**Fig. 11** EfficientNet architecture Ahmed and Sabab (2022)

## 4.7 Comparison of the algorithms

Summary of used architectures, their layers, and amount of trainable parameters is show in the Table 2.

**Table 2** Summary of used CCN architectures

| Architecture | Year | Depth | Layers | Trainable Param. |
|---|---|---|---|---|
| AlexNet | 2012 | Deep | 8 | ~60 million |
| VGG-16 | 2014 | Very Deep | 16-19 | ~138 million |
| GoogLeNet | 2014 | Deep | 22 | ~6.7 million |
| ResNet | 2015 | Very Deep | 50-152 | ~25.6 million |
| ZFNet | 2013 | Deep | 8 | ~62 million |
| DenseNet | 2017 | Very Deep | 121-169 | ~28.9 million |
| EfficientNetB1 | 2019 | Very Deep | ~66 | ~7 million |

Based on past research, several popular CNN architectures have been compared to each other. These results offer some indications and guidelines for this research.

Naseer et al. demonstrated that AlexNet outperforms LeNet, AlexNet, VGG16, ResNet-50, and Inception-V1 in the detection of lung cancer (LUNA16 dataset). The lung cancer dataset contains CT scan images with different shapes and colors that show whether the patient has lung cancer. Additionally, Naseer et al. showed that in their dataset, AlexNet works best when the SGD optimizer is used. Naseer et al (2022)

Neris et al. compared the ResNet50, AlexNet, VGG19, MobileNet, and DenseNet architectures on three different datasets, namely, the MASATI

dataset and the airplances dataset. The MobileNet and AlexNet architectures achieved the best performances on the MASATI dataset. On the airplane dataset, ResNet was the best architecture. Neris et al (2021)

Saikiaa et al. tested the performance of VGG16, VGG19, ResNet-50, and GoogLeNet-V3 on FNAC cell sample images. Their approach used only one channel, red, and thresholding methods to eliminate noise in images. In their research, fine-tuned GoogLeNet performed best. Saikia et al (2019)

In their research, Ahad et al. compared different CNNs for rice disease classification. Their use case is quite close to one that relates to this research. Rice leaves are colored mostly green, but diseases are shown as different colors in leaves, for example, brown or yellow areas. Ahad et al. used DenseNet121, Inceptionv3, MobileNetV2, ResNet101, ResNet152V, ResNet101, and Xception CNNs. Most of the models had over 95% accuracy. Ahad et al (2023)

The methods used by Maeda-Gutiérrez et al. were quite similar to those used by Ahad et al., as they focused on tomato leaves. They used AlexNet, GoogleNet, InceptionV3, ResNet18, and ResNet50. All the CNNs used had an accuracy of more than 98%, and GoogleNet was the best. Maeda-Gutiérrez et al (2020)

The performances of 21 different CNN architectures for detecting COVID-19 were evaluated byt Breve. In his research, he used a dataset that contained chest X-ray images. In some of the classification cases, the difference between positive and negative classifications was small. He showed that DenseNet169 outperformed other CNNs. Breve (2022)

Bressem et al. also used chest radiographs as their dataset. They used five different architectures, ResNet, DenseNet, VGG, SqueezeNet, Inception v4, and AlexNet, to classify images. Interestingly, shallow networks can compete with deeper and more complex CNNs. Specifically, they showed that eight-layer AlexNet can achieve the same results as 121-layer DenseNet. Bressem et al (2020)

Based on the results of past research, it can be expected that CNN architectures will perform quite equally in this use case. The deep architecture might not have better performance than the shallow architecture, but training the deep architecture requires more time and memory. Past research has shown that AlexNet, GoogLeNet, and DenseNet perform well in classification tasks that are closely related to the datasets used in this research.

# 5 Experiments

The experiments were run first with all the architectures mentioned in the previous chapters. For training, 30 epochs were used, and the total training time was measured when the training process was run with an Apple M2 processor, which had 8 cores and a total RAM of 16 Gt.

Only standard versions of each archicture were used, and optimizations regarding hyperparameters, such as momentum, learning rate, and batch size, were not used. The purpose of the experiments was to determine which of

the architectures provides the best baseline for further development. Only the needed top layers were changed to match the number of labels.

Data augmentation was used to extend the training set, and the parameters for the augmentation were as follows:

- Rotation of each image by 90 degrees
- Width shift = 0.2, the image is shifted by 0.2 fraction in horizontal directions
- Height shift = 0.2, the image is shifted by 0.2 fraction in vertical direction
- Shear range = 0.2, shears is a transformation where the image is skewed, the left edge of the image is moved up, and the right edge down.
- Zoom range = 0.2, the image is zoomed out so the image is only 20% as large as the input
- Horizontal flip, the image is flipped by its horizontal axis
- Vertical flip, the image is flipped by its vertical axis

Each CNN was trained in the first experiment with 80% of the images in the dataset, and 20% of the images were used for the validation. In total, 5051 images were used for training, and 1249 were used for validation. In the second experiment, the split between the training and validation sets was kept the same, totaling 1488 images belonging to the training set and 372 to the validation set.

All CNN models were compiled with the cross-entropy loss function and Adam optimizer. Research on other compiling options was excluded from the scope of this research. Cross-entropy is commonly used as a loss function in classification tasks. Its goal is to minimize the cross-entropy between the predicted probabilities and the true labels. In practice, this means that during the experiments, cross-entropy yields different values depending on how many of the images are correctly classified into the correct class Shore and Johnson (1981). In total, dataset one had 25 classes, and dataset two had 11 classes.

Experiments were run for both datasets: a) larger color differences and b) datasets with smaller differences. For the latter dataset, transfer learning was used. For transfer learning, the first training process was used as a pretrained model; this model contains the weights and biases from the first dataset. The model was customized by setting the trainable property of the model to false, which prevents the weights in nontrainable layers from being updated. Then, the top layers and fully connected layers were changed to matched classes in a later dataset.

Performance of the each model was evaluated with the following metrics.

- Accuracy, ratio of correctly predicted instances to the total instances:

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions}$$

Accuracy was calculated for the whole architecture and used as the main metric when architures were compared. Additionally, a confusion matrix was created for a detailed breakdown of correct and incorrect predictions for each class in a classification problem. From the confusion matrix, further metrics were derived:

- Precision (Positive Predictive Value): The ratio of correctly predicted positive observations to the total predicted positives.

$$Precision = \frac{TP}{TP + FP}$$

- Recall (Sensitivity or True Positive Rate): The ratio of correctly predicted positive observations to all actual positives.

$$Recall = \frac{TP}{TP + FN}$$

- Definition, the weighted average of precision and recall, providing a balance between the two metrics.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

Where TP is True Positives, FP is False Positives, FN is False Negatives and TN is True Negatives.

Recall, definition and F1-score were mostly used to analyze performance in more detail and to find out where each architecture makes mistakes or work well.

# 6 Results

This section shows the results of experiments performed for both datasets.

## 6.1 First dataset

Experiment 1 was conducted on a dataset with large colour differences. For this experiment, all architectures were used. The following table shows the total accuracy of each architecture. The accuracy is the final validation accuracy of the models achieved after 30 epochs of training.

**Table 3** Results of first experiment

| Architecture | avg. train time / epoch (s.) | Accuracy |
|---|---|---|
| DenseNet | 690s. | 1.00 |
| ResNet | 626s. | 0.98 |
| VGG-16 | 570s. | 0.97 |
| AlexNet | 110s. | 0.93 |
| EfficientNetB1 | 359s. | 0.90 |
| ZFNet | 340s. | 0.88 |
| GoogLeNet | 1639s. | 0.87 |

All the architectures achieve high accuracy. This means that all of them are suitable for color difference recognition in the first dataset, where color differences are larger. The results show that very deep networks perform better than deep networks. Most of these networks have a large number of layers, except for AlexNet, which has only eight layers. This is also the case for the number of epochs, for which AlexNet is the fastest to train. Based on the DenseNet experiment, RseNet, VGG-16 and AlexNet seem to be able to classify colours well, as they reach accuracies greater than 95%. All of these architectures have their own approach to creating the CNN architecture, but as seen from the following images, they can reach 100% or close accuracy during the training process. The best four architectures respond quite well to the training process, as shown in Figure 12, where their training and validation process evolution is shown.

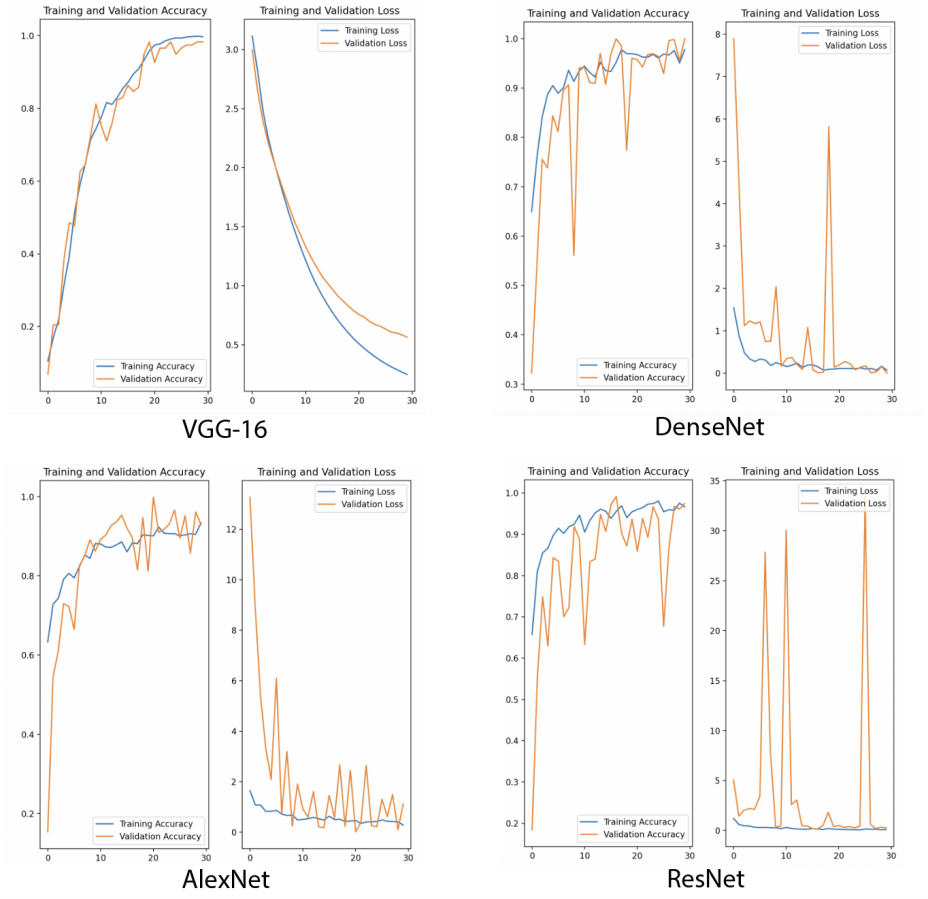



**Fig. 12** Training and validation results of best architectures

From the confusion matrixes, several conclusions can be drawn. The best architecture, DenseNet, classifies all the images correctly. Figure 13 shows the confusion caused by other architectures that achieved high accuracy. ResNet confuses shades of magenta and cyan in some cases. VGG-16 sometimes has

problems when classifying yellow shades. AlexNet mostly recognizes all classes well; some dark shades of gray are mixed, and magenta is sometimes recognized as black. Some examples of confusion are shown in the following Figure 13.

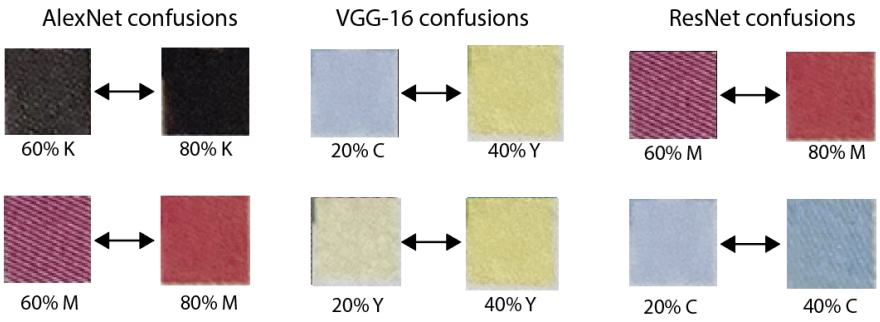



**Fig. 13** Some confusion made by best models

The first experiment yielded promising results in which selected CNNs could be used in the given use case to recognize colour differences. These differences are quite significant in dataset one; further experiments were performed with dataset two, where the differences are smaller. In the first dataset, there seems to be no indication of overfitting of the model.

## 6.2 Second dataset

For the second experiment, the four best CNN architectures were used; these CNNs obtained their weights and biases from the first dataset through transfer learning. In the second experiment, the same training and validation process was used. The Table 4 summarizes the time used per epoch and the final accuracy of the models.

**Table 4** Results of the second experiment

| CNN | avg train time / epoch (s.) | Accuracy |
|---|---|---|
| VGG-16 | 690s. | 0.34 |
| ALEXNET | 13s. | 0.47 |
| DENSENET | 201s. | 0.77 |
| RESNET | 725s. | 0.95 |

The result of the evolution of the model training process is shown in the following Figure 14

AlexNet benefits from the use of more epochs; its training accuracy improves over time, and the training loss decreases, although it seems to settle in the final epoch. Overall, the performance of AlexNet decreases significantly on the second dataset. The training and validation accuracies follow each other, so the model learns throughout the process, which also supports the use of

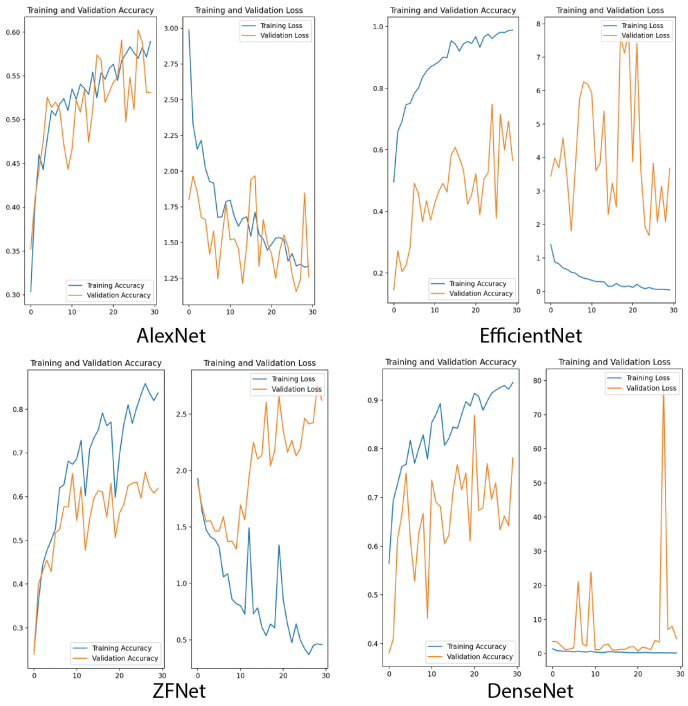



**Fig. 14** Training and validation results of second experiment

more epochs. The training accuracy of VGG-16 improves throughout the training process but not substantially (0.2... 0.3), while its training loss decreases when the model does not learn well enough. Even with more epochs, VGG-16 will not reach as high accuracy as better models. Additionally, validation loss seems not to follow training loss. DenseNet is the most accurate of all architectures in the second experiment. The model learns from the data throughout the process, and its training accuracy improves. However, although the validation accuracy increases, it behaves unpredictably. Additionally, the model validation loss does not decrease after the first epochs. As the best model, DenseNet might be the best model for use in the presented use case. ResNet achieves high accuracy in the first epochs, which shows that transfer learning works well. The accuracy of the model does not improve much after ten epochs, and the training loss seems to stabilize before ten epochs. This indicates that ResNet is prone to overfitting, the solution for this could be to implement early stopping.

