Social Network Analysis and Mining (2022) 12:158
https://doi.org/10.1007/s13278-022-00983-9

ORIGINAL ARTICLE

RunMax: fake profile classification using novel nonlinear activation in CNN

Putra Wanda[1]

Received: 3 April 2022 / Revised: 4 October 2022 / Accepted: 6 October 2022 / Published online: 27 October 2022

Abstract

Online social networks (OSN) are well-known platforms for exchanging various information. However, one of the existing OSN challenges is the issue of fake accounts. The attacker harnesses malicious accounts in the infected system to spread false information, such as malware, viruses, and harmful URLs. Based on the vast triumphs of deep learning in several fields, mainly automated representation, we propose RunFake, a convolutional neural network (CNN) to handle malicious account classification. We build a dynamic CNN to train a classification model instead of using regular machine learning. In particular, we create a general activation function called RunMax as a new element of the neural network's final layer. We improve accuracy in the training and testing procedure by utilizing the proposed activation layer instead of the traditional function. Based on the experimental result, our method can yield Precision = 94.00, Recall = 93.21, and F1-Score = 93.42 with a better area under curve (AUC) score = 0.9547 using user profile data as features. We harvest a promising outcome with greater accuracy with tiny loss than common learning architecture in a fake account classification problem.

Keywords Classification · Fake profile · Nonlinear activation · Online social network

1 Introduction

Millions of user accounts connect and share different data items, including movies, images, and messages, on public OSNs. On the other hand, fake accounts represent a significant risk in OSN protection. Counterfeit profiles are used in various current threats and other criminal actions. Approaches to detecting false profiles in the OSN may be divided into data analysis and individual accounts. The development of phony profiles is usually regarded as causing more damage than any other cybercrime. As a result, this danger should be recognized before the user is told that a bogus profile has been created. Anomaly detection is proposed in a variety of ways in many research. Another research proposed a fake account detection methodology based on the similarity of the user's friends. The graph's adjacency matrix is calculated using this method. It examines the friend's network structure to determine if the user is genuine or not (Muftic et al. 2016).

The traditional authentication methods to construct the protection scheme are passwords and PINs. Unfortunately, the conventional technique risks user data since the attacker unlocks or authenticates into the system. There is no ongoing security to monitor the user's activities and patterns. Moreover, unauthorized individuals may be able to crack simple passwords or PINs on mobile phones or even wearable devices because of these issues (Takahashi et al. 2018).

The OSN environment keeps a massive amount of private data. However, it has attracted attackers to steal sensitive data, share fake news, and spread malicious applications. Phishing is one of the most common methods of attack for suspicious reasons. The hacker can use the fraudulent practice of obtaining sensitive information such as PINs, passwords, and credit card details. Fake profiles play an important role in other malicious activities on the OSN. In this part of the thesis, we develop learning techniques to classify benign or fake accounts relying on user features with supervised learning (Takahashi et al. 2018; Abeer et al. 2016).

Furthermore, the flaws of these point-of-entry methods are frequently explored. It continues to be a flaw in the user's ability to regulate and monitor risk. The OSN security approach requires critical characteristics such as privacy, authenticity, integrity, and non-repudiation. Existing users utilize the

✉ Putra Wanda
wpwawan@gmail.com

[1] Universitas Respati Yogyakarta, Yogyakarta, Indonesia

OSNs ecosystem to communicate a variety of data elements. It is simple for attackers to gather critical data and damage the system (Abeer et al. 2016). A vast number of malicious accounts may cause the theft of private data because the first step to jeopardizing user privacy is data leaking (Kökciyan and Yolum 2016).

A compromised OSN system displays unusual activity because the infected user may disseminate false or misleading information to the target user. By using a fraudulent account, Sybil can attack the large OSN ecosystem. Both human and bot accounts have profiles with comparable features and attributes. For example, individual reports and bot accounts have a similar identification, such as a name or images. The strange intruder might use the disadvantage to watch the target. Telegram, one of the most popular social messengers, has a vulnerability with the encrypted messenger. It could be possible to reconstitute the data log that the user sends or receives (Anglano et al. 2017). Suspicious nodes may cause Sybil threats due to OSN's shortcoming protection mechanisms (Al-Qurishi et al. 2017).

