Received: 23 August 2023 | Revised: 4 December 2023 | Accepted: 20 December 2023

DOI: 10.1002/eng2.12838

RESEARCH ARTICLE

Engineering Reports Open Access WILEY

TL-PBot: Twitter bot profile detection using transfer learning based on DNN model

Maryam Bibi[1] | **Zahid Hussain Qaisar**[1] | **Naeem Aslam**[1] | **Muhammad Faheem**[2] | **Perveen Akhtar**[3]

[1]Department of Computer Science, NFC Institute of Engineering and Technology, Multan, Pakistan

[2]Department of Computing Science, School of Technology and Innovations, University of Vaasa, Vaasa, Finland

[3]Department of Physics, Govt. Associate College for Women, Nawab Pur, Multan, Pakistan

Correspondence
Muhammad Faheem, Department of Computing Science, School of Technology and Innovations, University of Vaasa, Vaasa, 65200, Finland.
Email: muhammad.faheem@uwasa.fi

Abstract

Online social networks (OSNs) have reduced global boundaries, with Twitter enabling perspective sharing. Bot profile-propagated false information misuse raises serious concerns. Considering this issue, we present our research on classifying Twitter accounts as "human" or "bot" using deep neural networks and transfer learning. Our proposed approach, TL-PBot, stands for bot profile detection using transfer learning. The TL-PBot framework utilizes Twitter account metadata such as follower count. Our TL-PBot also incorporates text data from the Twitter description field as a feature. Word representation of the text data is achieved using Global Vectors (GloVe), a pre-trained model. By employing user profile-based features, we significantly reduce the overhead of feature engineering. The hybrid nature of the model enables it to effectively handle mixed-type features, including text, binary, and numerical data. We design the network using long-short-term memory (LSTM) units. DNN model layers were trained, and the weights of the pre-trained model layers were frozen to apply the transfer learning, resulting in reduced training time and improved bot profile detection accuracy. The performance of the proposed TL-PBot is evaluated using publicly available datasets. The proposed approach is trained and tested on the same datasets and further evaluated on the validation datasets that were not used in the training phase, which is also a novelty in our approach. Comparative analysis with state-of-the-art approaches demonstrates that the TL-PBot approach achieves a higher accuracy of 98.07%, while excelling in precision of 99%, recall of 98%, f measure of 98.32%, and AUC of 0.99. Employing the transfer learning strategy resulted in an accelerated detection rate of 5.04 milliseconds, attesting to the effectiveness of this approach in enhancing computational efficiency.

KEYWORDS

Bot profile detection, deep learning, DNN model, GloVe embedding, transfer learning, Twitter

1 | INTRODUCTION

The emergence of technology, such as broadband internet and smartphones, has increased the demand for OSNs. OSN platforms have gained widespread popularity among internet users to connect and engage with others.[1] The quantity of multimedia content shared on social networks has significantly increased, and it is quickly disseminated and viewed by numerous users.[2] Twitter and Facebook, like social media platforms, have gained popularity and dominance in recent years due to the benefits of interchanging and publishing vast volumes of multimedia content across the network. By Kemp study,[3] globally, the number of social media users has exceeded 4.2 billion, representing a significant proportion of the world's population, accounting for more than 53%. Monthly active users on Twitter are about 330 million.[4] While Facebook, which is considered the biggest OSN globally, has 2.8 billion active users every month. Twitter has the power to impact individuals as it provides an inclusive medium for the free expression of opinions. OSNs influence people, which results in social bots or machine accounts exposure. In OSNs, bot detection approaches are classified as Crowd-sourcing, Graph-based,[5] and machine learning methods.[6]

According to the study by CMU researchers,[7] during the COVID-19 pandemic, approximately 200 million tweets related to the Coronavirus were discussed, of which approximately 82% were bot retweets. Moreover, bots were categorized into the following:

1. **Feed**: Bots that publish or retweet news about a particular issue.
2. **Quote**: Twitter bots that post famous person quotes.
3. **Template**: Bots that adhere to a defined template.
4. **Advanced**: Sophisticated machine learning techniques are used by bots to produce language that seems like a human.

Work by the Pew Research Center[8] studied that bot accounts shared about 66% of tweeted links to some of the most well-liked websites. Analysis of 3,79,341 tweeted links to 925 popular and current news or event websites collected from July 27 to September 11, 2017. As per their study, about two-thirds of estimated tweeted links to popular websites are advertised by bot accounts and not by humans. The most active Twitter bots produce extensive links shared to current and popular news or events websites. The statistic in this study is given in (Figure 1).

The impact of social bots on two significant events of 2020, the presidential election of the United States in 2020 and the COVID-19 pandemic was investigated.[9] It was observed that elevated proportions of tweets generated by bots were related to a specific political line. Some bot tweets favored that specific political line and some were against it. Social bots can also significantly contribute to the spread of disinformation about climate change, which in turn could undermine support for policies aimed at mitigating the effects of rising temperatures.[10]

During the COVID-19 pandemic in Mumbai,[11] social media bots were found to be responsible for posting dubious and false information about the pandemic, and rumors about vaccines circulated on social media like Twitter that the Indian government was accused of conspiring to use vaccines to harm children of Muslims, resulting in only vaccinating half of those expected to receive them.

Some social bots have a positive impact, and some have a negative impact. The positive impact social bots act as helpers in auto-responders for customer care or in conveying news feeds. But negative impact social bots have been misused to

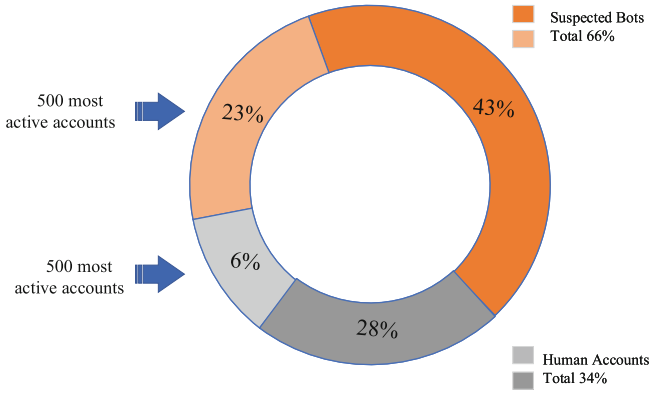

FIGURE 1 Involving both humans and bots, the proportion of tweeted links to popular websites.

build out fake news and rumors to mislead the public. As an online OSN, Twitter provides a platform for expressing one's interpretation in public. The user's online posts on the Twitter are used to construct their profile, which includes personal information like age, education, gender, and personality.

Bot-created accounts misuse their great potential by manipulating opinions and spreading false news or tweets.[12] Also, bots reduce the realistic number of tweets, retweets, likes, news, etc., from which false trends go viral and affect our public. Bot accounts also promote illegal ads and spread malicious URLs.[13] Hence, differentiating between genuine accounts and bot accounts becomes a hot topic for researchers due to the difficulty in accurately identifying them and becoming a privacy threat for the public.[14]

Artificial intelligence advancements have enabled social bots to produce messages that resemble those written by humans, increasing their success.[1] If we look through the same lens, then it is challenging to find spam bots because bots pose different behaviors.[15] For instance, numerous products are generated by spam bots that traditionally promote content. Social media accounts with fake followers and spam bots usually exhibit strong pursuit patterns, with the former being prone to attack and the latter supporting political candidates. Most work is centralized on particular behavioral patterns of bot detection, which makes it challenging to identify all the categories of bots.[16] A generalized method for detecting bots is needed for real-time bot detection. Methods using only user profile metadata information are primarily trained using Adaboost classifiers or Random Forests.[17]

This study presents an innovative framework, TL-PBot, for identifying automated accounts, commonly referred to as "bots," within the Twitter platform. TL-PBot is a transfer learning and deep neural network-based solution engineered to achieve a high degree of generalizability, allowing it to detect bots effectively even when applied to previously unseen datasets. Our approach relies exclusively on the user profile information extracted from Twitter accounts, obtained explicitly from the Twitter Application Programming Interface (API). (Figure 2) presents an overview of the TL-PBot framework, which comprises several modules. The framework involves vital components such as data preparation, feature engineering, training the DNN model, and applying transfer learning, which we do not need to prepare the model from scratch. The processed data are leveraged to classify bot and human accounts through these well-defined steps. The details of the proposed approach are discussed in section 3.3.

The issue is that with the wild expansion of Artificial Intelligence (AI) algorithms, it becomes a significant challenge to investigate the identification of bot accounts or dubious sources. It raised numerous publications and studies in deep transfer learning to differentiate human accounts from bot accounts. Also, there is an issue with the high usage of resources to train the model, which consumes a lot of time.

The main goals and objectives of the research are listed below:

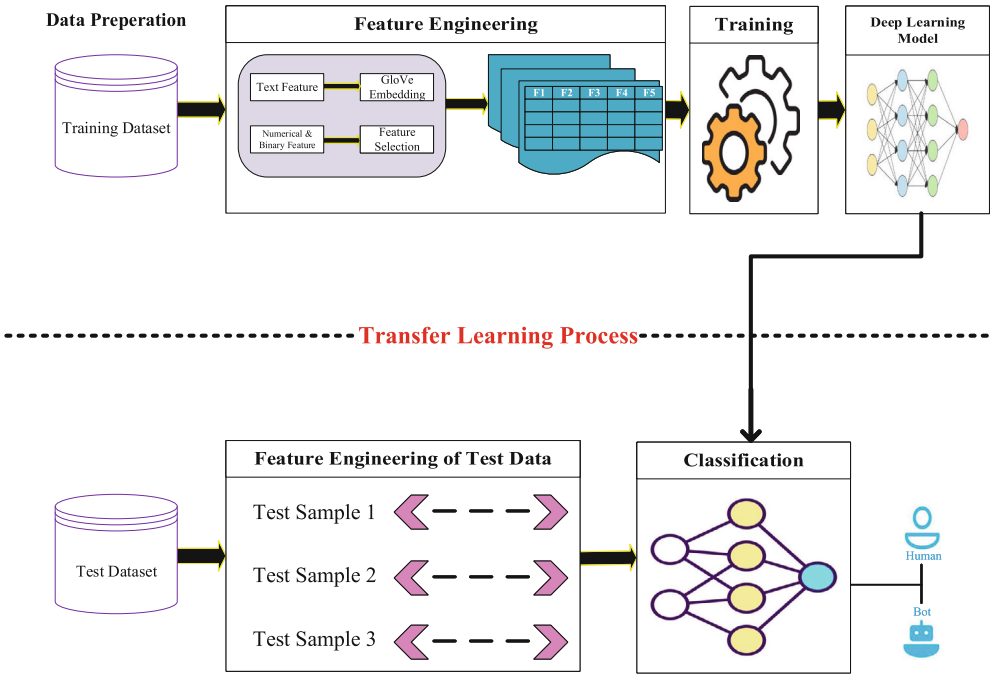

FIGURE 2 Overview of the TL-PBot framework.