According to the confusion matrix and further investigation of the per class performance of the models (Tables 5 and 6), most of the models can classify items correctly if there is no color presented (class = 0) or when there is at least 10% intensity of some color. However, when there is only 5% intensity different classes are confused.

**Table 5** AlexNet and VGG-16 results from second experiment

| | AlexNet | | | VGG-16 | | |
|---|---|---|---|---|---|---|
| Class | precision | recall | f1-score | precision | recall | f1-score |
| 0 | 0.00 | 0.00 | 0.00 | 0.48 | 0.90 | 0.63 |
| 10C | 1.00 | 1.00 | 1.00 | 0.19 | 0.75 | 0.30 |
| 10CY | 1.00 | 0.81 | 0.89 | 0.04 | 0.02 | 0.02 |
| 10K | 1.00 | 1.00 | 1.00 | 0.71 | 0.05 | 0.10 |
| 10M | 1.00 | 0.50 | 0.67 | 0.00 | 0.00 | 0.00 |
| 10Y | 0.00 | 0.00 | 0.00 | 0.30 | 0.42 | 0.35 |
| 5C | 0.20 | 1.00 | 0.33 | 0.00 | 0.00 | 0.00 |
| 5CY | 0.00 | 0.00 | 0.00 | 1.00 | 0.60 | 0.75 |
| 5K | 0.60 | 0.67 | 0.63 | 1.00 | 1.00 | 1.00 |
| 5M | 0.00 | 0.00 | 0.00 | 1.00 | 0.04 | 0.08 |
| 5Y | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

**Table 6** DenseNet and ResNet results from second experiment

| | DenseNet | | | ResNet | | |
|---|---|---|---|---|---|---|
| Class | precision | recall | f1-score | precision | recall | f1-score |
| 0 | 0.33 | 0.50 | 0.40 | 1.00 | 0.89 | 0.94 |
| 10C | 1.00 | 0.44 | 0.62 | 1.00 | 1.00 | 1.00 |
| 10CY | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 10K | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 10M | 1.00 | 1.00 | 1.00 | 0.71 | 1.00 | 0.83 |
| 10Y | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 5C | 1.00 | 0.67 | 0.80 | 1.00 | 1.00 | 1.00 |
| 5CY | 0.69 | 1.00 | 0.82 | 1.00 | 1.00 | 1.00 |
| 5K | 1.00 | 0.89 | 0.94 | 1.00 | 1.00 | 1.00 |
| 5M | 0.38 | 0.33 | 0.35 | 0.97 | 0.46 | 0.62 |
| 5Y | 0.50 | 0.67 | 0.57 | 0.86 | 1.00 | 0.92 |

Based on the results presented, the best accuracy can be achieved with the ResNet architecture, which had the best performance in both experiments. ResNet can accurately classify all colors other than very low-intensity yellow and magenta. The mistakes that ResNet makes are that 5% of the images are classified as magenta, while 10% are classified as magenta, and vice versa. Additionally, 5% yellow, which is quite challenging for the human eye to recognize, is classified as paper white.

# 7 Discussion

All the selected CNNs proved that they can be used for the recognition of color differences, especially those with large intensity differences (>10%). When the color difference decreases to only 5% or 10% (Figure 15) in some CMYK channels, most architectures struggle to classify images correctly.

The results show that different CNN architectures can be used for color difference recognition. Even worse, GoogLeNet achieved an accuracy of 88% in the first experiment. Deeper CNNs can usually identify more complex features,


**Fig. 15** Examples of small colour differences

but they are more difficult to train. Wider networks can capture more fine-grained features, but they have difficulties with high-level features. Tan and Le (2019)

In this use case, it is interesting that the depth of the network and the number of parameters are directly correlated with the accuracy of the network when the colour difference is large. All the very deep networks manage experiment one well, except for EfficientNetB1.

When the color differences decrease, the background noise from the paper and printer also becomes more significant. Another issue is that when images go through autoleveling, which should improve the colors of the image and make the model more generalized for different ambient light conditions, different colors close to paper-white might lose some important color information. In this experiment, only one of the models achieved an accuracy greater than 90% when the color difference was less than 5%. The best of the architectures was ResNet, with 95% accuracy. Most of the architectures were able to classify images correctly if the difference was 10%, but when the difference decreased to 5%, all architectures had problems. This research did not use K-fold cross-validation for validating the performance of the model, and the use of K-fold cross-validation should help to increase the reliability of the results obtained by Wong and Yeh (2019).

The best two architectures, ResNet and DenseNet, use an approach that connects layers not only to the previous and following layers. However, they have either a dense connection (DenseNet) where the architecture connects each layer to other layers separately in a feed-forward manner Zhu and Qiu (2021).For example, in ResNet, the architecture allows data to flow from other layers directly to the subsequent layers Zhang et al (2019b). These connectivity types seem to make a large difference in performance when the model is trying to classify items based on small differences.

Similar research performed in the past showed improved results when CNNs were used. In agriculture, Kumar et al. used the KNN algorithm to identify three colours (yellow, blue and green) in their research. They stated that KNN was the best method for machine learning colour classification Kumar et al (2022). In addition to artificial neural network (ANN) classifiers and support vector machine (SVM) classifiers, KNN was also used in research by Kurtulmus et al., who reported problems with fruit colors that were close to each other and could not reach an accuracy greater than 90%. Kurtulmus et al (2014) Aznan et al. also used ANNs to classify rice seeds; however, with 40 hidden

neuron models, they reached an accuracy of only approximately 70% Aznan et al (2017). The results presented in Kumar et al., Kurtulmus et al. and Aznan et al. not only show that various approaches can reach high accuracy in colour classification but also that none of them use convolutional neural networks. Based on the results presented in this research, convolutional neural networks might be used to achieve better results in agriculture and food production when items are classified based on their colors. In the context of clothes sorting, Furferi and Governi presented a method using a self-organizing feature map (SOFM) and a feed-forward backpropagation artificial neural network (FFBP ANN)-based approach to classify clothes into 85 classes. Their research was quite old but already showed only 5% mean error; they had problems mainly with beige and brown colours, which the ResNet approach presented in this research might overcome. Furferi and Governi (2008)

# 8 Conclusion

This research focuses on solving the following question: can convolutional neural networks be used to recognize small color differences in printed sources? Source data for the project contained various images, which stored different information between paper white and source color. The colors used varied among the different CMYK colors and intensities. The smallest intensity difference used was 5%. The nature of the problem is prone to overfitting, and the images that were used contain only the difference between color and paper-white, not actual high-level features; noise and small changes in images make the used models easily nongeneric. This makes it challenging for CNN models to learn from the data. When CNNs are compared in this research, making CNNs deeper or wider does not guarantee automatically better results. The results of the research show that the ResNet architecture can be used to identify small color differences. The difference between ResNet and other architectures is that ResNet implements residual connections. These connections enable the network to learn residual functions rather than directly learning the desired underlying mapping. In practice, this means that each layer can focus on learning the incremental changes needed to transform the input. The skip connections in ResNets facilitate the reuse of features from earlier layers in the network. This is advantageous for recognizing small differences because it allows the network to preserve and leverage low-level features that may be relevant for distinguishing between similar classes or categories in the data. As the best architecture, ResNet achieved 95% accuracy in the small difference experiment, but when the colour difference was larger (over 10%), ResNet's accuracy was 100%. Further research should be conducted with different ResNet modifications and versions, as well as with different training parameters (such as different optimizers). However, more research on improving the source data is needed. This can be seen especially in low-intensity images, where the color information is destroyed by noise, preprocessing, or camera settings. This research would also make the solution more generic. Future research could

address this problem through ablation study and look into more detailed explanation of why different architectures work Sheikholeslami (2019).

**Acknowledgements** This work was supported by Finnish Cultural Foundation's Central Ostrobothnia Regional Fund (Grant Number 25211242).

**Data availability** One of the datasets used in this manuscript is available as Zenodo repository: 10.5281/zenodo.7749912.

**Conflict of interest** There are no conflicts of interest.

## References

Ahad MT, Li Y, Song B, et al (2023) Comparison of cnn-based deep learning architectures for rice diseases classification. Artificial Intelligence in Agriculture 9:22–35

Ahmed HO, Nandi AK (2021) Connected components-based colour image representations of vibrations for a two-stage fault diagnosis of roller bearings using convolutional neural networks. Chinese Journal of Mechanical Engineering 34:1–21

Ahmed T, Sabab NHN (2022) Classification and understanding of cloud structures via satellite images with efficientunet. SN Computer Science 3:1–11

Albani D, Youssef A, Suriani V, et al (2017) A deep learning approach for object recognition with nao soccer robots. In: Behnke S, Sheh R, Sarıel S, et al (eds) RoboCup 2016: Robot World Cup XX. Springer International Publishing, Cham, pp 392–403

Alzubaidi L, Zhang J, Humaidi AJ, et al (2021) Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. Journal of big Data 8:1–74

Anandhakrishnan T, Jaisakthi S (2022) Deep convolutional neural networks for image based tomato leaf disease detection. Sustainable Chemistry and Pharmacy 30:100,793

Apriyanti DH, Spreeuwers LJ, Lucas PJ, et al (2021) Automated color detection in orchids using color labels and deep learning. PloS one 16(10):e0259,036

Arsenovic M, Karanovic M, Sladojevic S, et al (2019) Solving current limitations of deep learning based approaches for plant disease detection. Symmetry 11(7):939

Atha DJ, Jahanshahi MR (2018a) Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection. Structural Health Monitoring 17(5):1110–1128.

https://doi.org/10.1177/1475921717737051

Atha DJ, Jahanshahi MR (2018b) Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection. Structural Health Monitoring 17(5):1110–1128

Aznan A, Ruslan R, Rukunudin I, et al (2017) Rice seed varieties identification based on extracted colour features using image processing and artificial neural network (ann). Int J Adv Sci Eng Inf Technol 7(6):2220–2225

Bimler DL, Kirkland J, Jameson KA (2004) Quantifying variations in personal color spaces: Are there sex differences in color vision? Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur 29(2):128–134

Boulent J, Foucher S, Théau J, et al (2019) Convolutional neural networks for the automatic identification of plant diseases. Frontiers in plant science 10:941

Bressem KK, Adams LC, Erxleben C, et al (2020) Comparing different deep learning architectures for classification of chest radiographs. Scientific reports 10(1):13,590

Breve FA (2022) Covid-19 detection on chest x-ray images: A comparison of cnn architectures and ensembles. Expert systems with applications 204:117,549

Brodzicki A, Piekarski M, Kucharski D, et al (2020) Transfer learning methods as a new approach in computer vision tasks with small datasets. Foundations of Computing and Decision Sciences 45(3):179–193

Büyükarıkan B, Ülker E (2022) Using convolutional neural network models illumination estimation according to light colors. Optik 271:170,058. https://doi.org/https://doi.org/10.1016/j.ijleo.2022.170058

Cha YJ, Choi W, Büyüköztürk O (2017) Deep learning-based crack damage detection using convolutional neural networks. Computer-Aided Civil and Infrastructure Engineering 32(5):361–378

Chauhan MS (2018) Optimizing gaussian blur filter using cuda parallel framework. Information Technology Department, College of Applied Sciences Ibri, Sulatanate of Oman

Chen FC, Jahanshahi MR (2017) Nb-cnn: Deep learning-based crack detection using convolutional neural network and naïve bayes data fusion. IEEE Transactions on Industrial Electronics 65(5):4392–4400

Cheng G, Guo W (2017) Rock images classification by using deep convolution neural network. In: Journal of Physics: Conference Series, IOP Publishing, p 012089

Engilberge M, Collins E, Süsstrunk S (2017) Color representation in deep neural networks. In: 2017 IEEE International Conference on Image Processing (ICIP), IEEE, pp 2786–2790

Flachot A, Gegenfurtner KR (2021) Color for object recognition: Hue and chroma sensitivity in the deep features of convolutional neural networks. Vision Research 182:89–100

Fuentes A, Yoon S, Kim SC, et al (2017) A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. Sensors 17(9). https://doi.org/10.3390/s17092022

Furferi R, Governi L (2008) The recycling of wool clothes: an artificial neural network colour classification tool. The International Journal of Advanced Manufacturing Technology 37:722–731

Hakola L, Vehmas K, Smolander M (2021) Functional inks and indicators for smart tag based intelligent packaging applications. Journal of Applied Packaging Research 13(2):3

Han X, Zhong Y, Cao L, et al (2017) Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification. Remote Sensing 9(8). https://doi.org/10.3390/rs9080848

He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

Isohanni J (2022) Use of functional ink in a smart tag for fast-moving consumer goods industry. Journal of Packaging Technology and Research 6(3):187–198

Isohanni J (2023) Qr-code dataset, with colour embed inside. https://doi.org/10.5281/zenodo.7749912

Jhawar J (2016) Orange sorting by applying pattern recognition on colour image. Procedia Computer Science 78:691–697. https://doi.org/https://doi.org/10.1016/j.procs.2016.02.118, 1st International Conference on Information Security & Privacy 2015

Kagaya H, Aizawa K, Ogawa M (2014) Food detection and recognition using convolutional neural network. In: Proceedings of the 22nd ACM International Conference on Multimedia. Association for Computing Machinery, New York, NY, USA, MM '14, p 1085–1088, https://doi.org/10.1145/2647868.2654970

Kentsch S, Lopez Caceres ML, Serrano D, et al (2020) Computer vision and deep learning techniques for the analysis of drone-acquired forest images, a transfer learning study. Remote Sensing 12(8). https://doi.org/10.3390/rs12081287

Kim GI, Jang B (2023) Petroleum price prediction with cnn-lstm and cnn-gru using skip-connection. Mathematics 11(3):547

Kishore A, Singh S (2015) Natural language image descriptor. In: 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS), IEEE, pp 110–115

Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90. https://doi.org/10.1145/3065386

Kumar NS, Maheswari SU, Pramila P, et al (2022) Colour based object classification using knn algorithm for industrial applications. In: 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), IEEE, pp 1110–1115