Fake account concerns include a variety of risks, such as vertical attacks on OSN providers and horizontal attacks on OSN members. According to the server, fake accounts may mimic regular users by using a phony name and posting false information on their profiles. One of the most challenging problems is detecting fraudulent profiles in a large volume of unstructured data. It must categorize the characteristics of users in-depth to detect irregularities. The aspects of information for security analysis are frequently provided via the public OSN (Bindu and Thilagam 2016).

In the typical OSN graph, the OSN contains many actors (nodes) and connections (links). Actors include individuals, groups, organizations, computers, and other information knowledge processing entities. User profiles include many personal facts, such as usernames, complete names, and phone numbers. Because of the flaws in privacy protection, a competent attacker may modify or corrupt crucial information. They might use social engineering tactics and malicious accounts to steal information and modify data. Fake profile attacks threaten an organization's reputation because of odd patterns and unneeded updates (Meier and Johnson 2022).

According to the literature analysis, most existing public OSNs need fast and reliable privacy-preserving detection to discover anomalous accounts in an OSN dynamic context. To solve the problem of fake accounts in the dynamic environment, the communities should propose a novel approach to deal with intelligence and sophisticated attacks. Many studies suggest traditional methods for detecting malicious code, such as statistics, clustering, and machine learning. However, the techniques need frequent access to IDs or account monitoring to distinguish between regular and suspect users. These flaws still exist in conventional methodologies. For starters, existing approaches are incapable of detecting fake accounts quickly in the large and dynamic ecosystem. OSN has many users that may register tens of thousands of new users daily. Unfortunately, the attackers attempt to create fake accounts on a large scale (Vigliotti and Hankin 2015).

As a result, the OSN system must develop a pre-entry level detection model. Before joining the OSN system, it must develop a rapid and scalable mechanism for detecting and responding to fake accounts. Existing statistical analysis approaches have difficulty recognizing false account connections in dynamic user interaction. The strategy allows malicious accounts to remain in the network, make malicious connections, and collect extensive activity data (Vigliotti and Hankin 2015). In this study, we experimented with constructing a model with a new classifier function for classifying OSN fraudulent accounts. To conduct our experiment, we collect various user attributes from user profile information as samples to help our training model.

Instead of utilizing conventional learning techniques like linear or logistic regression and SVM, we construct a unique approach to identify false profile accounts effectively. To categorize the anomalous account in OSN, we use a supervised learning model. It's a technique for detecting rogue accounts using OSN user characteristics. The best of our survey and expertise is the one strategy for tackling the rogue profile issue by evaluating user profile information with advanced learning. The goal of the model is to create and choose subsets of relevant characteristics for building a viable predictor. The following is a summary of our primary contributions, which are focused on addressing fake profile categorization in the OSN system:

1. We present a novel method for identifying false profiles in the OSN using a learning technique. Its goal is to detect fake profiles before they remain and infect the system. The model honed the OSN accounts' engineered characteristics. To improve the learning classifier's accuracy and performance, we build a CNN model with a generic function of the final layer. Instead of employing a traditional CNN architecture, we used a generic linear classifier to train the suggested CNN model using a fictitious profile dataset.
2. We train the proposed classifier in the CNN architecture to measure whether it can provide cutting-edge results, particularly when identifying fake OSN profiles. Then, we give a metric assessment to check the model's quality. With the benchmark results, the suggested approach may increase classification performance. The model utilizes matrix inputs to calculate the large OSN features and train the features using a variety of hyperparameters.
3. We test the proposed activation function in a dynamic CNN architecture with different pooling functions using various valuable features. In CNN architecture, it is a new function for the final layer of the neural network. The

function may produce the best result by setting appropriate hyper-parameters and computing NN parameters. Furthermore, our experiment shows that the proposed approach can be a novel solution to classify fake profiles and a potential solution for improving the NN in classification tasks.