1. To describe the importance of bot detection.
2. To help the public distinguish authentic human accounts from bot accounts.
3. To refine the standard or quality of experience of consumers in such forums by increasing the reliability of the forum itself and simultaneously bringing down their privacy risks.
4. To improve DNN performance, reduce the training time and the utilization of resources using the transfer learning approach.
5. To achieve high accuracy in detecting bot accounts from benign and fake accounts samples by employing multiple approaches.

The following are the main goals of applying transfer learning at the initial stage:

Knowledge utilization: New target tasks can be trained using the source model's knowledge. As a result, no training is needed to build the new target model from scratch.

Computation cost: To train complex and hybrid data sets, deep learning models require high computing power.

Model development time: Only the final few data set-specific layers must be trained, substantially decreasing the overall time needed to create and train a new model.

Over-fitting problems: When deep learning models and classical machine learning are trained on a limited data set, over-fitting problems might arise. One solution to this problem is to employ transfer learning by adjusting the model layers through fine-tuning.

The proposed work makes main contributions to the current state-of-the-art in the following ways:

1. We have improved Twitter bot profile detection accuracy by putting forward a framework built upon the transfer learning model.
2. With the help of transfer learning, the model trained more quickly and efficiently while consuming fewer resources.
3. We evaluate TL-PBot on a variety of publicly accessible datasets, which are also used by the comparative method, to prevent data bias.
4. The model's accuracy and clarity of performance have been evaluated and improved by exclusively utilizing the profile features.
5. We have performed a cross-domain performance evaluation of TL-PBot by training the model on a set of heterogeneous Twitter datasets and performing testing on it. Subsequently, the model's performance was rigorously evaluated by conducting tests on four additional validation datasets not included in the training phase.

This article will follow the literature review (Section 2), proposed methodology (Section 3), results and their discussion (Section 4), and conclusions and future work (Section 5) at the end.

2 | LITERATURE REVIEW

This part of the article discusses earlier frameworks or techniques for detecting the bot. As we develop the model for bot profile detection, we also talk about the gaps in the current literature and how we can overcome them.

2.1 | Twitter bots

Twitter is widely recognized as a convenient platform for research purposes due to its frequent utilization of Twitter data. Additionally, it is one of the largest platforms for micro-blogging.[18] Twitter has experienced a growing user base across various age groups in the past ten years, functioning as a microblogging platform. Users on Twitter share their thoughts through tweets and engage with others in conversations. Individuals can follow their preferred politicians, celebrities, athletes, and friends and gain followers themselves.[19] Additionally, Twitter provides a daily compilation of popular subjects called trending topics, enabling users to stay informed about ongoing discussions.

Twitter presents unique set of difficulties when identifying and comprehending event-related details due to its concise nature and wide range of content.[20] Human accounts are labeled based on clusters with standard retweet patterns, and those with suspicious retweet patterns are labeled as bots. Applications that can carry out predefined tasks in a repetitive manner are commonly referred to as bots, agents, or actors.[21] Social media bots, sometimes called spam bots, are a unique

bot class that impersonates humans while hiding behind social network accounts, communicating with users, and inciting them, whether for ideological, political, or business reasons, users use them.[22]

Bots help reduce human efforts. On Twitter, bots have reduced the efforts of humans by tweeting, retweeting, and spreading any news.[23] Bots are also helpful for advertisement purposes by spreading any online company or business advertisement campaigns.[24] But some people use it falsely by spreading fake news through tweets, retweets, and false trends. Recently, in the US presidential election 2020 and the Indonesia presidential election,[25] bots actively participated by generating higher volumes of tweets on a specific political line. During the COVID-19 pandemic, bots were found responsible for spreading vaccine-related rumors on Twitter. Also, counter-vandalism bots are revolutionizing the process of identifying and banning vandals on Wikipedia, yet their valuable contributions are frequently overlooked and rarely discussed.

Detection of bots is necessary to control fake, false, or harmful news and trends.[26] False popularity and trends go viral, affecting humans' lives. It affects the population and guides them in the wrong direction. So, we have to detect bots and take all possible actions. To highlight the realistic number of tweets, retweets, likes, accounts, and news. Because bots have productive work, if bots are utilized positively. As per study,[27] social bots generate content on social media and interact with human beings automatically. A computer algorithm trying to alter or emulate their behavior. The classifier with the highest bot score determines the class of the bot.

2.2 | Role of transfer learning

When knowledge gained in one area is used to enhance generalization in another, this is referred to as transfer learning or domain adaptation. It is the process of learning a new model more effectively by applying what has already been learned about a previous one.[28] It is helpful, especially in data science, because most real-world scenarios don't require training complex models using millions of labeled data points. It is currently quite popular in deep learning due to its ability to train DNN (deep neural network) with a minimal dataset. With transfer learning, we apply the knowledge model learned to a new task to fully understand the concepts more effectively. Weights are automatically transferred from a network that completed the new "task B" to a network performing "task A."

Neural networks commonly focus on the initial layer to detect edges, the middle layer to identify shapes, and the final layer to capture task-specific features. In transfer learning, only the initial and middle layers are utilized, while the later layers undergo retraining. This approach leverages labeled data from the original task on which the network was trained. The three significant benefits of transfer learning are:

1. It reduced the time of training.
2. Transfer learning requires a small amount of training data.
3. Enhance the performance of the Neural Network (in most cases).
4. Low requirement of computational resources.

In cases where training a neural network from scratch requires a large amount of data that may only sometimes be available, transfer learning can be a helpful approach. Transfer learning offers a functional machine learning model when the model is already trained on a small training data set. In the Natural Language Processing (NLP) model requires an advanced level of skills or specialized knowledge to handle the massive amount of labeled datasets. Moreover, once the DNN model is created from scratch, it takes weeks or days to complete the complicated task, shortening the training time.

We can apply transfer learning when we don't have enough labeled data to train our model or when a model already trained on related data and tasks is available. If TensorFlow was used to train the initial model, then we could recover it and retrain a few layers to suit our needs. On the other hand, transfer learning only functions if the skills acquired in the initial task are general, implying they can be used in different contexts. Transfer learning falls into three categories:

1. ***Inductive Transfer Learning***: When labels are available only in the destination and unavailable in the source domain. The tasks at the source and the target are related but distinct. Inductive transfer learning is further divided into two subcategories:

   - **Multi-task Learning**: When labels are available in both source and target domain.
   - **Self-taught Learning**: When labels are unavailable in the source domain and only present in the target domain.

2. ***Transductive Transfer Learning***: When labels are unavailable in the destination domain and only present in the source domain.
3. ***Unsupervised Transfer Learning***: Where no labels are available in the source or destination domain.

In our proposed work, we will use the inductive transfer learning subcategory of multi-task learning with the deep learning technique. We will detect malicious activities on Twitter to find out the bot profiles.

Twitter plays a vital role in shaping public perspectives and debates. However, Twitter's automated bot profiles pose a massive threat to the reliability and authenticity of online user interactions, which can influence conversations and disseminate wrong information. Accurately detecting these Twitter bot accounts is crucial for preserving the platform's integrity.

Our research is motivated by an unwavering responsibility to handle this challenge head-on. By tackling the power of deep neural networks, we aim to take apart the complexity of Twitter bot profiles explicitly by analyzing the text in their profile descriptions and features. These deep learning frameworks have an established history of finding hidden patterns inside data, which makes them an invaluable resource in the quest to expose bots.

Furthermore, our inspiration is established in acknowledging that the shortage of labeled data for Twitter bot detection makes it a demanding task. Transfer learning becomes essential, empowering us to use previously stored knowledge in pre-trained models. The ability of transfer learning models to be flexible is vital for handling the unpredictable behavior of Twitter bots. Models must be able to adapt and learn from new patterns since bots constantly change to resemble human interactions and adjust their tactics to avoid discovery. Models that use transfer learning can update their comprehension of bot behavior without needing significant retraining, increasing their resistance to the ever-changing strategies used by malicious actors.

Our work is far more than just an academic pursuit; it is a crucial contribution to advancing social media platforms and safeguarding online debate. We want to make it easier for researchers, social media platforms, and everyday individuals to trust what they see and post online. By doing so, we hope to make the Internet a more secure place by reducing the dissemination of false information and fraud. Our contribution is to make the Internet a more knowledgeable, reliable, and friendly place for everyone.

So, Table 1 details the studies examined, including the authors' approach and dataset choices.

The DeeProBot [14] employed a methodology for choosing the best training data subset to generate a highly generalizable model. Dense layers of DNN and long short-term memory (LSTM) were utilized for the classification job, and the model was trained using various combinations of training datasets. Based on the performance results of cross-domain and cross-validation, DeeProBot discovered an efficient dataset combination for the training. We could replicate their findings by utilizing the identical dataset and collection of features. DeeProBot obtained the data from varol-icwsm by utilizing Twitter's API.

The presented research introduces an innovative framework, DeeProBot, leveraging deep learning techniques to identify bots by analyzing user profile metadata on the Twitter platform. The elimination of extra features, specifically interaction- and content-based features, is proposed to enhance the model's performance. It is imperative to note the continuous evolution of bots, exhibiting novel behavioral patterns. Consequently, a critical need arises for a model capable of adapting to and detecting these emerging behavioral trends. Regrettably, the current model lacks the capability to identify such new patterns.

TABLE 1 Related work summary.

Approach	Year	Dataset	Methodology	Accuracy%	Precision%	Recall%	F Measure%	AUC
DeeProBot[14]	2022	Botometer Repository	Hybrid DNN using LSTM	92.00	93.00	92.00	93.00	0.97
Bot-MGAT[29]	2022	Twibot20 Climate & Russia Hashtag	GAT using Transfer Learning	97.80	97.00	97.00	98.00	0.98
DNN-BD[30]	2018	Cresci(2017)	DNN using Contextual LSTM	96.30	96.00	96.00	96.00	0.96
GANBot[31]	2021	Cresci(2017)	GAN using Contextual LSTM	95.10	94.00	92.00	96.00	0.99

Bot-MGAT, a multi-view graph attention mechanism-based framework,[29] utilizing a transfer learning approach, is presented for predicting social bots. Despite the promising results, the research identifies critical challenges, such as computational costs associated with large social networks and the sensitivity of behavior simulation to scalability measurements. Additionally, concerns are raised regarding the sustainability of the transfer learning approach and potential performance deterioration. Future research should focus on developing strategies to mitigate computational costs in the context of large networks, refining scalability measurements for behavior simulation, and addressing sustainability and performance issues associated with the transfer learning approach.

The deep neural network model for bot detection, which we call DNN-BD, is the integration of a novel contextual LSTM architecture[30] article demonstrates that DNN-BD holds promise for tweet-level detection by leveraging both content and metadata. However, a critical research gap arises as the study acknowledges the substantial time investment in model development, which is a resistance to practical implementation. Simultaneously, the outlined future plans for the open-source framework highlight a separate gap, the need for more detailed methodologies and metrics to comprehensively evaluate the interference of bots in diverse social media conversations. Bridging these gaps is crucial for efficient bot detection, offering practical insights into their evolving capabilities and impact on public discourse.