Kumar SS, Abraham DM, Jahanshahi MR, et al (2018) Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. Automation in Construction 91:273–283

Kurtulmus F, Lee WS, Vardar A (2014) Immature peach detection in colour images acquired in natural illumination conditions using statistical classifiers and neural network. Precision agriculture 15:57–79

Lai P, Westland S (2020) Machine learning for colour palette extraction from fashion runway images. International Journal of Fashion Design, Technology and Education 13(3):334–340. https://doi.org/10.1080/17543266.2020.1799080

Lamb N, Chuah MC (2018) A strawberry detection system using convolutional neural networks. In: 2018 IEEE International Conference on Big Data (Big Data), pp 2515–2520, https://doi.org/10.1109/BigData.2018.8622466

Li J, Sun T, Lin Q, et al (2022) Reducing negative transfer learning via clustering for dynamic multiobjective optimization. IEEE Transactions on Evolutionary Computation 26(5):1102–1116

Luo J, Zhang D (2003) Automatic colour printing inspection by image processing. Journal of Materials Processing Technology - J MATER PROCESS TECHNOL 139:373–378. https://doi.org/10.1016/S0924-0136(03)00534-X

Maeda-Gutiérrez V, Galván-Tejada CE, Zanella-Calzada LA, et al (2020) Comparison of convolutional neural network architectures for classification of tomato plant diseases. Applied Sciences 10(4):1245

Mohanty SP, Hughes DP, Salathé M (2016) Using deep learning for image-based plant disease detection. Frontiers in plant science 7:1419

Muhammad K, Ahmad J, Baik SW (2018) Early fire detection using convolutional neural networks during surveillance for effective disaster management. Neurocomputing 288:30–42. https://doi.org/https://doi.org/10.1016/j.neucom.2017.04.083, learning System in Real-time Machine Vision

Nalinipriya G, Baluswarny B, Patan R, et al (2018) To detect and recognize object from videos for computer vision by parallel approach using deep learning. In: 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), pp 336–341, https://doi.org/10.1109/ICACCE.2018.8441718

Naseer I, Akram S, Masood T, et al (2022) Performance analysis of state-of-the-art cnn architectures for luna16. Sensors 22(12):4426

Neris R, Guerra R, López S, et al (2021) Performance evaluation of state-of-the-art cnn architectures for the on-board processing of remotely sensed images. In: 2021 XXXVI Conference on Design of Circuits and Integrated Systems (DCIS), IEEE, pp 1–6

Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Transactions on knowledge and data engineering 22(10):1345–1359

Paul R, Hassan MSu, Moros EG, et al (2020) Deep feature stability analysis using ct images of a physical phantom across scanner manufacturers, cartridges, pixel sizes, and slice thickness. Tomography 6(2):250–260

Przybyło J, Jabłoński M (2019) Using deep convolutional neural network for oak acorn viability recognition based on color images of their sections. Computers and electronics in agriculture 156:490–499

Przybyło J, Jabłoński M (2019) Using deep convolutional neural network for oak acorn viability recognition based on color images of their sections. Computers and Electronics in Agriculture 156:490–499. https://doi.org/https://doi.org/10.1016/j.compag.2018.12.001

Ravishankar H, Sudhakar P, Venkataramani R, et al (2016) Understanding the mechanisms of deep transfer learning for medical images. In: Deep Learning and Data Labeling for Medical Applications: First International Workshop, LABELS 2016, and Second International Workshop, DLMIA 2016, Held in Conjunction with MICCAI 2016, Athens, Greece, October 21, 2016, Proceedings 1, Springer, pp 188–196

Saikia AR, Bora K, Mahanta LB, et al (2019) Comparative assessment of cnn architectures for classification of breast fnac images. Tissue and Cell 57:8–14

Senthilkumar M (2010) 5 - use of artificial neural networks (anns) in colour measurement. In: Gulrajani M (ed) Colour Measurement. Woodhead Publishing Series in Textiles, Woodhead Publishing, p 125–146, https://doi.org/https://doi.org/10.1533/9780857090195.1.125

Sheikholeslami S (2019) Ablation programming for machine learning

Shore J, Johnson R (1981) Properties of cross-entropy minimization. IEEE Transactions on Information Theory 27(4):472–482

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556

Soukup D, Huber-Mörk R (2014) Convolutional neural networks for steel surface defect detection from photometric stereo images. In: International Symposium on Visual Computing, Springer, pp 668–677

Szegedy C, Liu W, Jia Y, et al (2015) Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1–9, https://doi.org/10.1109/CVPR.2015.7298594

Tan M, Le Q (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning, PMLR, pp 6105–6114

Taylor GA (2003) Improving image contrast. American Journal of Roentgenology 180(2):329–331

Tiwari S (2018) An analysis in tissue classification for colorectal cancer histology using convolution neural network and colour models. International Journal of Information System Modeling and Design (IJISMD) 9(4):1–19

Varde AS, Karthikeyan D, Wang W (2023) Facilitating covid recognition from x-rays with computer vision models and transfer learning. Multimedia Tools and Applications pp 1–32

Vidal-Calleja T, Miró JV, Martín F, et al (2014) Automatic detection and verification of pipeline construction features with multi-modal data. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 3116–3122, https://doi.org/10.1109/IROS.2014.6942993

Wong TT, Yeh PY (2019) Reliable accuracy estimates from k-fold cross validation. IEEE Transactions on Knowledge and Data Engineering 32(8):1586–1594

Wu J, Zhang B, Zhou J, et al (2019) Automatic recognition of ripening tomatoes by combining multi-feature fusion with a bi-layer classification strategy for harvesting robots. Sensors 19(3). https://doi.org/10.3390/s19030612

Wu Y, Qin X, Pan Y, et al (2018) Convolution neural network based transfer learning for classification of flowers. In: 2018 IEEE 3rd international conference on signal and image processing (ICSIP), IEEE, pp 562–566

Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13, Springer, pp 818–833

Zhang Q, Zhuo L, Li J, et al (2018) Vehicle color recognition using multiple-layer feature representations of lightweight convolutional neural network. Signal Processing 147:146–153

Zhang S, Huang W, Zhang C (2019a) Three-channel convolutional neural networks for vegetable leaf disease recognition. Cognitive Systems Research 53:31–41

Zhang W, Li X, Ding Q (2019b) Deep residual learning-based fault diagnosis method for rotating machinery. ISA transactions 95:295–305

Zhao Y, Shen Q, Wang Q, et al (2020) Recognition of water colour anomaly by using hue angle and sentinel 2 image. Remote Sensing 12(4). https://doi.org/10.3390/rs12040716

Zhu C, Zou B, Zhao R, et al (2017) Retinal vessel segmentation in colour fundus images using extreme learning machine. Computerized Medical Imaging and Graphics 55:68–77. https://doi.org/https://doi.org/10.1016/j.compmedimag.2016.05.004, special Issue on Ophthalmic Medical Image Analysis

Zhu D, Qiu D (2021) Residual dense network for medical magnetic resonance images super-resolution. Computer Methods and Programs in Biomedicine 209:106,330

Zhu Y, Newsam S (2017) Densenet for dense flow. In: 2017 IEEE international conference on image processing (ICIP), IEEE, pp 790–794

INTERNATIONAL JOURNAL OF COMPUTERS AND APPLICATIONS
2025, VOL. 47, NO. 4, 341–355
https://doi.org/10.1080/1206212X.2025.2465727



# Customised ResNet architecture for subtle color classification

Jari Isohanni

Digital Economy, University of Vaasa, Vaasa, Finland

**ABSTRACT**

This study addresses the challenge of recognizing subtle color differences, a problem critical to applications in fields such as healthcare, food production, and civil engineering. Specially research focusses on printed colors. The research evaluates multiple ResNet architectures, including ResNet-18, ResNet-34, and ResNet-50, to identify the most effective model for this task. Modifications to the ResNet-34 architecture are proposed, such as replacing average pooling with global max pooling and introducing max pooling layers within residual blocks, to enhance feature extraction and classification accuracy. The models were validated using a *K*-fold cross-validation, which confirms the effectiveness of the proposed approaches. The findings demonstrate the potential of these modifications to achieve high classification accuracy, showcasing their adaptability to real world scenarios. However, limitations such as the use of a specific dataset and the type of printer highlight the need for further research to generalize the approach across diverse datasets and conditions.

**ARTICLE HISTORY**
Received 24 May 2024
Accepted 5 February 2025

**KEYWORDS**
Machine vision; color difference; printed colors; convolutional neural networks (CNN); ResNet; max pooling

## 1. Introduction

ResNet (Residual neural network) by He et al. is a deep learning architecture for image classification and computer vision tasks. Unlike traditional convolutional neural networks (CNNs), ResNet uses skip connections. These connections allow the neural network to learn efficiently through residual mappings. Before ResNet was published, deep networks had mechanisms which attempted to directly approximate the underlying mapping. ResNet architecture presented learning by residuals, the difference between the input and output of a given layer [1–3].

ResNet architecture emerged as an advance in deep learning, addressing critical challenges associated with training very deep neural networks. By introducing skip connections, ResNet mitigates the vanishing gradient problem, enabling effective training of networks with hundreds or even thousands of layers. This not only improves the flow of gradients during backpropagation but also enhances the reuse of features across layers, leading to better representation learning [4, 5]. ResNet's ability to capture fine-grained details and subtle variations in data makes it a good choice for numerous computer vision applications, including image classification, object detection, and feature extraction. During the year 2024 over 41 000 articles mentioning ResNet were published, according to Google Scholar search.

ResNet architecture adds shortcuts (skip connections) every two layers. This enables the layers of residual networks to learn from residual mappings, this learning helps in representing identity mappings, and finally prevents networks from degradation as depth increases. The novelty of adding shortcuts every two layers is crucial, as previous research has shown that using skip connections on every or every third layer does not achieve the same performance [6].

The innovation of ResNet has sparked research around its approach, and there have been interest in modifying, extending, and adapting the ResNet architecture. Targ et al. presented an approach that adds convolutional layers and data paths to each layer [7]. Wide Residual Networks is a version of ResNet invented by Zagoruyko and Komodakis in their research where they showed the importance of residual blocks. Wide residual networks can outperform deep networks in some use cases, and their computational cost is lower [8]. Li and He explained how identity shortcut connections solve gradient fading problems and proposed adjustable shortcut connections. The authors stated that identity mappings are not reasonable to be adopted for all layer parameters [9]. HS-ResNet, an approach proposed by Yuam et al., implements a plug-and-play block that can be added to existing networks. HS-ResNet implements hierarchical split and concatenate connections within one single residual block [10]. Targ et al. came up in their research with ResNet in ResNet (RiR), RiR has a deep dual-stream architecture that generalizes ResNet and optimizes ResNet performance [10]. Stable ResNet by Hayou et al. addressed the problem of an exploding gradient which can occur if ResNet becomes very deep. Their approach looks to stabilize the gradient [11]. Another version, CO-ResNet was proposed by Bharati et al. Their model was optimized to detect COVID-19 in X-ray images. The authors mostly focused on hyperparameter tuning [12].

The popularity and good performance of the ResNet architecture have resulted in not only modified approaches, but also different depth variations of the ResNet. These variations are named based on how deep they are; for example, ResNet-18 has 18 neural network layers. Different versions of ResNet are, for example, ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-110, ResNet-152, ResNet-164, and ResNet-1202. These variations and their performance are evaluated in the first experiments of this research.

ResNet and its different versions have proven that the ResNet capabilities are useful in image classification tasks in many different use cases [13–19]. In one recent research, ResNet was used to classify different printed colors, in this use case ResNet achieved a high accuracy of 95% even when the CMYK color intensity was only 5% in some of the color channels.

The recognition of small color differences is not a commonly studied topic, but it is closely related to the development of functional inks. These inks are used in the packaging industry and can

**CONTACT** Jari Isohanni x2603813@student.uwasa.fi, jari.isohanni@gmail.com Digital Economy, University of Vaasa, Wolffintie 34, Vaasa 65200, Finland

be used as sensors for environmental values such as temperature or humidity. This is possible as functional inks change color according to their exposure to the variables [20].

In this research, ResNet variations are explored to see which of them can best classify subtle color differences. The best ResNet variation is selected for further fine-tuning to find out if it can be modified for optimized accuracy. Fine-tuning includes changing the architecture based on findings of the past research.

This article makes contributions to the optimization of ResNet architectures for addressing the challenge of detecting subtle color differences in printed sources, a task with significant implications in various applications. The study examines multiple ResNet models, including ResNet-18, ResNet-34, and ResNet-50, to identify a baseline architecture suitable for fine-grained color classification. Based on this foundation, modifications are proposed to the ResNet-34 architecture, focusing on adjustments to pooling strategies and feature extraction within residual blocks. The approach presented highlights the adaptability of ResNet architectures to subtle color classification tasks and underscores the potential of targeted architectural modifications to improve their performance. The findings can be used in various applications in sectors such as civil engineering, food production, and healthcare care, where precise color differentiation is needed.

This research is structured as follows. Section 2. contains an overview of the past research. Section 3. describes the methods and materials used, including the dataset, ResNet architecture, and other methods related to this research Section 4. describes the experiments used and their results. Section 5. discusses the results and provides information for future research. Section 6. goes trough the conclusions of the research.

## 2. Related past research

ResNet has been used to solve various problems in the past; some of the previous research uses different ResNet depth variations and applies them to specific use cases. Some past research adapts and modifies ResNet in various ways. This section summarizes some of the most relevant work done around ResNet in the past.

Sarwinda et al. used ResNet in the detection of colorectal cancer, experimenting with ResNet-50 and ResNet-18. All of their images were pre-processed with the contrast enhancement CLAHE method before making the dataset. In this research ResNet-18 reached an accuracy of 85% and ResNet-50 88% [14].

Subrataio et al. tuned the ResNet-101 hyperparameters to create a CO-ResNet model. This model was used to classify COVID-19 and pneumonia X-ray images. Their detection rate was 98.74% in cases of COVID-19, 92.08% and 91.32% in cases of normal and pneumonia. They also tested ResNet-152 which did not reach 90% accuracy [12].

Almoosawi and Khudeyer based their approach on ResNet-34 when they researched the identification of diabetic retinopathy. In their image dataset, the differences are quite small as in the research presented in this research. The F1-score of their solution was 93.2%. The F1-score is a combination of precision and recall, providing a balance between the two [21].

Yu et al. research recognition of early-stage breast cancer using the ResNet-50 architecture. In this research, SCDA (Scaling and Contrast Limited Adaptive Histogram Equalisation Data Augmentation) was used as a data augmentation tool. The model and approach used in this research reached 95.74% [16].

Gao et al. used ResNet-34 architecture and transfer learning as they had only a small amount of samples. ResNet-34 classified some classes with accuracy 100%, and even in the most challenging class, the leaf knot, the accuracy was 97%. The Leaf knot classification is closely related to the small difference classification problem presented in this research [17].