Organization: In Chapter I, the paper will address the topic and the paper's contribution, and in Chapter II, it will review similar studies. The experiment's background is described in Chapter III, the suggested model is described in Chapter IV, the experimental setup is discussed in Chapter V, and the results and analysis are discussed in Chapter VI. In this section, we also give a metric assessment to evaluate the model's efficacy. The conclusion and future study directions are discussed in Chapter VII.

2 Related works

OSNs provide a platform for users to interact with others by expressing their opinions, resharing content into different networks. Millions of user profiles and billions of transaction data make up a huge OSN. People's popularity and social evaluations may both benefit from increased OSN. OSN users, for example, may gain popularity by accumulating a large number of likes, follows, and comments. However, it is simple to establish phony accounts, or anyone may purchase them for a low price online. Typically, approaches for detecting abnormal accounts in OSN evaluate activity fluctuations. With an increase in the number of active users on OSN, the propagation of fake news became obvious (Uppada et al. 2022).

In most cases, the users' actions change over time. The server might detect the suspicious account due to a quick change in access patterns for information and behavior. If it fails, the anomaly can potentially infect the system with already existing fraud (Vigliotti and Hankin 2015). A Cyborg, a form of phony account with falsified identities, is also to blame for the infected account. It undermines the user's reputation by using the account to disseminate false information, create falsehoods, and polarize public opinion (Durst and Zhu 2016). On the other hand, diverse communities use various dataset analysis techniques to tackle the problem, including supervised and unsupervised learning. The model may train the features to calculate user categorization throughout the learning phase (Vigliotti and Hankin 2015).

OSN security and privacy is a growing research area in the current year (Wanda and Jie 2020, 2021; Wanda et al. 2020; Jie and Wanda 2020). The research of fake account identification may use dynamic data such as behavioral analysis, graph theory, learning algorithm, application design, and OSN characteristics data (Nadav 2021; Satish Kumar 2022; Kevin et al. 2021). Various papers design multiple ways to find and categorize abnormalities using the characteristics. Another research investigated how to thwart the intruder's suspicious actions in a big OSN by creating a community detection algorithm (Liu et al. 2014). Another study proposes a model with a social behavior foundation. To tackle the detection issue, it looks into the profiles of people. The model can categorize the compromised user by evaluating their behavior in a single OSN environment. The approaches use intelligent sensing models to identify abnormalities (Sharma et al. 2017) or assess suspect accounts at various grades without considering horizontal categorization (Ruan et al. 2016).

The traditional scheme of OSN security utilized CAPTCHA to authenticate users and acquire an effective authentication procedure for different concerns. To gain adequate authentication, the OSN environment must devise a system to address the issue of fraudulent accounts. The approach may be considered the physical–social location (Ni et al. 2016), rumors propagation (Tan et al. 2016), or even monitoring user activity in joint community OSN for roaming service with user anonymity (Baingana and Giannakis 2016). On the other hand, standard security techniques make discovering and blocking phony accounts challenging. Conventional approaches, including CAPTCHAs and SMS verification, are used in traditional security to authenticate accounts and prevent false accounts (Simon et al. 2021).

The current strategies present the technique to avoid assaults in the pre-stage rather than identifying phony accounts after they have penetrated the network. Some research suggested rewiring and augmenting social graphs (Mohaisen and Hollenbeck 2013) or constructing a design point between the network connection and threat resilience-based Sybil defensive models (Chiluka et al. 2015). Another method for identifying hacked accounts is to create user habits using the behavioral profile. The technique divides comparable material and hacked accounts using content similarity (Egele et al. 2017). Other current approaches are using the SENAD model incorporated as authenticity score and factors in user social engagement-centric measures (Uppada et al. 2022), utilizing SVM, ANN, and an RF technique to build a fake profile detection model (Prabhu Kavin et al. 2022) and other machine learning models (Al-Zoubi et al. 2021). However, since the OSN needs to maximize development and engagement, it is difficult to apply in the practice area unless misuse causes a severe problem inside the system.