Where GANBot,[31] which developed a novel framework by adapting the Generative Adversarial Network (GAN) concept, is a semi-supervised technique that can capture the behavioral patterns of the data, wherein an LSTM layer acts as a shared channel between the generator and classifier. While the GANBot framework effectively utilizes GAN technology to extract behavioral patterns from bot samples, a critical research gap is identified in its direct applicability to the bot detection task. Specifically, the GAN, acting as a generator mimicking human behavior, introduces complexities that need to be more seamlessly aligned with the objectives of bot detection. The overfitting problem further complicates the scenario, prompting the need for a more detailed exploration of how overfitting manifests within the GANBot framework. The article addresses this challenge through transfer learning, fine-tuning model layers for improved generalization. However, a more comprehensive understanding of how GANBot's unique features impact the bot detection algorithm is essential for refining its applicability and addressing potential limitations.

3 | PROPOSED METHODOLOGY

This section discusses datasets used in the TL-PBot proposed work, how the model performs feature engineering on the numerical and binary datasets, the text feature preparation using the GloVe for word representation, model architecture, and TL-PBot proposed work.

3.1 | Datasets

We used the dataset that was made publicly available by the bot repository of the Botometer.* Table 2 shows the list of all the used datasets with brief descriptions. The training set consists of icwsm-varol, botometer-feedback, cresci-17, political-bots, and celebrity datasets. We adopted the same dataset combination adopted by the DeeProBot article.[14,35] The model is trained using 80% of the datasets, and the remaining 20% is utilized for testing purposes. We replicated their findings by utilizing identical datasets and employing the same feature sets with a deep neural network classifier. The dataset is made up of heterogeneous datasets, which allows for the inclusion of bots that evolved with different behavioral patterns at different periods. They used the data selection technique to construct a more accurate and generalizable model to choose the best data subset.

Regarding cross-domain testing, it is essential to keep validation datasets in mind. These datasets are separate from the rest of the training datasets. Midterm-18, gilani-17, verified, botwiki, and cresci-rtbust are validation datasets. We combined the verified dataset with the botwiki dataset for balancing between humans and bots. Each dataset is collected at various time slots and labeled using different methods and strategies, which makes these datasets characteristically different from each other, as studied in DeeProBot.[14,35]

3.2 | Feature engineering

To identify optimal features, Alothali et al.[38] propose a hybrid feature selection method that evaluates metadata features of profiles using machine learning algorithms such as neural networks, random forests, support vector machines, and

TABLE 2 List of datasets and their description.

Dataset	Description	#Human	#Bot
Icwsm-varol[32]	We collected a sample of accounts that were manually labeled and drawn from various Botometer score ranges	1471	674
Botometer-feedback[22]	The collected data was obtained through manual labeling of the accounts that were identified by the responses from users of Botometer	379	139
Cresci-17,[33,34]	The dataset consists of four account categories, including social spambots fraudulent followers, traditional spambots, and real users	3474	10,894
Political-bots[22]	Data sourced from Twitter users' political bot posts	0	61
Celebrity[22]	Data sourced from celebrity accounts	5917	0
Verified[35]	Data collected from verified accounts via streaming API	1986	0
Botwiki[35]	Data sourced from bot accounts that self-identified in the archive of botwiki.org	0	697
Midterm-18[35]	Data sourced from political tweets during the 2018 midterm elections in the US	8092	42,445
Gilani-17[36]	Data collected from Twitter API, grouped and manually labeled the accounts	1413	1089
Cresci-rtbust[37]	Manually labeled dataset of Italian retweets, from June 17 to 30, 2018	339	353

naïve Bayes. The metadata represents extracted features, which are extracted from the Twitter account of the user profile account,[39] as listed in Table 3. The data were obtained from Twitter through the Twitter API. The Twitter API's user object provides the metadata on a user's profile. In contrast, the timeline object provides information on their timeline, which involves mentions and recent tweets of the user. We took only the features of the user profiles instead of considering various feature types to primarily find out the only contribution of profile information for bot detection. We ignore the extra overhead of Twitter data extraction.[14]

Additionally, from the Bot Repository, user profile features for all our datasets were available to download directly, except icwsm-varol. Unlike work done by others that uses the metadata feature from a user account, we consider the user profile description text in our TL-PBot proposed framework. The inspiration for the derivations came from the work of Yang et al.,[35] Hayawi et al.,[14] and Inuwa-Dutse et al.[40] Here are some of the extracted features that are explained; the descriptions of the remaining aspects may be found in Table 3. Temporal features of the user's profile can also be extracted from Twitter.[41]

Based on the bigram character combination of the account's *name_on_screen*, we computed the derived feature, *name_on_screen_freq*.

$$name\_on\_screen\_freq = \frac{\sum_{i=1}^{T} N(b_i)}{T}, \tag{1}$$

where

b_i: i^{th} bigram in *name_on_screen*.
N(b_i): total number of a specified bigram.
$\sum_{i=1}^{T}$: represents a summation, where "i" is a variable that starts at 1 and goes up to T.
T: count number of the exclusive character bigrams in equation (1).

Entropy of a specific character pattern after calculation gives name_entropy, name_on_screen_Entropy, and description_of_entropy. A sequence i's entropy can be calculated using,

$$E(l) = \frac{H(l)}{|l|}, \tag{2}$$

where in Equation (2)

$|l|$: is the sequence length.

TABLE 3 List of features used and their description.

	Feature name	Type of feature	Description
Raw features	Description	Text	The user-defined account description text
	Total_statuses	Numeric	User's total amount of Tweets (including retweets)
	Total_listed	Numeric	Public lists count of user's membership
	Verified	Binary	When true, this indicates the user's verified account
	Total_friends	Numeric	The count of accounts followed by this user
	Total_favorites	Numeric	Favorites count extracted from metadata
	Total_followers	Numeric	Followers count for this account
	Default_profile	Binary	When true, it shows the user used the default theme or background on their profile
Derived features	Age_of_user	Numeric	Account age in days is calculated by subtracting the date of account creation from the date of data collection
	Friends_growth_ratio	Numeric	Listed_totaluser_age
	Followers_friends_ratio	Numeric	Followers_totalfriends_total
	Names_ratio	Numeric	Name_on_Screen to length_of_name ratio
	Length_of_screen_name	Numeric	Length of the screen name
	Name_on_screen_freq	Numeric	Average frequency of bigrams (pairs of adjacent characters) in the screen name
	Followers_growth_ratio	Numeric	Followers_total/user_age
	Tweet_freq	Numeric	Statuses_total / Age_of_user
	Length_of_name	Numeric	Name length of the user
	Length_of_description	Numeric	Length of the user profile description
	Num_screen_name_digits	Numeric	Digits in the screen name
	Num_name_digits	Numeric	Number of digits in name
	Name_on_screen_Entropy	Numeric	Entropy of user name_on_screen
	Favorites_growth_ratio	Numeric	Total_Favorites / Age_of_user
	Entropy_of_description	Numeric	Strings of description entropy
	Listed_growth_ratio	Numeric	Total_Listed / Age_of_user
	Entropy_of_name	Numeric	User name entropy
	Name_sim	Numeric	Similarity between screen name and name

$H(l)$: is the Shannon's sequence entropy which is given by the equation (3).

$$H(l) = -\sum_{i=1}^{T} P(i) * log_2 P(i) \tag{3}$$

Where in Equation (3),

−: minus sign takes care of the fact that P(i) is a fraction.
$\sum_{i=1}^{T}$: denotes the sum over the variable's possible values 1 to T.
T: is the total number.
$log_2 P(i)$: logarithm of character i^{th} probability.
$P(i)$: is the probability of obtaining the value of unique character in sequence i.

The name_sim feature represents the similarities among the name and screen_name listed in the profile of account.

$$name\_sim = \frac{2M}{N} \tag{4}$$

where in Equation (4),

N: is the element's total number.
M: is the matches number in the two sets of elements.

Table 4 shows the derived features and their corresponding classes from the name and name_on_screen of either humans or bots.

3.2.1 | Binary and Numerical feature preparation

The preprocessing carried out on the binary and numerical features as well as the selection of the feature technique applied to this feature collection, are covered in this section.

(i) Numerical feature processing

Standardization and imputation of the missing value are involved in the processing of numerical features. Due to some accounts' description fields being null, we considered missing values only and gave it a value of 0 with the feature of *length_of_description*. Moreover, *Total_followers* as skewed numerical features, the range of values for this feature extends from 0 to 108, 990, 846. Standardization is applied to all the numerical features to handle their skewness.

$$z = \frac{(x - u)}{s}, \tag{5}$$

where in Equation (5),

z: to compute the standard score.
x: for an input.
u: sample's mean.
s: standard deviation.

(ii) Binary feature processing

Discrete data for binary features can only have two distinct values. Binary features such as default and verified profile since they only accept the values True or False. These values of features are encoded in such a way that, for input l and z for the encoded value given by equation (6),

TABLE 4 Derived features from *Name_on_screen* and *Name*.

Derived feature	Bot	Human
Name	Tennessee_jobs	David Warner
Name_on_Screen	Tennessee hire	davidwarner31
Name_on_screen_freq	1	1
Name_sim	0.624	0.712
Name_Entropy	0.178	0.263
Name_on_screen_Entropy	0.183	0.237

$$z = \begin{cases} 0 & \text{if } l = False, \\ 1 & \text{if } l = True. \end{cases} \tag{6}$$

Applying the same logic in the encoding process, 1 is assigned to the "bot" class and 0 to "human" in the Label field.

(iii) Feature selection

Model performance is enhanced by selecting an optimal feature subset using the feature selection method.[42] Reducing the input vector's dimension helps minimize the model's complexity.[43] Scalability is another concern that frequently arises while employing deep learning architectures. In the real-time classification of data, the model's scalability improved with the help of feature selection. Let's assume that we will train our model offline. Then, scalability will not be an issue if we use the model only for testing.

Feature selection has two methods: filter-methods and wrapper-methods. Filter-methods are an approach of machine learning to identify the relevance of features, utilizing statistical tests of features independent of using a particular algorithm.[44] Wrapper-methods in machine learning involve using a particular algorithm to select the optimal subset of the features from which the model's performance improved.[45] There is evidence to suggest that utilizing wrapper methods can enhance the performance of a model.

For feature selection, we chose backward elimination, which is a wrapper-based method. In the backward elimination method, a specific machine learning model is trained and cross-validated iteratively.[14] All of the features are initially considered, and irrelevant features are removed at every phase to optimize the performance of the model's cross-validation. This process is repeated until the optimal number of features is reached. To decrease the risk of time complexity and over-fitting involved with the sequential method of the feature selection method. Using threefold cross-validation, we employed a Random Forest model. Finding the top 10 features among those 24 features requires starting with all of them, excluding the description feature. After feature selection, Table 5 presents the final set of features.

3.2.2 | Text feature preparation

Typically, the description box contains text that describes how the account owner behaves. There has yet to be much research done on bot detection using description text. In our TL-PBot framework, we only consider the description of the user profile, which is a text field and gives authentic information regarding account holders' behavior. Before being fed to the embedding layer, each description word is transformed into an integer through encryption in this scenario. This process is also known as text tokenization. The following is part of the description field's feature engineering flow path:

TABLE 5 Final set of features after feature selection.