Hu et al. combined two ResNet-50 and one ResNet-34 architectures into a multidimensional feature compensation residual neural network. Each dimension was responsible for certain classifications related to crop diseases. As the last layer, authors had a compensation layer which used a compensation algorithm to determine final recognition result. Their approach achieved 85 % accuracy, which was better than other approaches with their dataset [22].

Al-Haija and Adebanjo worked on their research with the breast cancer dataset. With the ResNet-50 architecture and transfer learning, they achieved very high 99% accuracy [23].

ResNet-50 has shown its accuracy in disease identification in the past. Potato plant leaf disease identification was done by Shaheed et al. In this research ResNet reached 99.12% accuracy in only under 30 epochs. Zhang et al. performed another study that used ResNet-50 in disease identification. The researchers also used the coordinate attention module (CA) and the weight-adaptive multiscale feature fusion (WAMFF) and were able to achieve accuracy of 98.32 %. In addition, Li and Rai researched the identification of diseases in apples; however, they proved that shallow ResNet-18 outperformed ResNet-34 [24–26].

The summary of the previous research is shown in the following table.

| Research title | Authors | Findings |
|---|---|---|
| Sarwinda et al. | Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer | Comparison of the ResNet-18 and ResNet-50 |
| Subrataio et al. | CO-ResNet: Optimized ResNet model for COVID-19 diagnosis from X-ray images | Hyperparameter tuning to increase performance of the ResNet-101 |
| Almoosawi and Khudeyer | ResNet-34/DR: a residual convolutional neural network for the diagnosis of diabetic retinopathy | Using of preprocessing to increase performance of the ResNet |
| Yu et al. | ResNet-SCDA-50 for breast abnormality classification | Using contrast enhancement to improve ResNet performance in abnormality classification |
| Gao et al. | A Transfer Residual Neural Network Based on ResNet-34 for Detection of Wood Knot Defects | ResNet was used to identify differences in wood, some of the changes were quite subtle |
| Hu et al. | MDFC–ResNet: An Agricultural IoT System to Accurately Recognize Crop Diseases | Using of ResNet to identify small differences in images |
| Al-Haija and Adebanjo | Breast Cancer Diagnosis in Histopathological Images Using ResNet-50 Convolutional Neural Network | Identifying features from images |
| Zhang et al. | Classification and Identification of Apple Leaf Diseases and Insect Pests Based on Improved ResNet-50 Model | Improving ResNet-50 with coordinate attention (CA) module and weight-adaptive multi-scale feature fusion |
| Shaheed et al | EfficientRMT-Net – An Efficient ResNet-50 and Vision Transformers Approach for Classifying Potato Plant Leaf Diseases | Integration of Vision Transformer (ViT) and ResNet-50 architectures |
| Li and Rai | Apple leaf disease identification and classification using resnet models | Comparison of ResNet-18 and ResNet-34 in lead disease identification |

Previously ResNet has been used in various use cases, and the mentioned research are closely related to the use case presented in this research. The similarity comes from the problem of recognizing subtle differences in images. Previously, ResNet and related architectures have been shown to perform well when small differences in images are observed or located. Most research in the past has used the standard version of the ResNet, the most popular being 50-layer and 34-layer versions. These previous researches have mainly focused on improving the performance of ResNet via pre-processing or other methods that do not directly modify the architecture of ResNet. But there are also examples of modifications that make the architecture work better in specific use cases. In addition, there is some research showing that increasing the depth of the ResNet architecture does not correlate with accuracy. As ResNet achieves high accuracy in multiple of the presented cases, choosing the best optimizer or hyperparameter tuning has not been a very popular topic in relation to ResNet. Another lesson that can be learnt from the previous research is that using enough correct data augmentation helps when the generic and more accurate model is the objective.

The main gap in previous research is that ResNet (or any other CNN architecture) has not been used to classify subtle color differences. This research explores how ResNet can be used in this use-case and if better architecture can be developed with small changes. This article proposes two modified ResNet-34 architectures tailored to the subtle color difference classification in printed sources. The first modification replaces the average pooling layer with global maximum pooling before the fully connected layer, enhancing feature extraction by focusing on the most significant features across the input space. The second modification adds a maximum pooling layer after each convolutional operation within residual blocks, improving the network's ability to capture subtle differences by emphasizing local maxima. These adjustments are supported by additional techniques such as gradient centralization and the use of $K$-fold cross-validation for robust performance evaluation. These proposed approaches overcome identified gaps by improving model accuracy, reducing overfitting, and tailoring ResNet-34 for subtle color classification.

## 3. Methods and materials

### 3.1. The dataset

The dataset [27] used in this research contains images that have been acquired using normal smartphones in various real-life environments. The images contain a special QR-Code as a carrier of color information (Figure 1). Before any operation, the source images are run through an auto-leveling process. This process is a technique commonly used in image processing to automatically adjust the contrast of the image. Auto-leveling aims to spread out the brightness levels across the entire dynamic range available. The purpose of image auto-levelling is to enhance presentation of colors by automatically adjusting the brightness, contrast, and color balance of an image to achieve a more natural and evenly distributed tonal range. The auto-levelling process starts with the identification of the minimum and maximum intensity values within the image. After determining the minimum and maximum intensity values, each pixel's intensity in the image is mapped to a new value based on a linear transformation. Pixels with intensities below the minimum value are assigned 0 (black), while pixels with intensities above the maximum value are assigned 255 (white). The intensities between are linearly scaled to cover the entire dynamic range. The mapping process is done for all RGB-channels [28]. An example of auto-levelling can be seen in Figure 1. where image 1 is before and image 2 after auto-levelling. After the auto-levelling the image goes through a Gaussian blurring. Blurring reduces noise in an image by averaging the intensity values of neighboring pixels, thereby smoothing out abrupt variations caused by random noise or, as in the use case presented, printer patterns. Noise typically appears as sharp, high-frequency intensity fluctuations that stand out from the underlying signal or pattern in the image. Blurring applies a low-pass filter, which suppresses these high-frequency components while retaining the overall structure of the image.

The actual color information used is placed at two specific locations on the QR-Code. These locations and their cornerpoints are calculated as part of the QR-Code decoding process. The locations and their cornerpoints are used to correct area skew and orientation, resulting in square color areas. These color areas are then extracted into separate images for pre-processing and forming of the final dataset. The extraction process extracts 80% from the color area to ensure that only the needed area is left. In this way, the possible distortion and skew that is left will not impact the process.

Each of the source images originally contains two different areas (Figure 1). The first area is a rectangle with paper-white (green rectangle), e.g. no color printed, and the second one has some color printed on it (yellow rectangle). Printed color is the one CNN model is expected to classify. In the process, first an average RGB value for the paper-white area is calculated, this is done with the following formula:

$$\bar{C}(c) = \frac{1}{M}\sum_{x=1}^{W}\sum_{y=1}^{H} I(r,g,b) \tag{1}$$

where $I$ is the source image and $R$, $G$, and $B$ represent individual integer values in each RGB channel at pixel location $(x, y)$. $W$ and $H$ are the width and height of the image and $M$ is the total number of pixels in the image.

To form the final image, used as the dataset image, the average color of the white area is subtracted from the color area. This creates a difference image which contains information about how much color area differs from no color area. The difference image is calculated with the following formula:

$$I_{\text{difference}}(r,g,b) = |I_{\text{original}}(r,g,b) - C(r,g,b)| \tag{2}$$

where $I_{\text{difference}}$ is the final image and $I_{\text{original}}$ is the color area image (Figure 1(b)). Constant $C$ is the average RGB color value in the white area Figure 1(a).

In Figure 2 process of forming the difference image is shown with a few examples. In Figure 2 row (a) contains sample images of average paper-white color, row (b) contains the actual color area, and row (c) is the final dataset image. The columns in the figure represent different colors printed on paper 10% yellow, 10% cyan, 10% magenta, 5% yellow, 5% cyan, 5% magenta. In the final difference images (Figure 2 row (c)), it can be seen that the differences between the samples are very small. This makes it difficult for the convolutional neural network (CNN) to classify images correctly.

The whole process of making the dataset is illustrated in Figure 3.

The dataset is augmented with the following options, data augmentation is one option to enlarge the dataset and can provide better accuracy [29]. Images are rotated in angles, 90, 180, and 270, then images are flipped vertically and horizontally, and the same rotations are done for flipped version of images. Also, the 0.2 zoom option is used to make more images using data augmentation.

In total, the dataset contains 7855 RGB images. Each image is stored with 24-bit depth, which means 8 bits per each channel. The images are split into train (80%) and validation (20%) sets so that the train set contains 6284 images belonging to 11 classes and the validation set contains 1571 images belonging to 11 classes (Table 1).

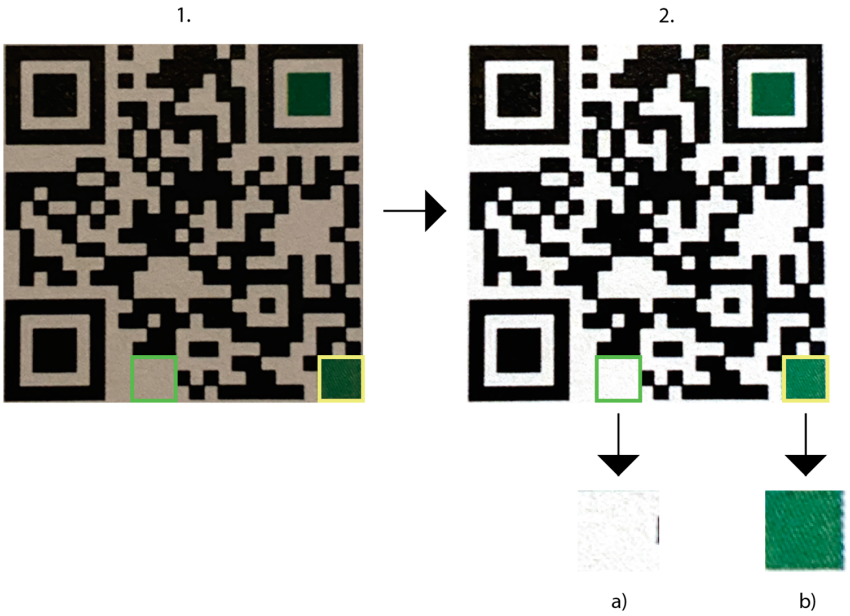



**Figure 1.** The process of color correction and area extraction.

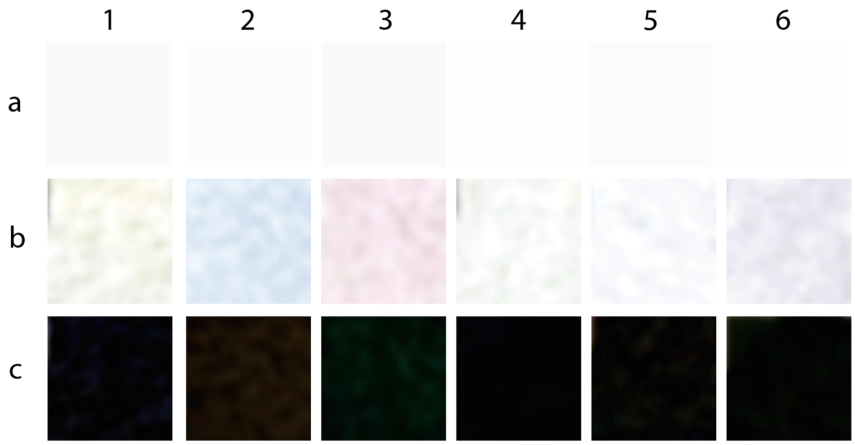



**Figure 2.** Samples of images used (best viewed online).

**Table 1.** Images per each class.

| Color | Images | Colour | Images |
|---|---|---|---|
| 10% K | 769 | 5% K | 542 |
| 10% M | 770 | 5% M | 638 |
| 10% Y | 770 | 5% Y | 650 |
| 10% CY | 818 | 5% CY | 506 |
| 10% C | 830 | 5% C | 626 |
| 0% K | 926 | | |

### 3.2. Convolutional neural networks

A Convolutional Neural Network (CNN) is a specialized type of artificial neural network specifically designed for processing and analyzing grid-structured data, such as images. Its architecture is optimized to reduce the number of parameters and computational overhead of the neural network, making it particularly well suited for the efficient handling of high-dimensional datasets [30].

The convolutional layer is the fundamental building block of a CNN (Figure 4). It applies convolutional filters (kernels) to localized regions of the input data, generating feature maps that capture essential patterns such as edges, color information, textures, and shapes. Each filter is specialized to detect a specific type of feature, and during training, the network optimizes these filters to best suit the task at hand [30].

The pooling layers in CNNs are used to down-sample feature maps, reducing their spatial dimensions and the number of parameters in the network. This reduction decreases the computational complexity and helps prevent overfitting, thereby enhancing the network's ability to generalize to new data. Common pooling methods include max pooling, which selects the maximum value within a specified window, and average pooling, which computes the average value, both contributing to feature extraction while simplifying the model [31, 32].

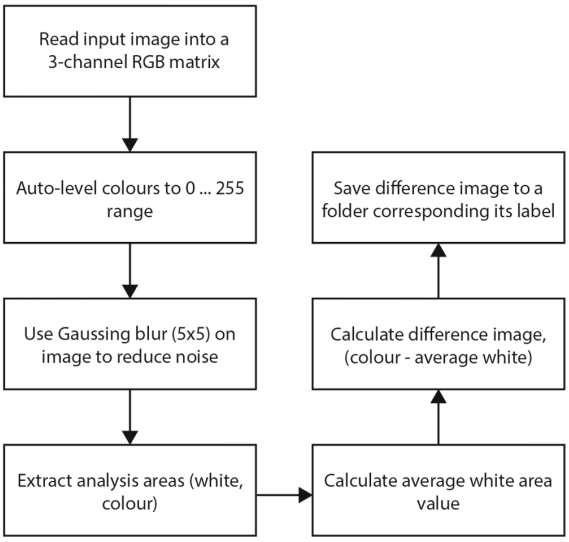



**Figure 3.** The process of color correction and area extraction.

A key feature of CNNs is the use of non-linear activation functions, such as the Rectified Linear Unit (ReLU). ReLU introduces non-linearity into the model, enabling it to capture complex patterns and relationships within grid-structured data. Furthermore, ReLU helps accelerate training convergence by alleviating the vanishing gradient problem, which is commonly encountered with activation functions such as sigmoid or tanh [30, 33].

In the final stages of a CNN, fully connected (dense) layers are used to consolidate the high-level features extracted by the convolutional layers. These layers are responsible for performing tasks such as classification or regression, using the learnt representations to generate the final output [30].

### 3.3. ResNet architecture

ResNet, short for Residual Network, was introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun from Microsoft Research in their paper titled 'Deep Residual Learning for Image Recognition', which was presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) in 2016 [1].

The ResNet architecture was the first convolutional neural network architecture to use residual learning. Residual learning addresses the vanishing gradient problem, which usually occurs when neural networks become deep. The vanishing gradient issues are closely tied to backpropagation (backward propagation of errors). In neural networks, backpropagation is a crucial element of the training process. In the backpropagation, the network learns to adjust its weights and biases to minimize the difference between its predicted output and the actual target output [34].