In this study, we design a new classifier using the learning approach to recognize the identification categories of the OSN account based on positive findings. According to our analysis, none of the available solutions present a classification strategy for detecting abnormalities in OSN using a unique supervised learning model that analyzes high data characteristics. As a result, to solve the issue of rogue profiles, we develop a new classifier in the neural network to enhance the mode that not

only detects OSN groups but also predicts future malicious activities pattern.

3 Proposed method

A common technique for building the OSN security model is machine learning. However, manual feature engineering is a costly and laborious task. Instead of using conventional ML, we proposed a deep learning algorithm to establish the learning model in OSN. Typically, the researcher can construct a classifier model that is appropriate to the problem of employing the DL algorithm as a solution for OSN protection and business purposes. To improve training and testing accuracy, a study can take advantage of several parameters in the DL algorithm, including epoch, regular, and optimizer.

In this paper, we construct a generic activation function called RunMax as a new part of the final layer by adding the Gaussian factor as an element of the function. Instead of training the model using a standard activation function, we introduce the technique as a linear classifier combined with the distribution to optimize the standard SoftMax classifier. We use gradient descent techniques to signify the stride size and attain a (local) minimum in the training phase. Based on the demonstration, we obtain higher accuracy in the training and testing result than the standard activation function. Notably, choosing a suitable Gaussian factor becomes essential for determining the NN performance in the fake accounts' classification problem.

In the first part, we construct a CNN architecture with several hidden layers and a hyperparameter. Instead of using the conventional architecture, we introduce a novel CNN architecture to deal with malicious activities in OSN. This technique adopts supervised learning to detect fake accounts by analyzing various useful features. Figure 1 depicts the CNN topology to train the classifier using engineered features.

The proposed CNN employs several hidden layers to train and test the model and utilizes a gradient descent to minimize the objective function with the model's parameters. The model updates the settings in the opposite direction of the gradient of the objective function. By using the proposed CNN topology, we calculate the accuracy and loss of the training and testing process to achieve the best result with the diverse input vector. This study feeds the 1D features dataset and tunes an appropriate hyperparameter is beneficial. We establish a supervised learning model by defining calculation over NN as follows:

Input features $x^{(i)} \in R$

Outputs $x^{(i)} \in Y(e.g.R, \{0,1\}, \{1,...,p\})$

Model parameters $\theta \in \mathbb{R}^k$

Hypothesis function $h_{\theta:} : \mathbb{R}^n \to \mathbb{R}$

Loss function $\ell : \mathbb{R} \times Y \to \mathbb{R}_{+:}$

In this study, we calculate the optimization problem as follows:

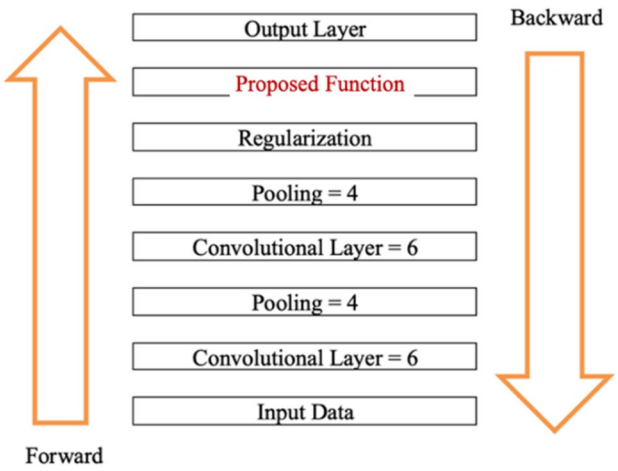


Fig. 1 Proposed CNN topology for training the OSN dataset

$$\underset{\theta}{\text{Minimize}} \sum_{i=1}^{m} \ell\left(h_{\theta:}\left(x^{(i)}\right), y^{(i)}\right) \tag{1}$$