Final set	Feature type
Total_statuses	Numeric
Name_on_screen_freq	Numeric
Total_friends	Numeric
Num_name_digits	Numeric
Total_listed	Numeric
Description	Text
Tweet_freq	Numeric
Total_favourites	Numeric
Entropy_of_name	Numeric
Total_followers	Numeric
Entropy_of_description	Numeric

Missing value imputation: Because the Twitter account's description field is nullable, the feature may contain missing values in the dataset. The 'missing' string is a default to replace the null values. The default string's purpose is to show the missing data. The feature of description length is still set to zero for null description values.[14,46]

Text cleaning: Emojis, emotions, and other special characters are also removed when the text is cleaned, along with URLs, hashtags, and mentions.[14,46]

Removing stop word: Stop words are vocabulary terms that are frequently used yet convey less information. These words are eliminated to minimize the vocabulary size and prioritize terms with high-level information.

Converting to lowercase: Any unwanted spaces are eliminated, and the content is converted to lowercase.

Text Tokenizer: The tokenization process in the description text involves assigning a unique integer value to each word based on its corresponding vocabulary index. A unique integer value is assigned to each word in the vocabulary to create a vocabulary index. This value is determined by how frequently the word occurs. The term with the highest frequency is given an index value of 1, and so on.[14]

3.2.3 | GloVe: Global vectors for word representation

We apply an embedding layer on the LSTM layer's top to convert the text's words into the actual numbered vectors. This is helpful for text display so that words with comparable semantic meanings are represented similarly. The layer's weights can be initialized randomly and updated during the model's training. However, when the training set is small, the model may fail to acquire the embeddings needed to comprehend the semantic relationships.

The effectiveness of the model's performance will be increased if we use the weighted vectors, which are constructed and pre-trained on a vast training dataset. The authors of Pennington et al.[47] provide GloVe word embedding, which is a pre-trained model, on their website[†]. Using unsupervised learning, the word vector representation is produced by the method of GloVe. GloVe is a model trained on a global word–word co-occurrence matrix that records the frequency of word pairs appearing together in a corpus, considering only the nonzero entries. With the objective of a log-bilinear model, GloVe is a weighted least-squares model. GloVe aims to learn word vectors that satisfy the training objective of having their dot product equivalent to the logarithm of the probability of co-occurrence of the words. They offer particular embeddings for Twitter data. 27 billion tokens from 2 billion tweets were used to train the Twitter model GloVe, which is pre-trained. There are four various dimension models available for word vectors. We used the 50-dimensional model. Hence, a vector of length 50 will represent each word.

3.3 | Proposed approach

The complete architecture of the TL-PBot framework is shown in (Figure 3). TL-PBot trained on the publicly available datasets from the bot repository. The proposed approach is trained and tested on the same datasets and further evaluated on the validation datasets not used in the training phase, which is also a novelty in our approach. To identify bots, a user's profile features undergo preprocessing and preparation through feature engineering. Our main contribution is that we used the Inductive transfer learning category multi-task learning, which used labeled datasets in source and target domains. It has been shown that transfer learning performs better in the DNN model. Transfer learning decreased the time required for training, enhanced neural network performance (under most conditions), and compensated for the lack of abundant data. In cases where training a neural network from scratch requires a large amount of data that may not always be available, transfer learning can be helpful.

Additionally, we must take note of non-text aspects like follower and friend counts, which are possible tools for bot detection. We have designed a hybrid model that consists of 2 input layers. The first input layer collects numerical features; the second is the descriptive text discovered after feature selection. We researched the description field to see how it would help the framework for detecting bots. When creating an account on Twitter, the user fills out the description area with their bio. The description field can be blank. We can see that (Figure 4a) shows the human class's description field status and found that 9.5% of the users have not filled in their description field (Figure 4b) the bot class's description field status shows that nearly half of the bots, almost 47.6%, have not filled in their description field.

A vector of 30 integers provides the tokenized description text's representation. Based on the description text's word count, this dimension was chosen. The most prolonged description in our dataset is 49 words long. But there are just two cases where the word count is above 40. Since 95% of the description contains 30 words or fewer, we have chosen 30 as

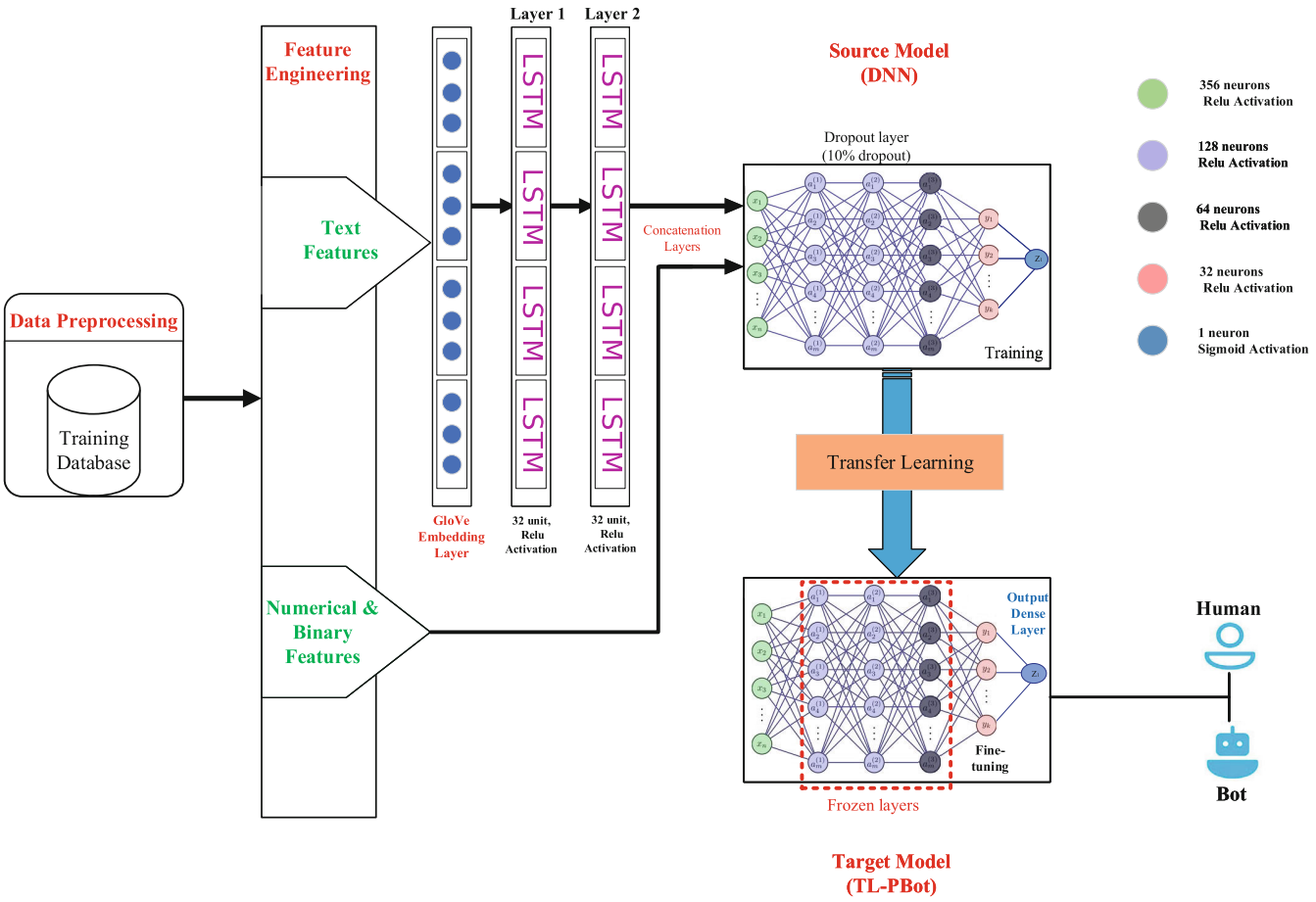

FIGURE 3 Architecture of the TL-PBot proposed framework.

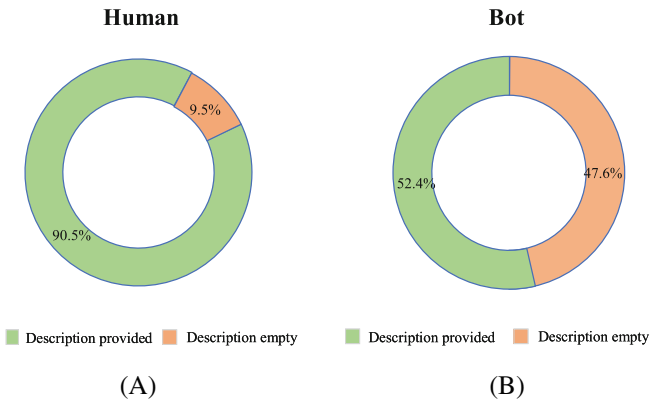

FIGURE 4 Comparison of empty description fields in human and bot profiles on the Twitter.

the integer vector size. It ensures that the necessary details are included in the description text without leading to data over-clipping. The embedding layer is then used to convert this sequence into GloVe-embedded word vectors.

Two LSTM (long short-term memory) layers then process the embedded descriptive text. A recurrent neural network (RNN) architecture with a specific type for processing time-series data is known as LSTM. Natural language processing (NLP) has been used successfully where text data is analyzed to handle various tasks like prediction and classification.[48,49] After feature selection, the dimensional numerical features obtained are concatenated with the output from the LSTM units.

To apply transfer learning, we have input layers, middle layers, and the output dense layer with their weights for prediction. The weights of the intermediate feature extraction layers were frozen, ensuring these parameters remained unaltered throughout the TL process. This judicious freezing of weights prevented updates to the middle feature extraction layers, thereby preserving the knowledge embedded in those layers.

Subsequently, we trained and fine-tuned the last layers of the model. This targeted fine-tuning process aimed to optimize the model's performance and achieve the desired output for TL-PBot. By selectively updating the parameters in the final layers, the model was adapted to the refinement of the bot detection task while leveraging the foundational knowledge encoded in the pre-trained middle layers. Thus, the output dense layer distinguished between human and bot accounts.

3.4 | TL-PBot algorithms

With the help of Algorithm 1, we summarize the TL-PBot framework's whole training phase in this section. The sub-module is processed by feature selection which is described in the algorithm. The role of each step is explained in detail in Section 3.2.2. Here, the training set's text data is used as the input. To separately manage each text of the user description, we define a for loop in step 1. As described in Section 3.2.2, step 3 replaces a default string S for the empty text description. Filtered the text in step 5 to remove all symbols, emojis, mentions, additional white space, hashtags, and stop words. In step 6, after the text has been filtered, it is converted to lowercase. Step 8 generates a Keras tokenizer class object, which step 9 fits onto the filtered text data. The text is tokenized in step 10 using the tokenizer object. To standardize the length of all sequences, the tokenized text is padded with additional tokens to reach a fixed length in step 11. This outputs the text that has been processed.