In backpropagation, the input data is moved through the network to produce predictions (forward pass). In this process, the difference between the predictions and the actual output (loss) is computed. During the backward pass, gradients of the loss concerning network parameters are calculated using the chain rule. These gradients are then used to guide the adjustment of weights and biases, the purpose is to minimize the loss through iterative updates. As the process is repeated for multiple epochs, it allows the neural network to learn and improve its predictive capabilities [35].

The vanishing gradient problem occurs in deep neural networks during backpropagation, as gradients are exponentially diminished in the process of backward passing through layers. In particular, the vanishing gradient problems occur in networks with many layers. Another context where vanishing gradient easily occurs are activation functions with saturating gradients, such as sigmoid or hyperbolic tangent functions. In the vanishing gradient problem gradients are approaching zero when moving closer to early layers, so these layers receive only minimal updates. As this happens, the learning process decreases, which eventually leads to slow convergence and poor performance in the training of deep networks [36, 37].

The vanishing gradient problem can easily occur in low-level features, where the difference on pixel level is small. DenseNet architecture or its modifications have been successfully used in the past to overcome problems with the low-level features [38–40].

The innovation presented in ResNet was skip connections (Figure 5), also known as shortcut connections, which skip one or more layers by adding the input of a layer to its output. This allows the network to learn residual functions instead of directly learning the underlying mapping functions. Instead of purely learning a mapping from input to output, each layer of a residual network is tasked with learning the residual function, the difference between the desired output and the input to that layer. With this approach ResNet enables the training of much deeper networks (hundreds of layers) while maintaining or improving performance [1].

Mathematically, this can be represented as follows:

$$Output = Input + Residual \tag{3}$$

When residuals are used, the network can focus on learning small incremental adjustments to the input rather than learning to generate the entire output from scratch. Residual connections enable the gradient to flow more easily during training, mitigating the problem of vanishing gradient. In practice (Figure 5), a residual block typically consists of two or more convolutional layers followed by a skip connection that adds the input to the output of the convolutional layers.

Recently ResNet has been used successfully, for example, in solving different problems. For example, Hossain et al. proposed a weighted ensemble deep transfer learning framework with ResNet152 to identify Alzheimer disease from MRI images. Hassan et al. used ResNet-50 to classify images in the Medical MNIST dataset [41]. Senapati et al. also used ResNet-50 to classify food images, Wu et al. for chicken gender identification and Lin & Wu for diabetic retinopathy detection. Madhukar et al. incorporated ResNet in cancer image segmentation [42–46]. ResNet has also been used in many other recent researches. Usually, it achieves high classification accuracy in cases where small differences play an important role.

The development of the ResNet architecture has also led to different variations, the most popular way being to change the amount of layers. For example, ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152 have been used in previous research. One of the most popular architecture of these is ResNet-50.

But also some variations have been researched, Wide ResNet increases the width of ResNet by increasing the number of channels in each convolutional layer. This can improve performance, but requires more computational resources [47].

ResNeXt introduced a new block structure that increased model capacity by aggregating multiple paths of information flow within each block [48].ResNetv2 introduced improvements to the original ResNet architecture, such as using preactivation residual units and employing a bottleneck structure in all layers [2].

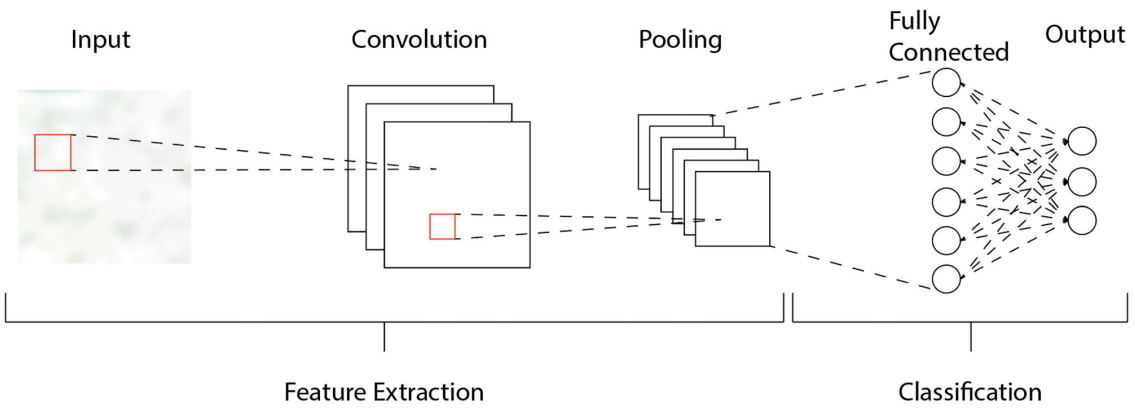



**Figure 4.** Simplified illustration of convolutional neural network.

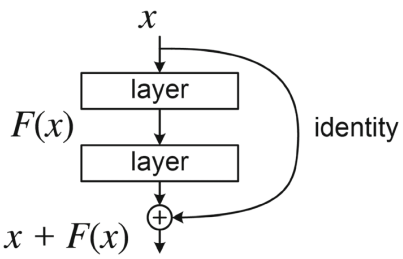



**Figure 5.** Skip connection [1].

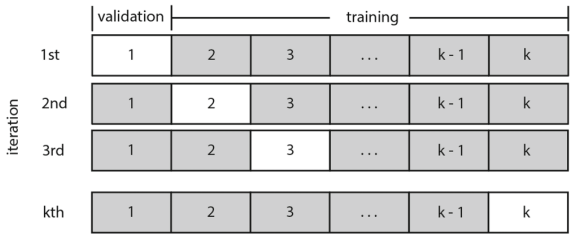



**Figure 6.** Illustration of *K*-fold cross-validation.

### 3.4. Cross-validation

In this research, the dataset used is quite small, with only 7855 images. For this reason, *K*-Fold cross-validation is used. In the *K*-Fold Cross-Validation, the training dataset is divided into *k* subsets of approximately equal size. These subsets are often called 'folds', so cross-validation is *K*-fold. With the used dataset size, the size of each subset is around 1560 images, when five folds are used. The advantage of using *K*-fold Cross-Validation is that model will be evaluated multiple times during the process. The final model when the *K*-fold cross-value process is used is generally less biased than if only one training / validation dataset is used [49, 50].

The process of the *K*-fold Cross-Validation (Figure 6) is following, before starting the process images are placed into corresponding folders, where folder represent image labels.

(1) Dataset is shuffled randomly
(2) Number of folds (*k*) is chosen and dataset is split into *k* groups
(3) For the each group:Group is taken either as hold out or test data set Rest of the groups are used as training data set Model is fitted with training data set and evaluated on the test set Evaluation score is kept and model discarded
(4) After each group is processed, model is summarized by using the sample of model evaluation scores [51]

During the process, it is important that the images remain in the same group, just because the whole group changes. During the process, this means that each image is used to train the model $k-1$ times

The advantage of *K*-Fold cross-validation is that by training the model *k* times using different combinations of training and validation sets, it is possible to obtain more reliable performance metrics than with a single train-test split. Also, when optimizing the hyperparameters, *K*-Fold cross-validation helps to choose the best hyperparameter values, without making model overfitted [52].

However, choosing the value *k* is important, typically *k* values like 2,5,10 are used [49]. The larger *k* uses more computational resources and could lead to overfitting, so the value of *k* should be kept as low as possible. In their research Yadav and Shukla have proposed that for small dataset *k* value 5–6 would provide the best results, and Wong et al. showed that cross-validation should be repeated twice for such dataset [49, 51].

### 3.5. Gradient centralization

Gradient Centralisation (GC), a method proposed by Yong et al., can be used instead of batch normalization (BN) and weight standardization (WS). As BN and WS make use of activations or weights, GC uses gradients directly and centralizes gradient vectors to have zero mean. Typically, using gradient centralization can lead to better generalization performance [53].

Batch normalization, being the most used of these, for internal normalization has the disadvantage of being an independent layer which processes data even after training. BN is usually also applied to a relatively large batch [54]. GC works directly on top of the gradient and centralizes the gradient vector so that it have zero means. The calculation of GC is done by getting the mean of each column of the gradient matrix/tensor, and then the center of each column is transferred to the zero means [55].

In the research conducted by Agarwal et al. GC was able to achieve higher accuracy of Denset models, which is a close relative of ResNet, than without GC. In this research, it was also shown that Adam's optimizer was the best option to train a neural network [56].

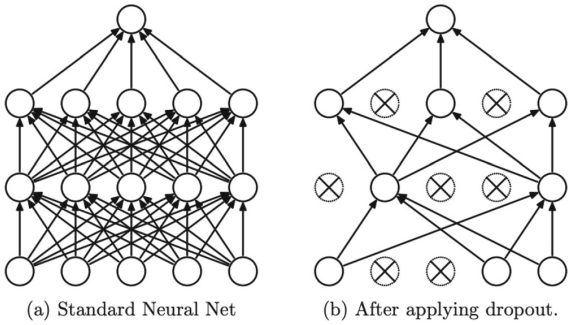



**Figure 7.** Illustration of dropout [60].

### 3.6. Dropout

Dropout can be used in convolutional neural networks to prevent the model from overfitting, also it has been shown that the use of dropout in the early phase of training can be used to avoid underfitting [57]. In the dropout, some hidden unit nodes are set stochastically to zero, which means that the nodes are removed and all forward and backward connections are also removed (Figure 7). This makes more gradient information flow through non-linear activation functions [58].

Dropout works show that during the training process, the nodes are dropped with a dropout probability of $p$ [59, 60]. The probability $p$ can vary, depending on the location of the dropout in the architecture. The probability is different if the dropout layer is placed close to the input or closer to the output. Especially dropout has been proven to work in connection with fully connected layers [61]. During the training process, probability can be seen as one hyper-parameter which can be adjusted to find the optimal model. The dropout rate is usually smaller ($p < 0.2$) in input layer and larger ($0.5 < p < 0.8$) when used internally or close to a fully connected layer [62]. Some research has also proposed a more controlled way to use dropout [63].

Dropout layers can be easily added to any architecture; this has some consequences. When dropout is added, it removes some units, which reduces the capacity of the network; this can be compensated by adjusting the number of units used by multiplying them by $1/(dropoutrate)$. As dropout introduces noise to the gradient, it has been shown that increasing the learning rate and momentum is needed; however, this depends on the optimizer used. If such a modification is made, max norm regularization might be needed [62].

When a dropout layer is added to an architecture it also makes it possible to fine-tune hyperparameters that relate. Depending on the used optimizer learning rate, weight decay and momentum parameters can be tuned, and this is usually even required to get the model to work in a optimized way. Hyperparameters are important especially if the standard stochastic descent gradient (SGD) optimizer is used instead of adaptive optimizers [62].

Regarding the development of different dropout methods, past research has considered many of those, for example [64–68]. In this research, standard version, where nodes area randomly dropped, is used.

### 3.7. Max and average pooling

The pooling layers are used in convolutional neural networks for downsampling. Currently, the most common pooling layers are max and average pooling. In addition, other pooling methods have been invented. The purpose of downsampling is to reduce the spatial resolution of feature maps to extract semantic information [69].

Pooling layers have two main objectives: (1) they reduce the number of parameters, which makes training of the networks less expensive, and (2) they help in avoiding possible overfitting of the network [70].

Pooling can either be done based on local regions, like $3 \times 3$, $5 \times 5$ or $7 \times 7$ areas. Another option is to use global pooling, where each feature maps across all spatial locations, resulting in a global representation of the features that summarizes the entire input volume [70].

When the values are calculated in the pooling layer (Figure 8), the maximum or average pooling is used. Maximum pooling uses the greatest value within a given region ($k \times k$) and pools it into the corresponding region in the downsampled feature map. Average pooling calculates the average value within a given region ($k \times k$) and uses this value in the downsampled feature map [70].

Using maximum or average pooling depends on the problem. In most cases, convolutional neural networks (CNNs) are looking to recognize significant values in images. This would make maximum pooling a preferable option, but average pooling is better at preserving localization [71].

## 4. Experiments and results

The experiments were run in a Python environment, with a laptop with an Apple M2 processor, which had 8 cores and a total RAM of 16 Gt. The environment used Python version 3.9.6 with Tensorflow and Keras 2.15.0. The architectures used were built manually on the basis of the ResNet architecture documentation and examples available.

The training process of the models was done with Adam optimizer, with default settings as the purpose of this research was not to focus optimizing individual models via hyper-parameters. For each of the training processes, 30 epochs were used.

### 4.1. Evaluation metrics

The purpose of the experiments was to find out which approach achieves the best accuracy. Accuracy is an effective performance metric when applied to datasets with balanced class distributions. A balanced dataset ensures that each class is equally represented, minimizing the risk that performance metrics are disproportionately influenced by the prevalence of a particular class. In these scenarios, accuracy offers a clear and straightforward measure of the proportion of correctly classified instances in the total number of instances, capturing the overall predictive capacity of the model. Accuracy serves as an appropriate metric when the application context assigns equal importance to all classes in the dataset. In classification problems where the costs or implications of misclassifications are uniform across all classes, accuracy provides a meaningful and holistic evaluation of the model's performance.

Accuracy can be calculated with the following equation:

$$Accuracy = \frac{Number\ Of\ Correct\ Predictions}{Total\ Number\ Of\ Predictions} \tag{4}$$

In the experiments, the accuracy was calculated for the entire dataset.

In addition, a confusion matrix was created for a detailed breakdown of correct and incorrect predictions for each class in a classification problem. A confusion matrix is an effective tool for evaluating the performance of classification models. Confusion matrix provides a detailed comparison of predicted versus actual outcomes, enabling the assessment of overall accuracy and the identification of specific error patterns. This analysis can be used to improve the

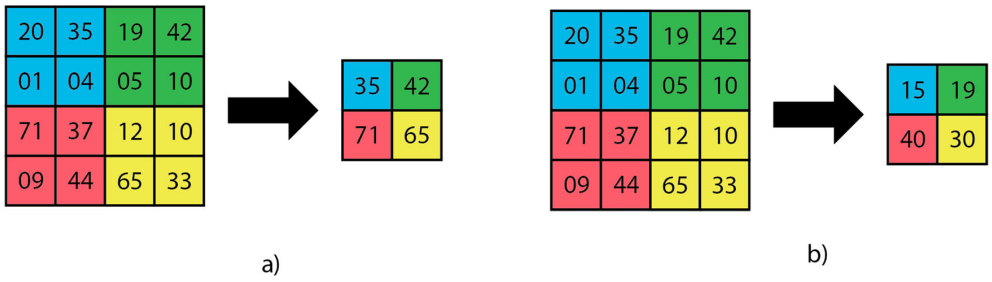



**Figure 8.** (a) Max pooling and (b) average pooling.

model architecture, optimize preprocessing methods, and improve data acquisition strategies [72, 73]

A confusion matrix is typically presented as a structured square table with rows and columns representing different classes in the classification task. For a binary classification problem (labelled as Positive and Negative), the matrix is a 2 × 2 table:

| | Predicted Positive | Predicted Negative |
|---|---|---|
| Actual Positive | True Positive(TP) | False Negative (FN) |
| Actual Negative | False Positive (FP) | True Negative (TN) |

True Positive (TP) refers to instances correctly identified as positive, while True Negative (TN) corresponds to instances accurately classified as negative. False Positive (FP) represents instances incorrectly predicted as positive, and False Negative (FN) denotes instances incorrectly predicted as negative.