In this paper, we provide hypothesis function $h_{\theta:} : \mathbb{R}^n \to \mathbb{R}$ in neural network processing. On a CNN, we need to calculate forward pass and backward pass to measure the gradient of the loss function in the model. The study calculates the forward pass to convolve input matrix x_i with filter W_i to produce convolution output $z_{i:}$ as follows:

$$f : \mathbb{R}^n \to \mathbb{R}^m \tag{2}$$

$$z_{i:}\left(x_i\right) = W_i x_i + b \tag{3}$$

The CNN consists of the filters W_i and bias term b as the parameters of the convolutional layer during training. CNN has many identical neurons among the layers to run large models' computation with a little number of parameters. The layer receives a single input (the feature maps) and computes the feature maps as its output by convolving filters across the feature maps. The parameters of the convolution layer called filters and the back-propagation model are used to learn during training.

In the second part, we construct a generic linear activation function as a linear classifier called RunMax combined with the CNN model to increase the model performance. This method establishes a classifier with a binary classification type, which may be the most widely applied kind of deep learning case. We build supervised learning to train the model by taking input data and crossing it onto hidden layers. To produce the output, the network undergoes computation in convolution, pooling, and fully connected layers. In this experiment, we classify the user profile as usual or malicious by calculating the explicit user features.

3.1 Nonlinear activation function

The CNN network has an output of positive numbers and a sum of one. In the final phase, the conventional model uses a SoftMax function to reduce noise signals in the fully connected layer before producing the classification result. To calculate probabilities, the classification layer utilizes the output result. In the final layer of a CNN, a model can employ SoftMax to reduce noise signals in the fully connected layer before producing the classification result. The SoftMax layer takes the logits as the input and has the probability values as the output layer.

After the last fully connected layer, a typical architecture utilizes the SoftMax function as a linear classifier. The network acts as a classifier for a problem with classes $c_1 \ldots ..c_n$, the output layer contains one neuron per class and building a vector $z = z_1 \ldots ..z_N$. The SoftMax will be used to convert the values into probabilities, where SoftMax $\sigma(z)_t$ is the probability of the input to belong to the class and e is the exponential function. The SoftMax can normalize the output of the fully connected layer. In the neural network, the SoftMax layer can be calculated with formula 4.

$$\sigma(z)_l = \frac{\exp^{z_l}}{\sum_{n=1}^{N} \exp^{z_n}}, \quad t = 1 \ldots N \tag{4}$$

SoftMax has two parts: a particular number e exponents to some power divided by a sum of some sort. Notation z_i refers to each element with the logits vector z. In deep learning, especially in SoftMax, the term logits are popularly used for the last neuron layer of a neural network for a classification task that harvests raw prediction values. Logits are the raw scores produced by the last layer of a neural network before activation. As a result, SoftMax produces numbers representing probabilities values between 0 and 1 valid value range of probabilities (denoted as [0,1]). Figure 2 depicts the SoftMax operation from input to multiclass output.

To train the SoftMax-based models, the function utilizes a gradient descent concept. The gradient of SoftMax is calculated by:

$$\frac{\partial \left[\log \sigma(z)\right]_t}{\partial_{z_n}} = \left\{ \begin{array}{ll} 1 - [\sigma(z)]n & \text{if } n = t \\ -[\sigma(z)]n & \text{if } n \neq t \end{array} \right\} \tag{5}$$

We can calculate the derivative of SoftMax where $\hat{y}_t$ is the output prediction of the target class t,

$$\hat{y}_t = \frac{e^{z_l}}{\sum_{n'=1}^{N} e^{z_{n'}}} \tag{6}$$

To train a neural network, we require to compute the gradient updates of each layer concerning its inputs. So, we need to calculate the partial derivative of the SoftMax function with respect to a single input z_n using the above rule. Thus, we can compute the gradient of SoftMax with respect to z_n for $n = t$ as follows:

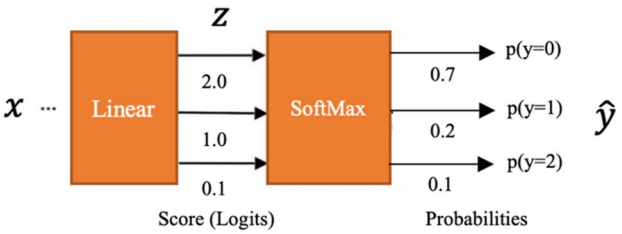


Fig. 2 SoftMax operation from input to multiclass output in the final layer of NN

$$\begin{aligned} \frac{\partial(\hat{y}_t)}{\partial z_n} &= \frac{\partial\left(\frac{e^{z_t}}{\sum_{n'=1}^{N} e^{z_{n'}}}\right)}{\partial z_n} \\ &= \frac{\left(\frac{\partial}{\partial_{z_n}} e^{z_t}\right) \sum_{n'=1}^{N} e^{z_{n'} - e^{z_t}} \left(\frac{\partial}{\partial^{z_n}} \sum_{n'=1}^{N} e^{z_{,'}}\right)}{\left(\sum_{n'=1}^{N} e^{z_{n'}}\right)^2} \\ &= \frac{e^{z_t} \sum_{n'=1}^{N} e^{z'_n} - e^{z_t} e^{z_n}}{\left(\sum_{n'=1}^{N} e^{z'_n}\right)^2} \\ &= \frac{\hat{y} e^{z_t}}{\sum_{n'=1}^{N} e^{z'_n}} \left(\frac{\sum_{n'=1}^{N} e^{z'_n} - e^{z_n}}{\sum_{n'=1}^{N} e^{z'_n}}\right) \\ \hat{y}_t &= \frac{e^{z_t}}{\sum_{n'=1}^{N} e^{z'_n}} \end{aligned} \tag{7}$$

And for $n \neq t$ as follows,

$$\begin{aligned} \frac{\partial(\hat{y}_t)}{\partial z_n} &= \frac{0 - e^{z_t} e^{z_n}}{\left(\sum_{n'=1}^{N} e^{z'_n}\right)^2} \\ \frac{\partial(\hat{y}_t)}{\partial z_n} &= \hat{y}_t \hat{y}_n \end{aligned} \tag{8}$$

3.2 Proposed nonlinear activation: RunMax

In this experiment, we construct a fully connected (FC) layer as the final layer in a the CNN architecture. In the layer, every neuron in the preceding layer is connected to every neuron and depend on the level of feature abstraction. This layer gets the convolutional, ReLU, or pooling layer as its input and calculates the accuracy and loss score. The FC layer computes that outputs a vector of K (the number of classes) dimensions in the classification process, and the vector owns the probabilities for each category.

Instead of using traditional approach like SoftMax which suffers from a bottleneck in a language modeling problem when training in extensive features, SoftMax bottleneck has been described in detail in the previous study (Yang et al. 2018). The bottleneck is caused by a rank deficiency in the final linear layer due to its connection with matrix factorization. To deal with the SoftMax bottleneck from the output set (range) and construct a function without additional parameters, we propose RunMax as an alternative activation function to improve the traditional SoftMax. RunMax contains desirable properties for output activation functions by composing an exponential function and Gaussian function for its gradient calculation. In this paper, we introduce RunMax nonlinear activation as an alternative solution to deal with the SoftMax bottleneck in the large 1D dataset. The function is used at the final layer of CNN for the classification tasks. We develop RunMax linear classifier $f(z)_i$ as follows:

$$\left[f(z)_i\right] = \frac{\exp\left(z_i\right)\varphi\left(z_i\right)}{\sum_{m=1}^{M}\exp\left(z_m\right)\varphi\left(z_m\right)}, \quad i = 1 \ldots M \tag{9}$$

Notably, instead of using a standard activation function that calculates logits' probability values, we propose a novel linear classifier with $\varphi\left(z_i\right)$ and $\varphi\left(z_m\right)$ as a Gaussian function (φ) for the z_i and z_m samples to construct the fully connected layer. Using the OSN dataset for experiments, we conduct training models using RunMax and provide the CNN performance.