Algorithm 1. Preprocessing (P_f)

Input: P_f: The training set's text data
Output: P_{ft}: Processed text data
1: **for** each f in P_f **do**
2: **if** f = NULL **then**
3: Replace (f, S) ▷ /* Replace NULL value with a default string value S */
4: **end if**
5: $fl \leftarrow FilterText(f)$ ▷ /* Remove symbols, emojis, mentions, additional white space, hashtags, and stop words */
6: $fl \leftarrow Lowercase(fl)$ ▷ /* Convert to lowercase */
7: **end for**
8: T_k = Tokenizer () ▷ /* Create an instance of Keras tokenizer class */
9: T_k.fit_on_texts(P_{fl}) ▷ /* P_{fl} is the set of filtered text data */
10: $P_{fk} \leftarrow T_k.texts\_to\_sequences(P_{fl})$ ▷ /* Tokenize the text */
11: $P_{ft} \leftarrow Pad_sequences(P_{fk,L})$ ▷ /* Pad the sequence to max. length, L */
12: **return** P_{ft}

We explain the creation of a deep neural network (DNN) model using transfer learning in Algorithm 2. We take input layers for text input in step 1 and non-text input in step 2. Then we embedded these inputs in GloVe in step 3. Then we concatenate these layers with the LSTM layers in step 4. We freeze some dense layers and train them using transfer learning in step 5. In the end, we used the output layer for prediction in step 6. The model has been created in step,7.

The overarching TL-PBot framework is outlined in Algorithm 3. It is designed to work with a training dataset that includes labels and a mixture of binary, numerical, and text-based features as inputs. In this context: T_t represents a subset of the training dataset containing text features. T_b represents a subset of the training dataset consisting of binary features. T_n represents a subset of the training dataset comprising numerical features. L_t denotes the training dataset labels. In Step 1, feature preprocessing is conducted, as detailed in Algorithm 1. Step 2 involves encoding the data into binary format, with further information available in Section 3.2.1. Step 3 standardizes the numerical data, as described in Section 3.2.1.

Algorithm 2. Model Creation (D_n, D_t)

Input: D_n: Non-Text input dimension
D_t: Text input dimension
Output: Model

$I_n \leftarrow Input(D_n)$ ▷ /* Input layer for non-text input */
$I_t \leftarrow Input(D_t)$ ▷ /* Input layer for text input */
$E_t \leftarrow Embedding(I_t, Glove(emb_size)$ ▷ /* Input text Glove embedding */
$L_l \leftarrow LSTM\_Layers(E_t)$
$C_l \leftarrow Concatenate(L_l, I_n)$
$D_l \leftarrow Dropout\_and\_Dense\_Layers(C_l)$ ▷ /* see (Figure~3) */
$T_l \leftarrow Transfer - Learning$ ▷ /* Transfer learning applied */
$O_l \leftarrow Dense(D_l, sigmoid_activation)$ ▷ /* Output layer */
$model \leftarrow Model([I_n, I_t], O_l)$ ▷ /* Model created */
return model

Algorithm 3. TL-PBot framework

Input: T_t, T_b, T_n, L_t
Output: Trained Model

$T_{tp} \leftarrow Features\_Preprocessing(T_t)$
$T_{bp} \leftarrow Encode(T_b)$
$T_{np} \leftarrow Standardize(T_n)$
$SS_b, SS_n \leftarrow Feature\_Selection(K, T_{bp}, T_{np})$
$TS_{bp} \leftarrow Subset(T_{bp}, SS_b)$
$TS_{np} \leftarrow Subset(T_{np}, SS_n)$
$N_{ts} \leftarrow TS_{bp} \cup TS_{np}$
$N_{tt} \leftarrow T_{tp}$
$M \leftarrow Model\_Creation(dimensions.(N_{tt}, N_{ts}))$
$M.compile(Validation, Optimizer, Loss, metrics)$
$Trained\_Model \leftarrow M.fit([N_{tt}, N_{ts}], M_t)$
return model

In Step 4, K represents the total number of selected features, SS_b denotes the binary features that have been chosen, and SS_n represents the selected numerical features. Step 5 involves selecting only SS_b from the T_{bp} dataset. In Step 6, only SS_n is selected from the T_{np} dataset. Step 7 updates the selected features based on the non-text training dataset. Step 8 updates the tokenized text based on the text training dataset. In Step 9, the model created in Algorithm2 is fitted. Step 10 compiles the model. Step 11 shows the training of the model after applying transfer learning. Finally, in Step 12, a model is created at the conclusion of the process. This framework follows a step-by-step procedure to transform and train a model, leveraging transfer learning for effective performance.

4 | RESULTS AND DISCUSSION

A comparison analysis of the results of our several experiments was done. Here, we examine the experiments' results, and the TL-PBot framework emphasizes detecting Twitter bot profiles.

4.1 | Experimental setup

In our TL-PBot work, we undertook all our experiments on a 64-bit Operating system with an Intel (R) Core i5 CPU and installed RAM of 20 GB under Windows 10. Using the Anaconda platform the whole work is developed

on the open-source Anaconda platform. It is used to develop different machine learning and data science projects using different types of languages, like R or Python. It supports various libraries of Python, such as "Pandas," "Keras," "NumPy," etc., and also different Integrated Development Environments (IDEs) for the development of the code, such as Jupyter Notebook, Spyder, etc. The application was implemented using the version of Python 3.10. We implemented our module of deep learning using transfer learning by using the Keras library, and at the backend, we used TensorFlow.

4.2 | Performance measurements

To assess the performance of the classification algorithm, it's necessary to visualize the confusion matrix and compute specific performance metrics. It enables a more convenient analysis of the effectiveness of various methods and facilitates the comparison of their respective performances.

4.2.1 | Confusion matrix

The confusion matrix (Figure 5) is helpful for evaluating classification models by comparing the predicted and actual classes. It presents the number of samples in each quadrant, allowing us to identify True Positives, False Positives, True Negatives, and False Negatives. By analyzing these values, we can assess the model's performance and understand how accurately it classified the bots.

Our model made the following predictions on the datasets:

1. **True Positives (TP)**: The accounts that were accurately classified as bots.
2. **True Negatives (TN)**: The accounts that were accurately classified as human.
3. **False Positives (FP)**: The accounts that were inaccurately classified as bots when they were actually human.
4. **False Negatives (FN)**: The accounts that were inaccurately classified as human when they were actually bots.

A model trained on 23009 datasets, of which 11241 are human and 11768 are bot accounts. Performance analysis is performed on the 80% of training and the 20% of testing datasets.

4.2.2 | Evaluation metrics

In parallel with the confusion matrix, we also calculate various evaluation metrics to compare and determine the performance of different models. These metrics help us assess and identify the best-performing model among them. We can make informed decisions about which model is most effective by analyzing these metrics. To assess the effectiveness of the proposed approach, five performance metrics have been employed: Accuracy (equation 7), Precision (equation 8), Recall (equation 9), F-measure (equation 10) and Area Under Curve (AUC) (equation 11,12). These metrics are utilized to evaluate and compare the performance of the approaches and determine their efficacy.

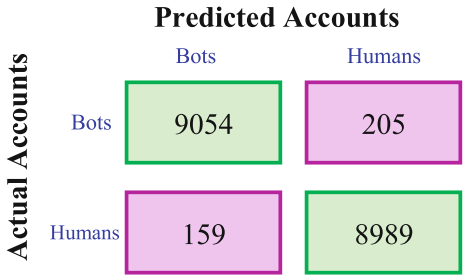

FIGURE 5 Confusion matrix of TL-PBot.

Accuracy: is the correctly classified examples percentage.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}$$

Precision: is a measure that quantifies the percentage of correctly classified predicted positives.

$$Precision = \frac{TP}{TP + FP}. \tag{8}$$

Recall: is a measure that quantifies the percentage of correctly classified actual positives.

$$Recall = \frac{TP}{TP + FN}. \tag{9}$$

F Measure: is the precision and recall combination to get a single measure.

$$FMeasure = \frac{2 * (Precision * Recall)}{Precision + Recall}. \tag{10}$$

AUC: Through receiver operator characteristic (ROC), we will also evaluate the model's performance as a metric using the Area Under Curve (AUC). A probability curve called a ROC, which plots the True Positive Rate (TPR) equation 11 versus the False Positive Rate (FPR) equation 12 at different values of threshold.

$$TPR = \frac{TP}{FN + TP}, \tag{11}$$

$$FPR = \frac{FP}{TN + FP}. \tag{12}$$

To assess the model's cross-domain performance, we evaluated it on four separate validation test datasets that were neither used for training nor testing.

4.3 | TL-PBot without transfer learning

As an initial step, we used the dataset made publicly available by the bot repository of the botometer to train and test the TL-PBot without the Transfer Learning (TL) phase. We utilized the profile features outlined in Section 3.2. We compared our model with other base articles, namely DeeProBot, Bot-MGAT, DNN-BD, and GANBot, as shown in Table 6. The selection of these models was based on their close alignment with our model's implementation and the criteria for profile feature selection. For instance, DeeProBot implemented a deep neural network that trained dense layers for prediction and classification. A simplified version of DeeProBot was tested to discern distinctions between the base and DNN-BD models. According to the findings, DNN-BD exhibited superior performance compared to DeeProBot and outperformed GANBot by a margin of 1.2%, albeit falling behind Bot-MGAT by 1.5%. Within the proposed TL-PBot framework, the model was engineered to mitigate over-fitting concerns, ensuring enhanced performance on validation datasets.

TABLE 6 TL-PBot without transfer learning (TL) compared to other baseline models.

Model	Accuracy%	Precision%	Recall%	F Measure%	AUC
DeeProBot[14]	92.00	93.00	92.00	93.00	0.97
Bot-MGAT[29]	97.80	97.00	97.00	98.00	0.98
DNN-BD[30]	96.30	96.00	96.00	96.00	0.96
GANBot[31]	95.10	94.00	92.00	96.00	0.99
$TL - PBot_{(No-TL)}$	97.85	97.95	97.42	97.53	0.98

The TL-PBot model, when implemented without transfer learning, demonstrated slightly superior performance compared to other base models, achieving an accuracy score of 97.85%, a precision score of 97.95%, and a recall score of 97.42%. However, it exhibited comparatively lower performance in F-Measures, with a score of 97.53% and an AUC of 0.98. The model $TL - PBot_{(No-TL)}$ demonstrated a significantly longer detection time, notably at 12.19 milliseconds, and incurred higher computational costs. This result suggested the potential for improvement in TL-PBot by incorporating transfer learning. Additionally, it is noteworthy that Bot-MGAT performed similarly to $TL - PBot_{(No-TL)}$ concerning AUC, achieving a score of 0.98.