In multiclass classification problems, the confusion matrix extends the binary version into a square matrix, with rows and columns representing the various classes. This structure provides a detailed view of the performance of the model, showcasing accurate classifications and misclassifications in all categories.

For a classification problem with $n$ classes, the confusion matrix will be an $n \times n$ matrix. Each element in the matrix at position $(i, j)$ represents the number of instances where the true class is $i$, and the predicted class is $j$. The diagonal elements represent the number of correct predictions for each class, and the off-diagonal elements represent misclassifications.

In Figure 9, 13 images of *label 1* were classified as *label 1*, and no images were classified as other labels. Then, in *label 2*, some images have been labeled as *label 3*. Furthermore, all images of *label 3* are correctly labeled. Therefore, the model may require some improvement.

A confusion matrix offers a more nuanced understanding of a classification model's performance compared to scalar metrics such as accuracy. The confusion matrix identifies the specific classes that the model tends to confuse and highlights strengths and weaknesses in its predictions. This detailed analysis provides critical information for the refinement of model design, the optimization of training processes, and the addressing of specific areas for improvement.

### 4.2. Standard ResNet architectures

As the purpose of the experiment was to find the best model for the use case, the fine-tuning of the hyperparameters and choosing the best optimizer were left for another research. All models were compiled with categorical cross-entropy loss and Adam optimizer (Table 2).

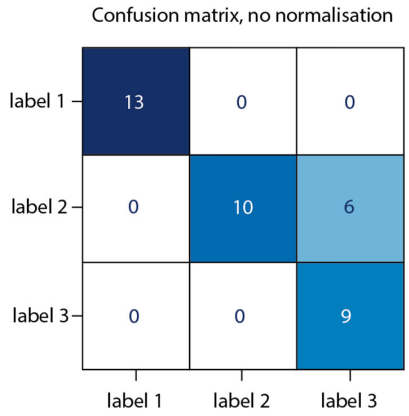



**Figure 9.** Example of confusion matrix for multi-classification.

**Table 2.** Results of first cycle.

| Architecture | Centralised gradient used | avg. train time/epoch (s.) | Accuracy |
|---|---|---|---|
| ResNet-18 | Y | 323 s. | 0.934 |
| ResNet-18 | N | 316 s. | 0.952 |
| ResNet-34 | Y | 236 s. | 0.969 |
| ResNet-34 | N | 233 s. | 0.966 |
| ResNet-50 | Y | 785 s. | 0.958 |
| ResNet-50 | N | 781 s. | 0.901 |
| ResNet-101 | Y | 1414 s. | 0.941 |
| ResNet-101 | N | 1350 s. | 0.948 |
| ResNet-153 | Y | 2079 s. | 0.682 |
| ResNet-153 | N | 1094 s. | 0.935 |

From the experiments (Table 3) carried out with the standard implementations of ResNet, it can be seen that all implementations perform well. Using gradient centralization improves results in ResNet-50 and ResNet-34. But when architecture is deep, centralization of gradients decreases accuracy. When using the ResNet-153 architecture, the centralization of the gradients significantly reduces the accuracy than without it. This might be an indication that when the differences are small in the data, the vanishing gradient problem becomes a problem with gradient centralization.

Most of the ResNet architectures were prone to overfitting in the given problem, especially architectures deeper than ResNet-34. Overfitting of the model signals that training goes well, but the model fails to generalize to the validation set. In practice, this means that the model learns the training data too well, capturing noise and specifics of the training set that do not generalize to new, unseen data [74].

**Table 3.** *K*-fold accuracy.

| Fold | Architecture (a) | ResNet-34 | Architecture (b) |
|---|---|---|---|
| 0 | 97.66% | 95.31% | 97.85% |
| 1 | 99,51% | 98.93% | 99.22% |
| 2 | 96.58% | 97.66% | 97.17% |
| 3 | 96.29% | 98.73% | 98.54% |
| 4 | 100.00% | 96.58% | 98.63% |
| Final | 98.00% | 97.44% | 98.28% |

The small dataset can also be an issue, as the dataset contains only 7855 images. For extending the dataset, more images could be collected or different augmentation options could be used [75].

Of all architectures tested, ResNet-34 with gradient centralization had the best accuracy, as the model reached 96.9% accuracy. And it was selected for fine-tuning the architecture.

In Figure 10, it can be seen that the training accuracy of ResNet-34 improves throughout the epochs. In addition, training loss decreases during the training process. After 10 epochs, validation accuracy starts to behave unexpectedly and drops, while training accuracy seems to settle. At the end of the training process, the training loss seems to increase. Some of the reasons behind this could be overfitting, a small dataset, inappropriate learning, or simply that the model has already reached its optimal performance. To overcome these and build better models, ResNet-34 was modified.

If the model reaches its peak performance, it might be wise to implement early stopping for the training process; this prevents overfitting and saves computation time when further training does not yield significant improvements [76].

### 4.3. Proposed architecture

The base model of ResNet-34 was able to reach the accuracy of 96.90%, this is already very high, but with some modifications it might be possible to reach even higher accuracy. Some approaches were first experimented individually and finally, all of them were combined.

One of the approaches used in past research to improve CNN is adding a dropout layer to fully connected layers. With the usage of dropout, it is possible to reduce overfitting and regulate neural networks [60]. Proper usage of dropout layers can increase the accuracy of neural networks [77]. The traditional approach to using dropout is to add dropout after the convolutional layer; however, there are different variations of using dropout [78]. In this research, two variations of the use of dropout in the ResNet-34 architecture were used. In the first dropout, it was added after the first convolutional layer. This convolutional layer pools feature maps down. Then, the dropout was added after each residual group. The performance of these architectures was 98.90% and 94.34% accordingly. This shows that dropping out at the correct location in the architecture can make the model perform more accurately.

Adding batch normalization can also make the network more resistant towards degradation of the between-class distance to the within-class distance ratio. With batch normalization, it is possible to perceive the between-class angle [79]. When batch normalization was added at the end of the residual block accuracy of the model dropped 1.0% from standard ResNet-34 implementation.

Another option to improve the accuracy of the CNN is to use different grouping layers. Pooling layers play a crucial role in reducing the spatial dimensions of the feature maps, controlling the overfitting, providing translation invariance, and facilitating feature learning and abstraction in convolutional neural networks [80]. Local pooling, which is used to pool data from smaller regions, or global pooling, which works on the feature map, can be used to improve the accuracy and sensitivity of feature translation [70]. With the usage of different pooling methods, it might be possible to increase the accuracy of CNN architecture, as, for example, in the research by Momeny et al. [81].

In the residual block of ResNet after each convolutional operation, a maximum pooling layer was added, which led to a accuracy of 95.10%. When max-pooling was added only after both convolutional operations were performed, residual block accuracy reached as high as 99.70%. As a last modification to the standard ResNet-34 architecture just before the fully connected layer average pooling was changed to global max pooling instead of average pooling 99.60%

When finally all the best options from the above experiments were combined, using dropout after input layers, the global maximum pooling at the end of the residual block and changing the average pooling to the global maximum pooling accuracy was dropped to 40.52%. This shows that even if optimisations work independently they don't work together. In addition, some combinations were also tested to see if they can reach a higher accuracy than individual modifications. Using global maximum pooling instead of average pooling with dropout performed with 99.20% accuracy. Using global maximum pooling instead of average pooling with a maximum pooling layer at the end of the residual block reached 99.50% accuracy.

For the initial experimentation, without cross-validation, training and validation accuracy and loss, together with confusion matrixes, are shown in Figure 11. For both of the architectures it can be seen that models are learning quite well throughout the epochs, and if 30 epochs are used, early stopping does not stop the training process. Architectures reach maximum accuracy at the end of the process (architecture a 99.70% and b 99.60%). In both cases, the training loss seems to stabilize at the end of the training process.

Some mistakes that version a) makes are that low-level green (10% CY) and yellow (10% Y) images are confused with very low yellow-color (5% Y) images. With this approach, it was possible to reach an accuracy of 99.7%. Version b) of the architecture almost reaches the same accuracy (99.6%). In this architecture, there was more confusion between classes. Different green colors are confused (5% CY and 10% CY), as well as different yellow classes. Also, very light yellow is sometimes confused with paper white.

Based on the experiments, two best options to improve the ResNet-34 architecture were:

- (a) changing the average pooling to the max pooling before fully connected layer.
- (b) using the maximum pooling at the end of the residual block. These architectures are presented in Figure 12.

### 4.4. K-fold cross-validation

The accuracy of both proposed architectures was finally verified with *K*-Fold cross-validation using five folds. For each fold 30 epochs of training were run. The same process was performed for the standard ResNet-34 architecture in order to obtain the accuracy for comparison. The following table summarizes these results. The final accuracy in *K*-Fold cross-validation is calculated by averaging the accuracy scores obtained from each fold.

The results show that architecture (b), where no other changes were made than changing the average pooling to the maximum pooling, is the most accurate of all versions tested. In Figure 13 the training and validation loss through epochs for each fold can be seen. The figures show that model reaches its peak accuracy typically between 20 and 30 epochs, after that model starts to show signs over overfitting.

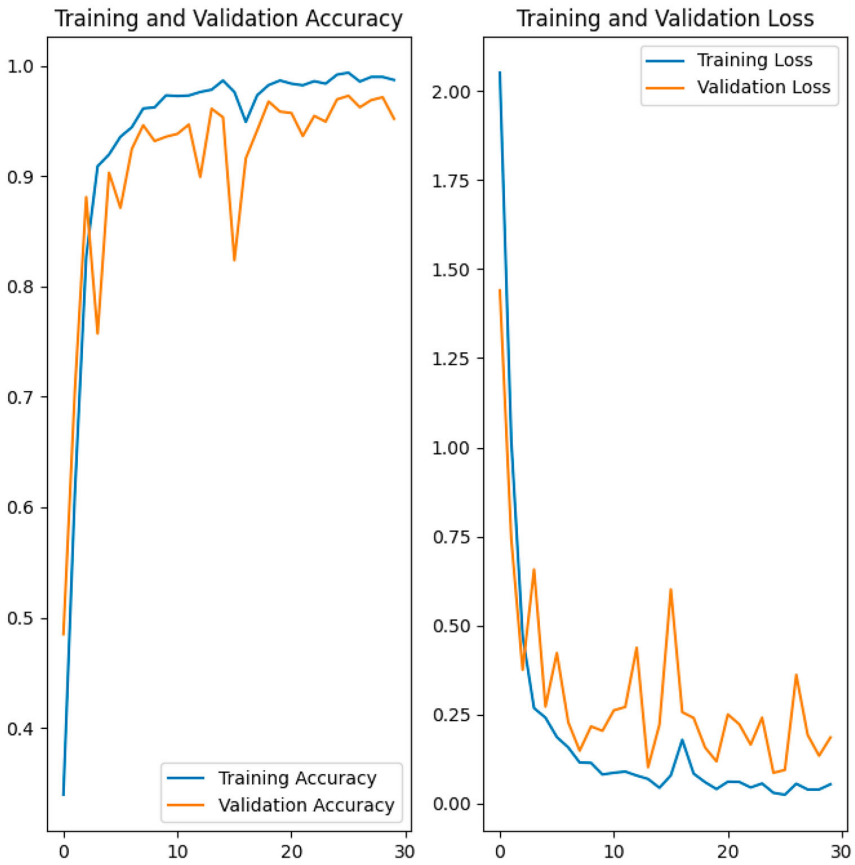



**Figure 10.** Results of the train and validation process of ResNet-34.

Finally, Figure 14 show the combined confusion matrix of *K*-Fold cross-validation. In this figure, values are normalized so that it can easily be observed how well the model perform. The confusion matrix for *K*-Fold cross-validation shows similar results to those without *K*-Fold cross-validation. The most challenging class to identify is 5% Y, this is confused with white paper in 8% of cases, also sometimes 10% and 5% Y are confused. Other classes are rarely confused.

## 5. Discussion

This article presented two optimized versions of ResNet, with the modified version, where a maximum pooling layer was added after convolutional operations, the residual block accuracy was 98.00% in the cross-validation of the *K*-Fold. The second modification was to change the last average pooling layer to the maximum pooling layer, this version of the ResNet reached 98.28% accuracy being the most accurate version. The results show that ResNet can be used to recognize subtle color differences. The mentioned modifications make it even better for the use case, as both versions outperformed the standard ResNet-34 architecture.

With the proposed approach, it is possible to implement a more accurate color-based classification system that can help, for example, in civil engineering [82–84], food production [85–87] and healthcare [88, 89]. In these and some other use cases, color and color differences play an important role.

The findings of this research support previous research like that by Sukhetha et al. and Goudha et al., showing that the ResNet architecture works well when color is a criterion in the classification task [90, 91]. Singh et al. [92] have also shown in their research that convolutional networks are highly color dependent. This dependency can be seen in the presented research; CNNs can learn to classify color even with shallow networks. Modifications to the standard ResNet architecture are a good way to make ResNet suitable for various color recognition tasks; this is in line with previous research such as Mathew et al. [93], Zhang et al. [94] and Reddy et al. [95].

One limitation of this research is that it uses a relatively small dataset of images which are printed with a specific color printer. This might have an impact on the results presented; however, the presented approach can be adjusted to different image sources with more training and possibly by adjusting preprocessing methods for images.