4 Experimental setup

4.1 Dataset

A user profile records a person's activity in the OSN environment over time. Its purpose is to capture the typical behavior of a frequent user. This research focuses on accumulating the stream of information posted on the user wall to create user profiles. Then we gather additional essential information such as profile images and social activities (e.g., connecting friend or follower relationships). Unfortunately, the OSN system seldom provides historical data on feature changes, and as a result. We give full OSN profile data as the dataset at this time, which includes a sample with a benign and bogus label. Identity, complete names, number of friends, number of followers, time, language, a link, message samples, and profile photographs are all included in the sample.

To create a benchmark dataset, we collect and extract several profile characteristics from the corpus. By deconstructing certain user profile information, we gathered sample OSN profiles with over 3000 unique users as the dataset for this research. The study looks at user data from over three thousand OSN members. To obtain a promising accuracy and tiny loss in the training and testing process, we separate the dataset into training and testing datasets. To achieve a better performance of the proposed learning model, we developed a learning method to train and evaluate the dataset using various features. Table 1 shows the training and testing dataset of this study.

Table 1 OSN dataset as features in training and testing

Dataset	Training	Testing
Normal	1200	300
Fake	1230	320

Table 2 OSN profile's features

Features	Description
Username_identity	The account holder's identification number
Screen_name	Account's full name
Friend_count	The account's total number of friends
Follower_count	The account's total amount of followers
Time_interaction	The account holder's time information
Language	The account holder's language
Location	The account holder's location
Profile_background	The profile background information in the OSN account message
Description_account	The account holder's brief description
Profile_image	The account holder's profile picture

We give the OSN dataset to train the learning classifier, containing numerous user attributes such as identity P_u and relation R_u. User identity is an individual account in the OSN that relates to a genuine natural member. P_u is a user representation that includes usernames, ages, and other information. In the OSN, R_u symbolizes the user's social relationships. R_u network includes how many friends they have. The OSN connection is defined as and is represented as $\mathbb{C}(\cup, \varepsilon)$ where $\cup = \{p_1, p_2, \ldots p_n\}$ is the set of user profiles and the collection of OSN links is stated as $\varepsilon \subseteq \cup \times \cup$. Table 2 lists the high-profile data used to train the classifier.

This research includes a dataset with many profiles, both real and fraudulent. In the OSN environment, we give certain fundamental characteristics that depict dynamic interaction. To increase the dataset reliability, we separate the sample data by splitting the train and test datasets. Splitting the sample data into two parts can improve the classifier's quality of training results on a benchmark dataset. By dividing the dataset, the classifier can ensure the model performance accuracy on the unseen data. We undergo the preprocessing stage to enable the learning model to learn more efficiently.

The research also includes assessment measures for determining how effectively a neural network generalizes to non-dataset situations. In the first step, we prepare the user profile data as input and utilize it to create the extracted characteristics. The model analyzes the data, extracts a few features, and

then uses the learning process to train the features. Finally, we compute the extracted features by examining existing attributes to identify whether a user profile is fraudulent or benign.

4.2 Preprocessing

The experiment obtains the user information as features containing various user profiles information. To build adaption on the network, the model feeds tensor values to carry out CNN computation, including the training and testing process. This experiment converts all the user features into vectors to allow the training process. The profile features depict some explicit user data, including identity, username, location, profile image information, etc.

All of the input samples are converted into numerical arrays during data preprocessing. The model should transform the array into tensors to allow matrix computation before beginning the training process. After converting the features to byte representation values, the model generates the input vectors. To support the quality of training in the learning issue, the feature extraction procedure is critical. In most cases, the learning process is required to train the model in categorizing user profiles based on implicit and explicit information. The explicit characteristics of raw user metadata such as information credibility (Castillo et al. 2011) and user identity linkage (Shu et al. 2017).

Implicit characteristics are not explicitly stated in the profile. They are often helpful for representing user profiles for specialized jobs. Age, gender, and personality are all typical implicit profile characteristics. We use the explicit attributes in this research and input them into the model. As a result, we collect the users' feature types to build the classifier, which is a crucial aspect of the OSN connection for inferring friendship (He, et al. 2020).