4.4 | TL-PBot with transfer learning

We used the same dataset made publicly available by the bot repository of the botometer to train and test the TL-PBot with the Transfer Learning (TL) phase. The identification of bot profiles was conducted through a preprocessing and feature engineering process applied to the user profile features. A strategic approach was adopted during the Transfer Learning (TL) phase to enhance the model's performance. We trained the last layers of the model. Specifically, the weights of the intermediate feature extraction layers were frozen, ensuring these parameters remained unaltered throughout the TL process. This judicious freezing of weights prevented the updates to the middle feature extraction layers, thereby preserving the knowledge embedded in those layers. Freezing middle layer weights also affected the overall training time(as fewer parameters were being updated) effectiveness with other factors like accuracy, F measures, etc.

Subsequently, fine-tuning was executed on the last layers of the model. This targeted fine-tuning process aimed to optimize the model's performance and achieve the desired output for TL-PBot. By selectively updating the parameters in the final layers, the model was adapted to the refinement of the bot detection task while leveraging the foundational knowledge encoded in the pre-trained middle layers. This methodology allowed for the harmonious integration of domain-specific insights from the source task, ultimately contributing to the robustness and efficacy of the TL-PBot model.

To optimize model performance, we conducted an exhaustive analysis encompassing 40 trials, wherein subtle parameter adjustments were implemented, and weights were systematically updated. The iterative refinement process executed 30 epochs yielded discernible improvements in model efficacy. Specifically, the accuracy rate exhibited a marginal yet noteworthy enhancement, ascending from 98.02% to 98.07%, denoting a commendable improvement of 0.05%. Simultaneously, the F-measure rate experienced a corresponding augmentation, progressing from 98% to 98.32%, signifying a discernible advancement of 0.07%.

In Table 7, the performance evaluation of the TL-PBot model revealed notable enhancements facilitated by integrating transfer learning. Specifically, the model exhibited a discernible improvement in various vital metrics: a 0.17% increase in accuracy, a 1.05% elevation in precision, a 0.58% advancement in the recall, a 0.47% refinement in F measure, and a marginal 0.01 augmentation in the Area Under the Curve (AUC). These improvements underscored the efficacy of the transfer learning approach in refining the model's ability to accurately identify bot profiles, thereby contributing positively to its overall performance.

Implementing the transfer learning methodology had yielded a pronounced enhancement in training efficiency, manifesting as a refined temporal efficiency in the detection rate, notably at 5.04 milliseconds. In contrast, the TL-PBot deep neural network model without transfer learning was time-consuming and required training from scratch, resulting in slightly lower performance. This faster performance was caused by the system's intelligent use of weights and pre-existing knowledge stored in a trained model, which enabled it to quickly adjust and identify patterns unique to the intended detection task.

Significantly, TL-PBot reduced the need for computational costs, training time, and extra resource requirements. TL-PBot also eliminated the necessity for retraining the entire model from scratch. Collectively, these outcomes

TABLE 7 TL-PBot performance with and without transfer learning (TL)

Model	Accuracy%	Precision%	Recall%	F Measure%	AUC	Predict Time (ms)
$TL - PBot_{(No-TL)}$	97.85	97.95	97.42	97.53	0.98	12.19
$TL - PBot_{(TL)}$	98.07	99.00	98.00	98.32	0.99	5.04

underscored the superiority of the transfer learning approach over alternative methodologies, substantiating its efficacy in enhancing model performance.

4.5 | Comparison of TL-PBot framework with baselines

The model is trained using 80% of the datasets and tested on 20% of the datasets in the training set that are separate from the validation datasets. Regarding cross-domain testing, it's important to keep validation datasets in mind. Midterm-18, gilani-17, verified, botwiki, and cresci-rtbust are validation test sets. These datasets are separate from the rest of the training and testing datasets. We combined the verified dataset with the botwiki dataset for balancing between humans and bots. Each dataset is collected at various time slots and labeled using different methods and strategies, making these datasets characteristically different from each other. The TL-PBot framework was rigorously evaluated on multiple validation test datasets to assess its performance in bot account detection. Across all test datasets, the model consistently demonstrated high results.

Accuracy, precision, recall, and F-measure values were computed for each validation test set, excluding those used in the training dataset. This evaluation aimed to assess the model's performance before and after applying transfer learning. Specifically, for the verified-botwiki[35] datasets (Figure 6a), the model achieved an accuracy of 98.78%, precision of 98.59%, recall of 98.27%, and an F-measure rate of 98.69% without employing transfer learning. Subsequently, transfer learning was applied to the same dataset to observe how the TL-PBot model would perform. After implementing transfer learning, the model's accuracy increased to 99.07%, precision value improved to 98.89%, recall increased to 98.67%, and the F-measure rate improved to 99.04%. These results demonstrate the effectiveness of the TL-PBot framework after applying transfer learning to the verified-botwiki datasets, indicating its ability to enhance model performance.

In the case of the Midterm-18[35] validation test dataset (Figure 6b), before applying transfer learning, the model achieved an accuracy rate of 98.35%, precision of 98.27%, recall of 98.33% and an F-measure value of 98.43%. After the application of transfer learning, there was a noticeable performance improvement, with the accuracy rate increased to

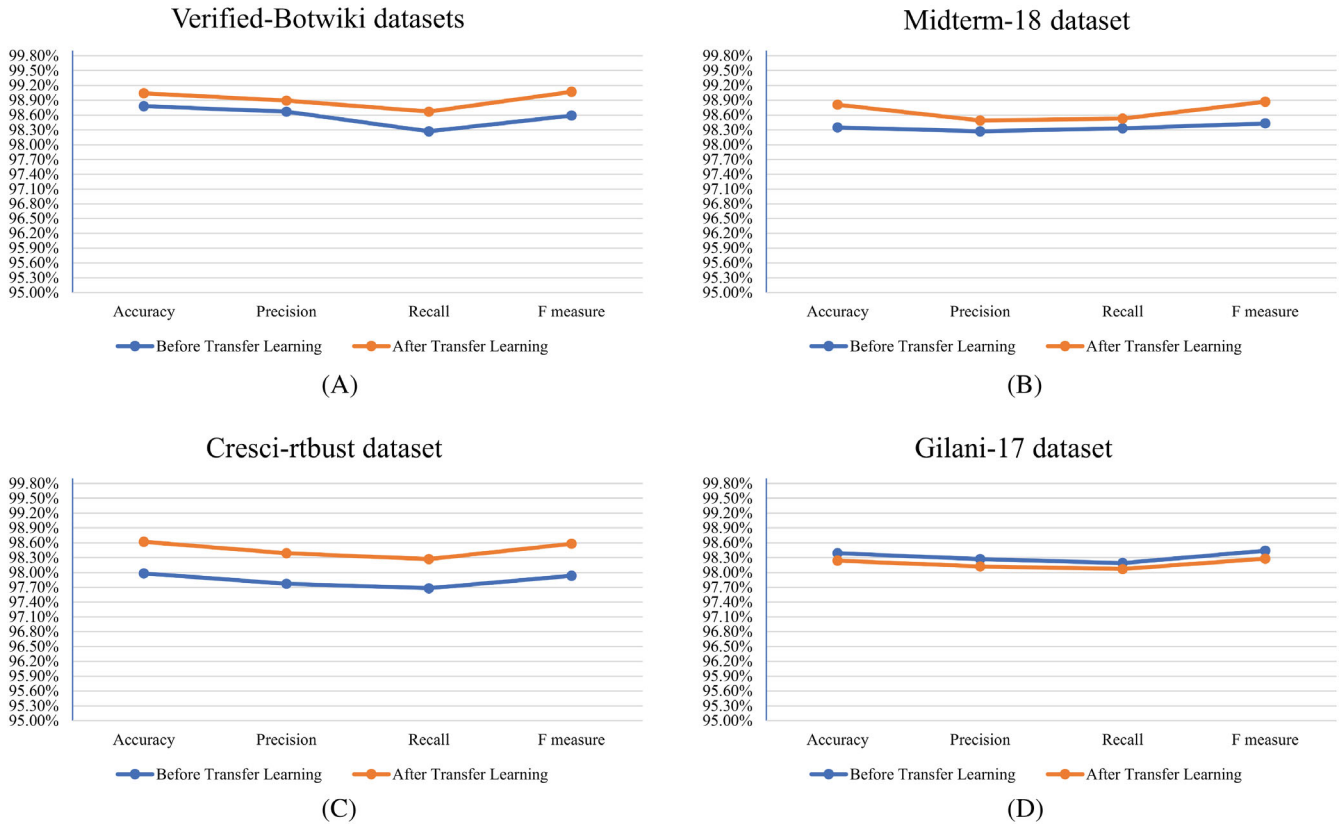

FIGURE 6 Datasets comparison before and after applying transfer learning.

98.81%, precision value improved to 98.49%, recall increased to 98.53%, and the F-measure value raised to 98.87%. These results underscore the positive impact of transfer learning on the TL-PBot model's ability to accurately detect bot accounts within the Midterm-18 dataset.

For the Cresci-rtbust[37] validation test dataset (Figure 6c), the model achieved an accuracy rate of 97.98%, precision of 97.77%, recall of 97.68% and an F-measure value of 97.93% before applying transfer learning. Subsequently, after the implementation of transfer learning, there was a notable performance improvement, with the accuracy rate increased to 98.62%, precision value improved to 98.39%, recall increased to 98.27%, and the F-measure value raised to 98.58%. These findings demonstrate the significant enhancement in the TL-PBot model's performance on the Cresci-rtbust dataset when transfer learning is employed, resulting in more accurate bot account detection.

In the case of the Gilani-17[36] validation test dataset (Figure 6d), the initial performance of the model before applying transfer learning showed an accuracy rate of 98.39%, precision of 98.27%, recall of 98.19% and an F-measure value of 98.44%. However, following the application of transfer learning, the model's accuracy rate slightly decreased to 98.24%, and the precision, recall, and F-measure value also showed a minor decrease. Despite the slight reduction in performance after transfer learning, the TL-PBot framework still exhibited high accuracy and F-measure values on the Gilani-17 dataset, confirming its robustness and effectiveness in bot account detection.

Considering the overall performance across the four distinct validation test datasets in (Figure 6), the model has demonstrated exemplary performance with the verified-botwiki, midterm-18, and Cresci-rtbust datasets. We can see that the model achieved the highest accuracy and f measures with verified-botwiki datasets after transfer learning was applied.

The lower performance observed for gilani-17 can be attributed to several factors. Firstly, the absence of certain types of bots in the training dataset impacts the model's ability to effectively identify them in the test dataset. Secondly, our chosen features need to be comprehensive enough to adequately distinguish bot accounts in this dataset. Incorporating additional high-level features could help learn the distinct characteristics of bot behavior. Lastly, manual annotation of the gilani-17 dataset is challenging for humans and introduces the possibility of errors.

Overall, the TL-PBot framework consistently outperformed in bot account detection across all datasets after applying transfer learning. Showcasing its effectiveness and robustness, as evidenced by the high accuracy and F-measure values achieved. These results validate the model's capability to accurately distinguish between human and bot accounts, making it a valuable tool in addressing the challenges posed by bot activity on social media platforms.