The proposed solution was run with a standard laptop environment; more research would be needed about how to make the system run on mobile devices, if color recognition is wanted to be done in real-time, for example, in the consumer context. The use of extended datasets and different preprocessing algorithms might also make the proposed approach more generalizable. Deploying the system in a server-client infrastructure involves a client-side application, such as a Web or mobile app, for user interaction and image uploads, paired with a server-side back-end hosting the trained ResNet-34 model for pre-processing and image classification. The trained model can be deployed for web with frameworks such as TensorFlow together with the help of Flask. In such an application, continuous training of

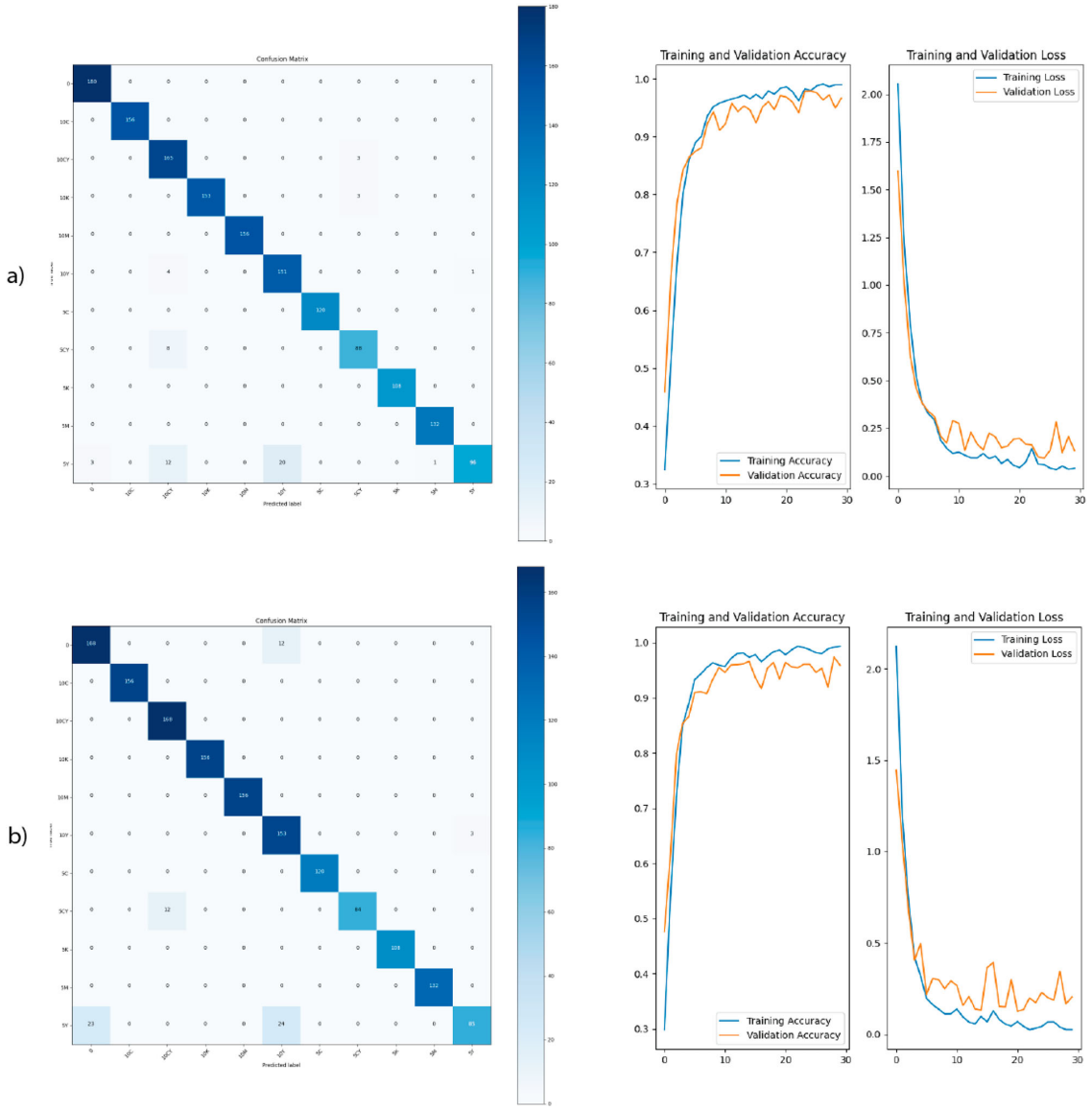



**Figure 11.** Results of the process.

the model might also be a feasible feature and might lead to a more accurate solution.

Future research could also focus on tailoring the model for specific industrial applications, such as automating quality assurance in manufacturing or improving diagnostic capabilities in telemedicine. This involves optimizing the system to meet operational constraints, such as processing speed, scalability, and integration with existing workflows. In preprocessing, exploring alternative preprocessing techniques, such as adaptive histogram equalization or domain-specific adjustments, could further enhance the robustness and reliability of the model under varying lighting and environmental conditions, making it more adaptable to real-world scenarios. In addition, creating diverse datasets with variations in imaging conditions, color ranges, and patterns would help generalize the system to a wider variety of use cases and ensure its effectiveness in different domains.

## 6. Conclusions

The research highlights the effectiveness of different ResNet architectures for subtle color classification and demonstrates how targeted modifications can enhance model performance. By systematically evaluating different ResNet variants, the research identified ResNet-34 as the most suitable baseline model, with gradient centralization further enhancing its accuracy. Two custom versions of ResNet-34 were proposed, achieving a classification accuracy of up to 99.28% through the use of global max pooling instead of global average pooling. The results of the experiment were verified with a 5-fold $K$-Fold cross-validation. These results underscore the flexibility of ResNet architectures and the importance of fine-tuning their components for specific applications. The proposed architectures demonstrate potential for real-world applications in fields like civil engineering, food production, and healthcare, where precise color differentiation plays

| Layer name | output size | custom Resnet - version a |
|---|---|---|
| conv1 | 112 x 112 | 7x7, 64 stride 2 |
| conv2 | 56 x 56 | 3x3 max pool, stride 2<br>[3 x 3, 64<br>3 x 3, 64<br>3x3 max pool, stride 2] x 3<br>[3 x 3, 64<br>3 x 3, 64] x 1 |
| conv3 | 28 x 28 | [3 x 3, 128<br>3 x 3, 128<br>3x3 max pool, stride 2] x 3<br>[3 x 3, 128<br>3 x 3, 128] x 1 |
| conv4 | 14 x 14 | [3 x 3, 256<br>3 x 3, 256<br>3x3 max pool, stride 2] x 5<br>[3 x 3, 256<br>3 x 3, 256] x 1 |
| conv5 | 7x 7 | [3 x 3, 256<br>3 x 3, 256<br>3x3 max pool, stride 2] x 3 |
| | 1 x 1 | average pool, fc, softmax |

| Layer name | output size | custom Resnet - version b |
|---|---|---|
| conv1 | 112 x 112 | 7x7, 64 stride 2 |
| conv2 | 56 x 56 | 3x3 max pool, stride 2<br>[3 x 3, 64<br>3 x 3, 64] x 3<br>[3 x 3, 64<br>3 x 3, 64] x 1 |
| conv3 | 28 x 28 | [3 x 3, 128<br>3 x 3, 128] x 3<br>[3 x 3, 128<br>3 x 3, 128] x 1 |
| conv4 | 14 x 14 | [3 x 3, 256<br>3 x 3, 256] x 5<br>[3 x 3, 256<br>3 x 3, 256] x 1 |
| conv5 | 7x 7 | [3 x 3, 256<br>3 x 3, 256<br>3x3 max pool, stride 2] x 3 |
| | 1 x 1 | maximum pool, fc, softmax |

**Figure 12.** Final architectures used.

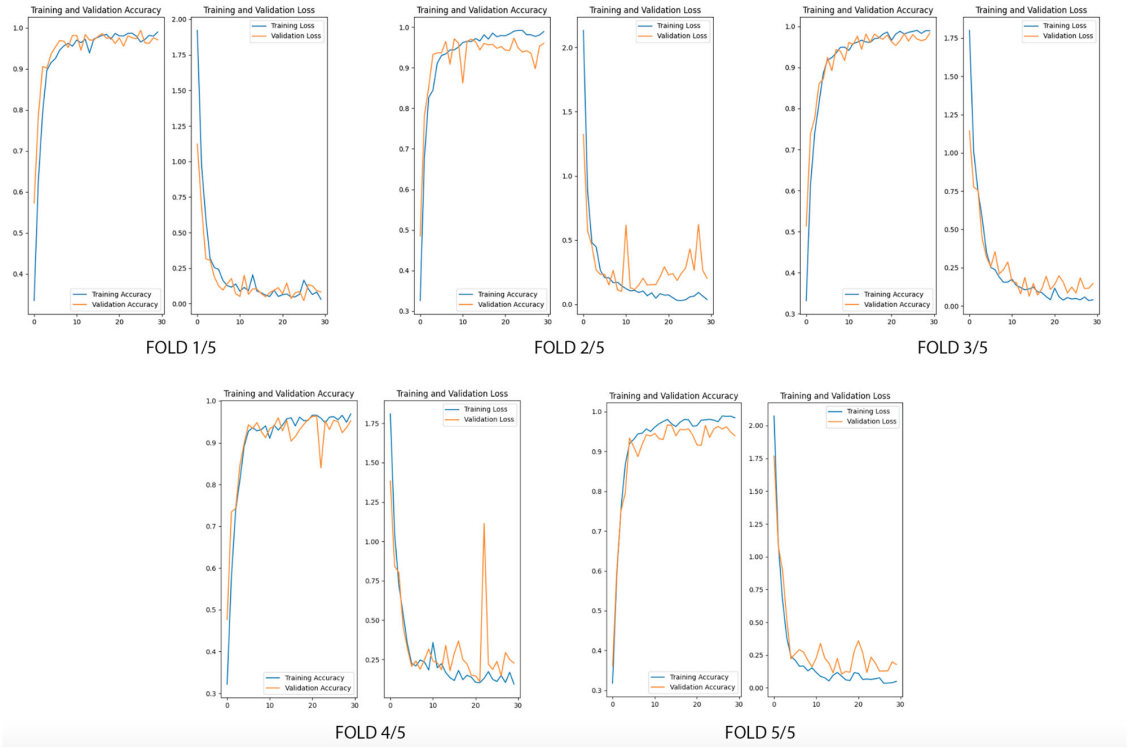



**Figure 13.** Training process of each fold.

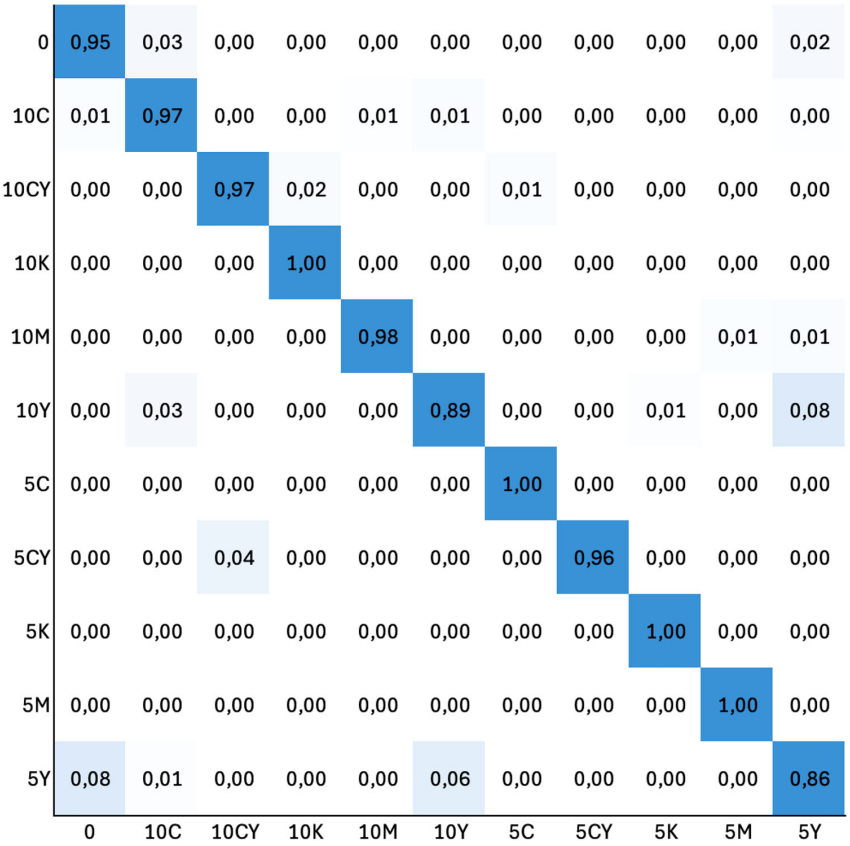



**Figure 14.** Combined confusion matrix of *K*-fold cross-validation.

a crucial role. Some restrictions of the experiment were small dataset, that was augmented and usage of one printer type to print QR-code that worked as color information carries. Future work around the topic could explore broader datasets, alternative preprocessing methods, and real-time implementation on mobile platforms to extend the applicability of the proposed approaches.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

This work was supported by the Finnish Cultural Foundation's Central Ostrobothnia Regional Fund (Suomen Kulttuurirahasto) [grant number 2521 1242].

## Data availability statement

One of the datasets used in this manuscript is available as Zenodo repository: https://doi.org/10.5281/zenodo.11079897.

## Notes on contributor

*Jari Isohanni*, MSc (Computer Science), is currently working with his doctoral studies at the University of Vaasa (Digital Economy). His dissertation compares different approaches in recognition of subtle color differences in printed sources. Jari has been working in the software industry since 2004, currently acting as Director (Education) at Centria University of Applied Sciences.

## ORCID

*Jari Isohanni* http://orcid.org/0000-0002-7154-2515

## References

[1] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA; 2016. p. 770–778.
[2] He K, Zhang X, Ren S, et al. Identity mappings in deep residual networks. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. Springer; 2016. p. 630–645.
[3] He K, Zhang X, Ren S, et al. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA; 2015. p. 1026–1034.
[4] Yang S, Liu S, Yang C, et al. Re-rank coarse classification with local region enhanced features for fine-grained image recognition. Preprint; 2021. arXiv:210209875.
[5] Ailing Q, Ning T. Fine-grained vehicle recognition method based on improved ResNet. In: 2nd International Conference on Information Technology and Computer Application (ITCA) IEEE; 2020. p. 588–592.
[6] Li S, Jiao J, Han Y, et al. Demystifying ResNet. Preprint; 2016. arXiv:16110 1186.
[7] Targ S, Almeida D, Lyman K. Resnet in ResNet: generalizing residual architectures. Preprint; 2016. arXiv:160308029.
[8] Zagoruyko S, Komodakis N. Wide residual networks. Preprint; 2016. arXiv:160507146.
[9] Li B, He Y. An improved ResNet based on the adjustable shortcut connections. IEEE Access. 2018;6:18967–18974. doi: 10.1109/ACCESS.2018.2814605
[10] Yuan P, Lin S, Cui C, et al. HS-ResNet: hierarchical-split block on convolutional neural network. Preprint; 2020. arXiv:201007621.

[11] Hayou S, Clerico E, He B, et al. Stable ResNetA Virtual Conference International Conference on Artificial Intelligence and Statistics; 2021. p. 1324–1332.
[12] Bharati S, Podder P, Mondal M, et al. Co-ResNet: optimized ResNet model for COVID-19 diagnosis from X-ray images. Int J Hybrid Intell Syst. 2021;17(1–2):71–85.
[13] Wen L, Li X, Gao L. A transfer convolutional neural network for fault diagnosis based on ResNet-50. Neural Comput Appl. 2020;32:6111–6124. doi: 10.1007/s00521-019-04097-w
[14] Sarwinda D, Paradisa RH, Bustamam A, et al. Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer. Procedia Comput Sci. 2021;179:423–431. doi: 10.1016/j.procs.2021.01.025
[15] Li B, Lima D. Facial expression recognition via ResNet-50. Int J Cogn Comput Eng. 2021;2:57–64.
[16] Yu X, Kang C, Guttery DS, et al. ResNet-SCDA-50 for breast abnormality classification. IEEE/ACM Trans Comput Biol Bioinform. 2020;18(1):94–102. doi: 10.1109/TCBB.8857
[17] Gao M, Qi D, Mu H, et al. A transfer residual neural network based on ResNet-34 for detection of wood knot defects. Forests. 2021;12(2):212. doi: 10.3390/f12020212
[18] Hammad M, Pławiak P, Wang K, et al. Resnet-attention model for human authentication using ECG signals. Expert Syst. 2021;38(6):e12547. doi: 10.1111/exsy.v38.6
[19] Han C, Shi L. ML–ResNet: a novel network to detect and locate myocardial infarction using 12 leads ECG. Comput Methods Programs Biomed. 2020;185:105138. doi: 10.1016/j.cmpb.2019.105138
[20] Isohanni J. Use of functional ink in a smart tag for fast-moving consumer goods industry. J Packag Technol Res. 2022;6(3):187–198. doi: 10.1007/s41783-022-00137-4
[21] Almoosawi NM, Khudeyer RS. ResNet-34/DR: a residual convolutional neural network for the diagnosis of diabetic retinopathy. Informatica. 2021;45(7):115–124.
[22] Hu WJ, Fan J, Du YX, et al. MDFC–ResNet: an agricultural IoT system to accurately recognize crop diseases. IEEE Access. 2020;8:115287–115298. doi: 10.1109/Access.6287639
[23] Al-Haija QA, Adebanjo A. Breast cancer diagnosis in histopathological images using ResNet-50 convolutional neural network. In: IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Vancouver, BC, Canada. IEEE; 2020. p. 1–7.
[24] Zhang X, Li H, Sun S, et al. Classification and identification of apple leaf diseases and insect pests based on improved ResNet-50 model. Horticulturae. 2023;9(9):1046. doi: 10.3390/horticulturae9091046
[25] Shaheed K, Qureshi I, Abbas F, et al. EfficientRMT-Net – an efficient ResNet-50 and vision transformers approach for classifying potato plant leaf diseases. Sensors. 2023;23(23):9516. doi: 10.3390/s23239516
[26] Li X, Rai L. Apple leaf disease identification and classification using ResNet models. In: 3rd International Conference on Electronic Information and Communication Technology (ICEICT); Shenzhen, China. IEEE; 2020. p. 738–742.
[27] Isohanni J. QR-codes with colour embed inside. Zenodo. 2024. doi:10.5281/zenodo.11079897.
[28] Basuki A, Ramadijanti N. Improving auto level method for enhancement of underwater images. In: Manado International Conference on Knowledge Creation and Intelligent Computing (KCIC); 2016; p. 120–125. doi:10.1109/KCIC.2016.7883635.
[29] Zhang G, Lin L, Wang J. Lung nodule classification in CT images using 3D DenseNet. J Phy Conf Series 2021;1827:012155.
[30] Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge, USA: MIT Press; 2016.
[31] LeCun Y, Boser B, Denker J, et al. Handwritten digit recognition with a back-propagation network. Adv Neural Inf Process Syst. 1989;2:396–404.
[32] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–2324. doi: 10.1109/5.726791
[33] Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10); 2010; p. 807–814.
[34] Goh GB, Hodas NO, Vishnu A. Deep learning for computational chemistry. J Comput Chem. 2017;38(16):1291–1307. doi: 10.1002/jcc.v38.16
[35] Zweiri YH, Whidborne JF, Seneviratne LD. A three-term backpropagation algorithm. Neurocomputing. 2003;50:305–318. doi: 10.1016/S0925-2312(02)00569-6
[36] Wang X, Qin Y, Wang Y, et al. ReLTanh: an activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis. Neurocomputing. 2019;363:88–98. doi: 10.1016/j.neucom.2019.07.017
[37] Rehmer A, Kroll A. On the vanishing and exploding gradient problem in gated recurrent units. IFAC-PapersOnLine. 2020;53(2):1243–1248. doi: 10.1016/j.ifacol.2020.12.1342
[38] Tong T, Li G, Liu X, et al. Image super-resolution using dense skip connections Venice. In: Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 4799–4807.
[39] Ooi YK, Ibrahim H, Mahyuddin MN. Enhanced dense space attention network for super-resolution construction from single input image. IEEE Access. 2021;9:126837–126855. doi: 10.1109/ACCESS.2021.3111983
[40] Haider A, Arsalan M, Choi J, et al. Robust segmentation of underwater fish based on multi-level feature accumulation. Front Mar Sci. 2022;9:1010565. doi: 10.3389/fmars.2022.1010565
[41] Hassan E, Hossain MS, Saber A, et al. A quantum convolutional network and ResNet (50)-based classification architecture for the MNIST medical dataset. Biomed Signal Process Control. 2024;87:105560. doi: 10.1016/j.bspc.2023.105560
[42] Md Rabiul Hasan ABMAH, Ullah SMA. Ensemble ResDenseNet: Alzheimer's disease staging from brain MRI using deep weighted ensemble transfer learning. Int J Comput Appl. 2024;46(7):539–554. doi: 10.1080/1206212X.2024.2380648 .
[43] Senapati B, Talburt JR, Naeem AB, et al. Transfer learning based models for food detection using ResNet-50. In: Romeoville IEEE International Conference on Electro Information Technology (EIT); 2023. p. 224–229.
[44] Wu D, Ying Y, Zhou M, et al. Improved ResNet-50 deep learning algorithm for identifying chicken gender. Comput Electron Agric. 2023;205:107622. doi: 10.1016/j.compag.2023.107622
[45] Lin CL, Wu KC. Development of revised ResNet-50 for diabetic retinopathy detection. BMC Bioinform. 2023;24(1):1–18. doi: 10.1186/s12859-022-05124-9
[46] Madhukar BN, Bharathi SH, Polnaya AM. Multi-scale convolution based breast cancer image segmentation with attention mechanism in conjunction with war search optimization. Int J Comput Appl. 2023;45(5):353–366. doi: 10.1080/1206212X.2023.2212945 .
[47] Zagoruyko S, Komodakis N. Wide residual networks; 2017. Available from: https://arxiv.org/abs/1605.07146 [cs.CV].
[48] Xie S, Girshick R, Dollár P, et al. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 1492–1500.
[49] Wong TT, Yeh PY. Reliable accuracy estimates from *k*-fold cross validation. IEEE Trans Knowl Data Eng. 2019;32(8):1586–1594. doi: 10.1109/TKDE.69
[50] James G, Witten D, Hastie T, et al. An introduction to statistical learning. Vol. 112. New York, USA: Springer; 2013.
[51] Yadav S, Shukla S. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In: IEEE 6th International Conference on Advanced Computing (IACC); 2016. p. 78–83.
[52] Lyu Z, Yu Y, Samali B, et al. Back-propagation neural network optimized by *k*-fold cross-validation for prediction of torsional strength of reinforced concrete beam. Materials. 2022;15(4):1477. doi: 10.3390/ma15041477
[53] Yong H, Huang J, Hua X, et al. Gradient centralization: a new optimization technique for deep neural networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. Springer; 2020. p. 635–652.
[54] Fuhl W, Kasneci E. Weight and gradient centralization in deep neural networks. Preprint; 2020. arXiv:201000866.
[55] Lu F, Niu R, Zhang Z, et al. A generative adversarial network-based fault detection approach for photovoltaic panel. Appl Sci. 2022;12(4):1789. doi: 10.3390/app12041789
[56] Agarwal V, Lohani M, Bist AS. Comparative analysis of deep learning models for various optimizer embedded with gradient centralization. Int J Intell Syst Appl Eng. 2024;12(15s):445–454.
[57] Liu Z, Xu Z, Jin J, et al. Dropout reduces underfitting. In: International Conference on Machine Learning. PMLR; 2023. p. 22233–22248.
[58] Hahn S, Choi H. Understanding dropout as an optimization trick. Neurocomputing. 2020;398:64–70. doi: 10.1016/j.neucom.2020.02.067
[59] Wu H, Gu X. Towards dropout training for convolutional neural networks. Neural Netw. 2015;71:1–10. doi: 10.1016/j.neunet.2015.07.007
[60] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–1958.
[61] Wu H, Gu X. Max-pooling dropout for regularization of convolutional neural networks. In: Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9–12, 2015, Proceedings, Part I 22. Springer; 2015. p. 46–54.
[62] Garbin C, Zhu X, Marques O. Dropout vs. batch normalization: an empirical study of their impact to deep learning. Multimed Tools Appl. 2020;79(19):12777–12815. doi: 10.1007/s11042-019-08453-9
[63] Ko B, Kim HG, Oh KJ, et al. Controlled dropout: a different approach to using dropout on deep neural network. In: 2017 IEEE International

Conference on Big Data and Smart Computing (BigComp). IEEE; 2017. p. 358–362.
[64] Skourt BA, El Hassani A, Majda A. Mixed-pooling-dropout for convolutional neural network regularization. J King Saud Univ Comput Inf Sci. 2022;34(8):4756–4762.
[65] Khan SH, Hayat M, Porikli F. Regularization of deep neural networks with spectral dropout. Neural Netw. 2019;110:82–90. doi: 10.1016/j.neunet.2018.09.009
[66] Shirke V, Walika R, Tambade L. Drop: a simple way to prevent neural network by overfitting. Int J Res Eng Sci Manag. 2018;1:2581–5782.
[67] Hou W, Wang W, Liu RZ, et al. Cropout: a general mechanism for reducing overfitting on convolutional neural networks. In: 2019 International Joint Conference on Neural Networks (IJCNN). IEEE; 2019. p. 1–8.
[68] Poernomo A, Kang DK. Biased dropout and crossmap dropout: learning towards effective dropout regularization in convolutional neural network. Neural Netw. 2018;104:60–67. doi: 10.1016/j.neunet.2018.03.016
[69] Bieder F, Sandkühler R, Cattin PC. Comparison of methods generalizing max-and average-pooling. Preprint; 2021. arXiv:210301746.
[70] Zafar A, Aamir M, Mohd Nawi N, et al. A comparison of pooling methods for convolutional neural networks. Appl Sci. 2022;12(17):8643. doi: 10.3390/app12178643
[71] Chollet F. Deep learning with python. Shelter Island: Simon and Schuster; 2021.
[72] Han J, Kamber M, Pei J. 2 – getting to know your data. In: Han J, Kamber M, Pei J, editors. The Morgan Kaufmann series in data management systems. Boston: Morgan Kaufmann; 2012. p. 39–82. doi: 10.1016/B978-0-12-381479-1.00002-2
[73] Japkowicz N, Shah M. Evaluating learning algorithms: a classification perspective. Cambridge: Cambridge University Press; 2011.
[74] Ying X. An overview of overfitting and its solutions. In: Journal of Physics: Conference Series. Vol. 1168. IOP Publishing; 2019. p. 022022.
[75] Brigato L, Mougiakakou S. No data augmentation? Alternative regularizations for effective training on small datasets. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2023. p. 139–148.
[76] Prechelt L. Early stopping-but when? In: Neural networks: tricks of the trade. Heidelberg: Springer; 2002. p. 55–69.
[77] Ba J, Frey B. Adaptive dropout for training deep neural networks. Adv Neural Inf Process Syst. 2013;26.
[78] Cai S, Shu Y, Chen G, et al. Effective and efficient dropout for deep convolutional neural networks. Preprint; 2019. arXiv:190403392.
[79] Furusho Y, Ikeda K. Resnet and batch-normalization improve data separability. In: Asian Conference on Machine Learning. PMLR; 2019. p. 94–108.
[80] Murray N, Perronnin F. Generalized max pooling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2014. p. 2473–2480.
[81] Momeny M, Jahanbakhshi A, Jafarnezhad K, et al. Accurate classification of cherry fruit using deep CNN based on hybrid pooling approach. Postharvest Biol Technol. 2020;166:111204. doi: 10.1016/j.postharvbio.2020.111204
[82] Lehmann MK, Nguyen U, Allan M, et al. Colour classification of 1486 lakes across a wide range of optical water types. Remote Sens. 2018;10(8):1273. doi: 10.3390/rs10081273
[83] Hu H. Research on colour recognition sorting method of waste plastic bottles based on computer perspective. In: 2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM). IEEE; 2022. p. 914–918.
[84] Petus C, Waterhouse J, Tracey D, et al. Using optical water-type classification in data-poor water quality assessment: a case study in the torres strait. Remote Sens. 2022;14(9):2212. doi: 10.3390/rs14092212
[85] Wei X, Bohrer B, Uttaro B, et al. Centre pork chop colour classification using image analysis on the ventral surface of the loin. Can J Anim Sci. 2023:123–126.
[86] Pegalajar M, Ruiz L, Criado-Ramón D. Munsell soil colour classification using smartphones through a neuro-based multiclass solution. AgriEngineering. 2023;5(1):355–368. doi: 10.3390/agriengineering5010023
[87] Reyes JF, Contreras E, Correa C, et al. Image analysis of real-time classification of cherry fruit from colour features. J Agric Eng. 2021;52(4). doi: 10.4081/jae.2021.1160
[88] Göksel Duru D, Alobaidi M. Classification of brain electrophysiological changes in response to colour stimuli. Phys Eng Sci Med. 2021;44(3):727–743. doi: 10.1007/s13246-021-01021-2
[89] van Minderhout HM, Joosse MV, Grootendorst DC, et al. Eye colour and skin pigmentation as significant factors for refractive outcome and residual accommodation in hypermetropic children: a randomized clinical trial using cyclopentolate 1% and tropicamide 1%. Acta Ophthalmol. 2022;100(4):454–461. doi: 10.1111/aos.v100.4
[90] Sukhetha P, Hemalatha N, Sukumar R. Classification of fruits and vegetables using ResNet model. agriRxiv; 2021. 20210317,450.
[91] Gouda N, Amudha J. Skin cancer classification using ResNet. In: 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA); 2020. p. 536–541. doi: 10.1109/ICCCA49541
[92] Singh A, Bay A, Mirabile A. Assessing the importance of colours for CNNs in object recognition. Preprint; 2020. arXiv:201206917.
[93] Mathew MB, Surya Manjunathan G, Gokul B, et al. Banana ripeness identification and classification using hybrid models with ResNet-50, VGG-16 and machine learning techniques. In: Machine Intelligence Techniques for Data Analysis and Signal Processing: Proceedings of the 4th International Conference MISP 2022. Vol. 1. Springer; 2023. p. 259–273.
[94] Zhang C, Xia K, Feng H, et al. Tree species classification using deep learning and RGB optical images obtained by an unmanned aerial vehicle. J For Res. 2021;32(5):1879–1888. doi: 10.1007/s11676-020-01245-0
[95] Reddy SR, Varma GS, Davuluri RL. Resnet-based modified red deer optimization with DLCNN classifier for plant disease identification and classification. Comput Electr Eng. 2023;105:108492. doi: 10.1016/j.compeleceng.2022.108492