We conducted a comprehensive evaluation by combining all validation test datasets and comparing our model's performance with that of DeeProBot[14] and other deep learning methods. DeeProBot employed a methodology to select the most appropriate training data subset to create a highly generalizable model. Similarly, we obtained data from varol-icwsm using Twitter's API, consistent with DeeProBot's data source. We employed dense layers of Deep Neural Networks (DNN) and Long Short-Term Memory (LSTM) units for the classification task. Building on their research, we adopted the top-performing dataset combinations from DeeProBot's training dataset.

We could replicate their findings effectively by utilizing the same dataset and feature set. However, we observed that the author primarily focused on textual descriptions, and their accuracy rate for distinguishing between disgusting bot accounts and genuine accounts reached only 92%, which we considered relatively low compared to our model's accuracy rate. Due to the use of multi-task learning, the accuracy value produced in our study is high.

We compared our model and several other deep learning and transfer learning approaches in our evaluation. For instance, GANBot[31] introduced a novel framework that adapted the GAN concept, incorporating an LSTM layer as a shared channel between the generator and classifier.

Another model, DNN-BD,[30] utilizes a contextual LSTM architecture and incorporates both content and metadata to detect bots at the tweet level. Furthermore, we assessed the Bot-MGAT framework,[29] which relies on the multi-view graph attention mechanism, employing transfer learning with unlabeled and labeled datasets.

This section also presents a comprehensive graphical comparison of the results. As depicted in (Figure 7), our proposed TL-PBot model has exhibited remarkable performance, surpassing other state-of-the-art articles in bot account detection accuracy. Notably, our TL-PBot model achieved an impressive accuracy rate of 98.07%.

When comparing our model to DeeProBot, we observed a significant difference in account detection accuracy despite using the same best features and datasets. Our TL-PBot model outperformed DeeProBot, achieving an accuracy rate exceeding 92%.

Similarly, Bot-MGAT, which also employed transfer learning, obtained a competitive accuracy rate of 97.80% for bot account detection. Although slightly lower than our TL-PBot's performance, Bot-MGAT demonstrated strong results.

Furthermore, there is a substantial difference in accuracy rates between our model, DNN-BD, and GANBot. While our TL-PBot model achieved an accuracy of 96.02%, DNN-BD and GANBot lagged with accuracy rates of 96.30% and 95.10%,

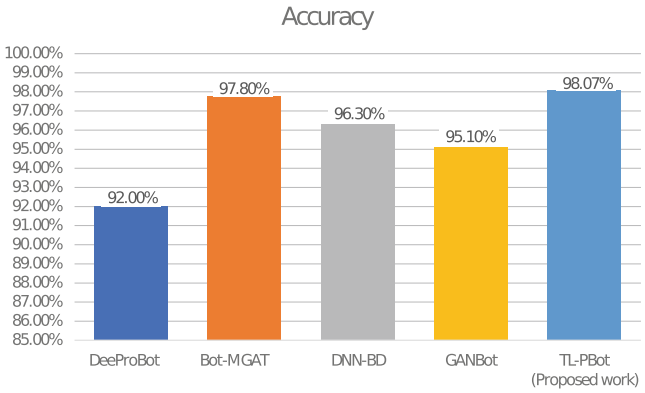

FIGURE 7 Accuracy comparison with state-of-the-art articles.

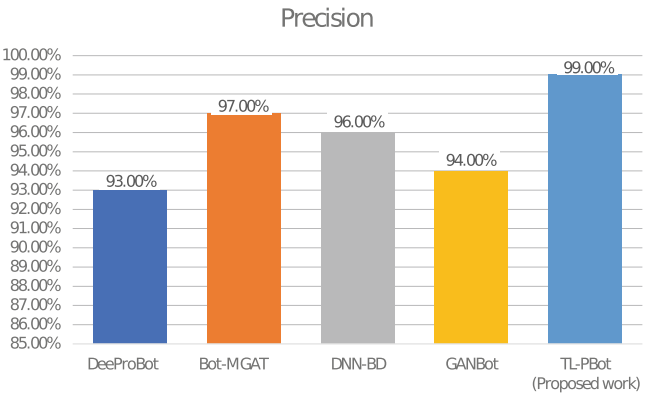

FIGURE 8 Precision comparison with state-of-the-art articles.

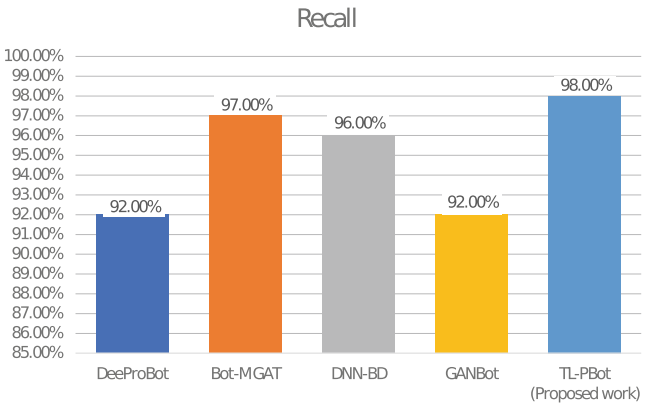

FIGURE 9 Recall comparison with state-of-the-art articles.

respectively. These findings underscore the superior performance of our TL-PBot framework in accurately detecting bot accounts compared to the other evaluated models.

Figures 8 and 9 present the Precision and Recall values for our TL-PBot proposed framework, which exhibits high performance. In comparison, DeeProBot achieved a precision value of only 93%, which is relatively low compared to the TL-PBot framework and other state-of-the-art articles. Additionally, DeeProBot attained a recall value of 92%, which is lower than the TL-PBot framework but identical to GANBot's recall performance. Moreover, the precision and recall percentages of GANBot, Bot-MGAT, and DNN-BD are also lower when compared to our TL-PBot framework.

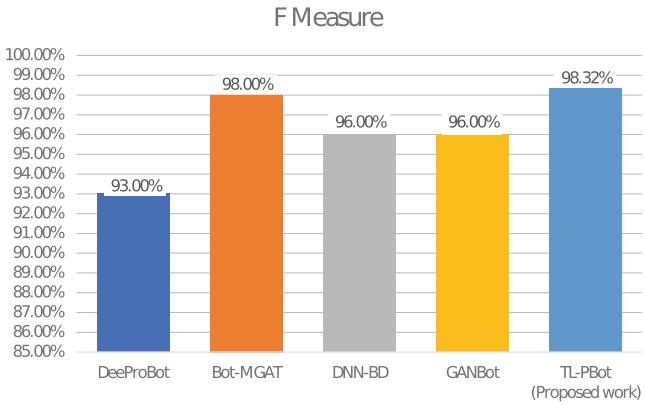

FIGURE 10 F-Measure comparison with state-of-the-art articles.

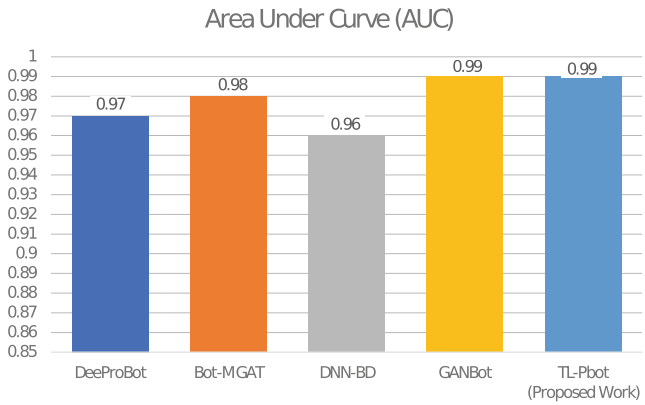

FIGURE 11 AUC comparison with state-of-the-art articles.

These results further affirm the effectiveness and superiority of the TL-PBot framework in achieving high Precision and Recall values for bot account detection.

In (Figure 10), The F-measure outcomes attained by our innovative model, TL-PBot, surpass those of comparable models, including DeeProBot, DNN-BD Bot-MGAT, and GANBot, registering an impressive F-measure score of 98.32%.

In (Figure 11), our model demonstrated an impressive Area Under Curve (AUC) value of 0.99, comparable to the results of the GANBot study. However, our AUC value surpassed those achieved by DeeProBot, Bot-MGAT, and DNN-BD, showcasing the superior performance of the TL-PBot framework. The comprehensive evaluation of the TL-PBot model demonstrates its consistent superiority over other approaches across various evaluation metrics. The results highlight the effectiveness and robustness of the TL-PBot framework in detecting bot accounts, outperforming well-known methods such as DeeProBot, GANBot, Bot-MGAT, and DNN-BD. These findings validate the TL-PBot model as a highly competitive and reliable solution for bot account detection on the Twitter social media platform.

5 | CONCLUSIONS AND FUTURE WORK

This research proposed a TL-PBot framework, which takes into account transfer learning and deep neural networks to identify bot profiles based on user profile metadata from Twitter. The novel approach incorporated inductive transfer learning, specifically in multi-task learning, to enhance the accuracy of bot account prediction. The framework encompassed various stages, starting with feature engineering and then the preparation of training and validation test datasets. Then, the model was implemented using deep neural network design, incorporating transfer learning. Notably, model performance was enhanced by training the input layers, freezing the middle feature extraction layers, and transferring their training to the dense output layer by fine-tuning. This transfer learning process was crucial in achieving high accuracy.

Moreover, the research improved the model's cross-domain performance by incorporating text from the profile description during training. A feature selection method using backward elimination was applied to identify the optimal feature subset of features to enhance the model's efficiency further. Using GloVe to embed the text feature also contributed to improving feature learning. LSTM units are leveraged to learn features from the description text effectively. A comprehensive cross-domain performance assessment of TL-PBot was conducted, involving the training and testing of the model on the same datasets. Subsequently, the model's performance was rigorously evaluated by conducting tests on four additional heterogeneous validation datasets not included in the training phase. This assessment provided insights into the model's adaptability and performance across different domains, which was also a contribution. The TL-PBot framework demonstrated impressive results, achieving an accuracy of 98.07%, precision of 99%, recall of 98%, f measure of 98.32%, and AUC of 0.99, to reinforce its effectiveness in accurately identifying bot accounts. Employing the transfer learning strategy resulted in an accelerated detection rate of 5.04 milliseconds, attesting to the efficacy of this approach in enhancing computational efficiency.

In the future work, we will test our model by reducing some more features, applying some new transfer learning techniques, like unsupervised or transductive transfer learning, and changing different parameters.

AUTHOR CONTRIBUTIONS

Maryam Bibi: conceptualization (lead); data curation (equal); formal analysis (equal); funding acquisition (equal); investigation (equal); methodology (equal); project administration (equal); resources (equal); software (equal); supervision (equal); validation (equal); visualization (lead); writing – original draft (lead); writing – review and editing (equal). **Zahid Hussain Qaisar:** conceptualization (equal); data curation (equal); formal analysis (equal); funding acquisition (equal); investigation (equal); methodology (equal); project administration (equal); resources (equal); software (equal); supervision (equal); validation (equal); visualization (equal); writing – original draft (equal); writing – review and editing (equal). **Naeem Aslam:** conceptualization (equal); data curation (equal); formal analysis (equal); funding acquisition (equal); investigation (equal); methodology (equal); project administration (equal); resources (equal); software (equal); supervision (equal); validation (equal); visualization (equal); writing – original draft (equal); writing – review and editing (equal). **Muhammad Faheem:** conceptualization (equal); data curation (equal); formal analysis (equal); investigation (equal); methodology (equal); project administration (equal); resources (equal); software (equal); supervision (equal); validation (equal); visualization (equal); writing – original draft (equal); writing – review and editing (equal). **Perveen Akhtar:** conceptualization (equal); data curation (equal); formal analysis (equal); funding acquisition (equal); investigation (equal); methodology (equal); project administration (equal); resources (equal); software (equal); supervision (equal); validation (equal); visualization (equal); writing – original draft (equal); writing – review and editing (equal).

ACKNOWLEDGMENTS

The authors are highly grateful to their affiliated universities and institutes for providing research facilities and funding for this research work.

CONFLICT OF INTEREST STATEMENT

The authors have no conflicts of interest to declare that are relevant to the content of this article.

PEER REVIEW

The peer review history for this article is available at https://publons.com/publon/10.1002/eng2.12838.

DATA AVAILABILITY STATEMENT

The datasets will be available upon request to the corresponding author.

ENDNOTES

*https://botometer.osome.iu.edu/bot-repository/.
†https://nlp.stanford.edu/projects/GloVe.

ORCID

Muhammad Faheem https://orcid.org/0000-0003-4628-4486

REFERENCES

1. Arin E, Kutlu M. Deep learning based social bot detection on Twitter. *IEEE Trans Inform Forensics Secur*. 2023;18:1763-1772.
2. Alom Z, Carminati B, Ferrari E. A deep learning model for Twitter spam detection. *Online Soc Netw Media*. 2020;18:100079.
3. Kemp S. Digital 2021: The latest insights into the 'state of digital'. https://wearesocial.com/uk/blog/2021/01/digital-2021-the-latest-insights-into-the-state-of-digital/
4. Tankovska S. Statistics shows number of monthly active twitter users. https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/
5. Balaanand M, Karthikeyan N, Karthik S, Varatharajan R, Manogaran G, Sivaparthipan C. An enhanced graph-based semi-supervised learning algorithm to detect fake users on Twitter. *J Supercomput*. 2019;75:6085-6105.
6. Shafiq M, Tian Z, Bashir AK, Du X, Guizani M. CorrAUC: a malicious Bot-IoT traffic detection method in IoT network using machine-learning techniques. *IEEE Internet Things J*. 2021;8(5):3242-3254.
7. Researchers CMU. Study by Carnegie Mellon University researchers during the Covid-19 pandemic. https://www.cs.cmu.edu/news/2020/nearly-half-twitter-accounts-discussing-reopening-america-may-be-bots
8. Stefan W, Solomon M, et al. Bots in the twittersphere. https://www.pewresearch.org/internet/2018/04/09/bots-in-the-twittersphere/
9. Chang HCH, Chen E, Zhang M, Muric G, Ferrara E. Social bots and social media manipulation in 2020: the year in review. *Handbook of Computational Social Science, Volume 1*. Vol 2021. Routledge; 2021:304-323.
10. Corbin H. Twitter bots are a major source of climate disinformation. https://www.scientificamerican.com/article/twitter-bots-are-a-major-source-of-climate-disinformation/
11. Goldenberg MJ. Stuck: How Vaccine Rumors Start—and Why They Don't Go Away, by Heidi J. Larson. New York: Oxford University Press, 2020. *J Med Human*. 2023;3:1-3.
12. Feng Y, Li J, Jiao L, Wu X. Towards learning-based, content-agnostic detection of social bot traffic. *IEEE Trans Depend Secure Comput*. 2021;18(5):2149-2163.
13. Rout RR, Lingam G, Somayajulu DVLN. Detection of malicious social bots using learning automata with URL features in Twitter network. *IEEE Trans Comput Social Syst*. 2020;7(4):1004-1018.
14. Hayawi K, Mathew S, Venugopal N, Masud MM, Ho PH. DeeProBot: a hybrid deep neural network model for social bot detection based on user profile data. *Social Netw Anal Min*. 2022;12(1):1-19.
15. Lingam G, Rout RR, Somayajulu DVLN, Ghosh SK. Particle Swarm optimization on deep reinforcement learning for detecting social spam bots and spam-influential users in twitter network. *IEEE Syst J*. 2021;15(2):2281-2292.
16. Fazil M, Sah AK, Abulaish M. DeepSBD: a deep neural network model with attention mechanism for socialbot detection. *IEEE Trans Inform Forens Secur*. 2021;16:4211-4223. doi:10.1109/TIFS.2021.3102498
17. Daouadi KE, Rebaï RZ, Amous I. Real-time bot detection from Twitter using the Twitterbot+ framework. *J Univ Comput Sci*. 2020;26(4):496-507.
18. Monica C, Nagarathna N. Detection of fake tweets using sentiment analysis. *SN Comput Sci*. 2020;1:1-7.
19. Çitlak O, Dörterler M, Doğru İA. A survey on detecting spam accounts on Twitter network. *Soc Netw Anal Min*. 2019;9:1-13.
20. Ilias L, Roussaki I. Detecting malicious activity in Twitter using deep learning techniques. *Appl Soft Comput*. 2021;107:107360.
21. Goyal B, Gill NS, Gulia P, et al. Detection of fake accounts on social media using multimodal data with deep learning. *IEEE Trans Comput Social Syst*. 2023;8:1-12. doi:10.1109/TCSS.2023.3296837
22. Yang KC, Varol O, Davis CA, Ferrara E, Flammini A, Menczer F. Arming the public with artificial intelligence to counter social bots. *Human Behav Emerg Technol*. 2019;1(1):48-61.
23. Gianvecchio S, Xie M, Wu Z, Wang H. Humans and bots in internet chat: measurement, analysis, and automated classification. *IEEE Trans Netw*. 2011;19(5):1557-1571.
24. Khaund T, Kirdemir B, Agarwal N, Liu H, Morstatter F. Social bots and their coordination during online campaigns: a survey. *IEEE Trans Comput Social Syst*. 2022;9(2):530-545.
25. Pratama PG, Rakhmawati NA. Social bot detection on 2019 Indonesia president candidate's supporter's tweets. *Proc Comput Sci*. 2019;161:813-820.
26. Orabi M, Mouheb D, Al Aghbari Z, Kamel I. Detection of bots in social media: a systematic review. *Inform Process Manage*. 2020;57(4):102250.
27. Pastor-Galindo J, Marmol FG, Pérez GM. Profiling users and bots in Twitter through social media analysis. *Inform Sci*. 2022;613:161-183.
28. Priyadarshini I, Sahu S, Kumar R. A transfer learning approach for detecting offensive and hate speech on social media platforms. *Multimed Tools Appl*. 2023;1-27:27473-27499.
29. Alothali E, Salih M, Hayawi K, Alashwal H. Bot-MGAT: a transfer learning model based on a multi-view graph attention network to detect social bots. *Appl Sci*. 2022;12(16):8117.
30. Kudugunta S, Ferrara E. Deep neural networks for bot detection. *Inform Sci*. 2018;467:312-322.
31. Najari S, Salehi M, Farahbakhsh R. GANBOT: a GAN-based framework for social bot detection. *Soc Netw Anal Min*. 2022;12(1):1-11.
32. Varol O, Ferrara E, Davis C, Menczer F, Flammini A. Online human-bot interactions: Detection, estimation, and characterization. *Proceedings of the International AAAI Conference on Web and Social Media*. Vol 11. Indiana University; 2017:280-289.
33. Cresci S, Di Pietro R, Petrocchi M, Spognardi A, Tesconi M. Social fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling. *IEEE Trans Depend Secure Comput*. 2017;15(4):561-576.
34. Cresci S, Di Pietro R, Petrocchi M, Spognardi A, Tesconi M. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. *Proceedings of the 26th International Conference on World Wide Web Companion*. ACM Digital Library; 2017:963-972.

35. Yang KC, Varol O, Hui PM, Menczer F. Scalable and generalizable social bot detection through data selection. *Proc AAAI Confer Artif Intell*. 2020;34(1):1096-1103. doi:10.1609/aaai.v34i01.5460
36. Gilani Z, Farahbakhsh R, Tyson G, Wang L, Crowcroft J. Of Bots and Humans (on Twitter). *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017 (ASONAM'17), New York, USA*. 2017;349-354.
37. Mazza M, Cresci S, Avvenuti M, Quattrociocchi W, Tesconi M. Rtbust: Exploiting temporal patterns for botnet detection on twitter. *Proceedings of the 10th ACM Conference on Web Science*. ACM Digital Library; 2019:183-192.
38. Alothali E, Hayawi K, Alashwal H. Hybrid feature selection approach to identify optimal features of profile metadata to detect social bots in Twitter. *Soc Netw Anal Min*. 2021;11:1-15.
39. Ouni S, Fkih F, Omri MN. Toward a new approach to author profiling based on the extraction of statistical features. *Soc Netw Anal Min*. 2021;11(1):59.
40. Inuwa-Dutse I, Liptrott M, Korkontzelos I. Detection of spam-posting accounts on Twitter. *Neurocomputing*. 2018;315:496-511.
41. Zahra AA, Widyawan W, Fauziati S. Development of bot detection applications on twitter social media using machine learning with a random forest classifier algorithm. *Int J Inform Technol Electr Eng*. 2020;4(2):66-73.
42. Moghaddam SH, Abbaspour M. Friendship preference: scalable and robust category of features for social bot detection. *IEEE Trans Depend Secure Comput*. 2023;20(2):1516-1528.
43. Khalid S, Khalil T, Nasreen S. A survey of feature selection and feature extraction techniques in machine learning. *2014 Science and Information Conference*. IEEE; 2014:372-378.
44. Dash M, Liu H. Consistency-based search in feature selection. *Artif Intell*. 2003;151(1/2):155-176.
45. Kohavi R, John GH. Wrappers for feature subset selection. *Artif Intell*. 1997;97(1/2):273-324.
46. Srijith S. Efficient Tweet preprocessing. https://www.kaggle.com/code/sreejiths0/efficient-tweet-preprocessing/
47. Pennington J, Socher R, Manning CD. Glove: global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL Anthology; 2014:1532-1543.
48. Yilmaz S, Toklu S. A deep learning analysis on question classification task using Word2vec representations. *Neural Comput Appl*. 2020;32(7):2909-2928.
49. Wang J, Peng B, Zhang X. Using a stacked residual LSTM model for sentiment intensity prediction. *Neurocomputing*. 2018;322:93-101.

How to cite this article: Bibi M, Hussain Qaisar Z, Aslam N, Faheem M, Akhtar P. TL-PBot: Twitter bot profile detection using transfer learning based on DNN model. *Engineering Reports*. 2024;6(9):e12838. doi: 10.1002/eng2.12838