The data preprocessing offers the dataset as a table in the dimensional grid in this section. The table's rows represent individual components, while the columns represent the number of characteristics. Before changing the features matrix's integer, the dataset containing features and label refers to rows and columns. Features and targets in data preprocessing should be numerical arrays. As a result, before feeding the integer values to the classifier, we must transform them into tensor forms.

5 Experiment result

5.1 Proposed CNN

We train the model using convolutional architecture in the first step, followed by a fully connected layer. In a fully connected layer, all input units have a separate weight from each output unit. For n inputs and m outputs, the number of weights is $n * m$. The research uses SGD to train the model and adjusts the learning rate to decide the stride size to obtain a (local) minimum. During the training and testing procedure, we used a gradient descent (SGD) optimizer with momentum. To minimize the objective function j(x) using the model's parameters, we use gradient descent. We use many optimizers with epoch = 12 to train the proposed CNN, and we use gradient descent to minimize the objective function j(x). Using the suggested function in a sample of user profile characteristics, Fig. 3 demonstrates the performance in our model graph's training and testing procedure.

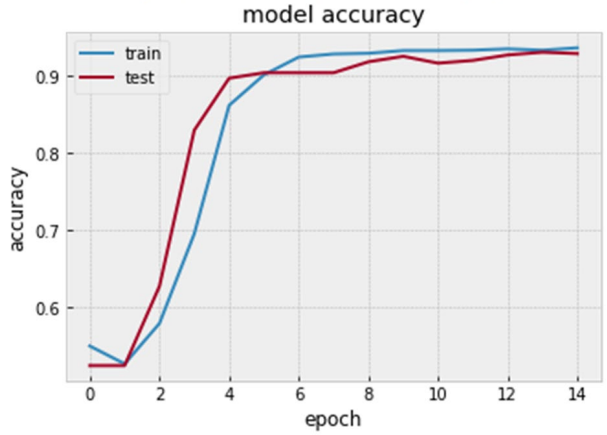
We achieve the best accuracy and little loss in the training and testing process while training the model using SGD + momentum for CNN architecture, based on many testing processes. To prove the model's performance, we calculate Recall, Precision, F1-Score, and ROC using evaluation measures. Then, to prove our proposed model's performance, we compare it to other learning algorithms using the same dataset and hyperparameter settings. Table 3 displays the Recall, Precision, and F1-Score with different methods.

Because it is relevant in the classification context, we depict the ROC curve to measure sensitivity and specificity. To widen the ROC curve of binary classification, it must transform the output into binary values. As the criteria change, it becomes a relative operating characteristic curve (TPR and FPR). This experiment also calculates the AUC score to measure sensitivity and specificity in various thresholds without altering the threshold. The AUC measures how well a model can distinguish between positive and negative classifications. In the classifier, we use binary classification.

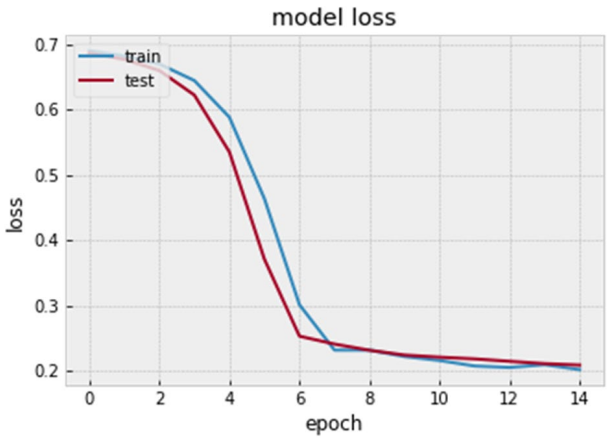
The AUC in this research is 0.9547, which indicates that the value of the area is essential in metric assessment. The AUC works by calculating a rating based on the distance between two classes, which shows the model's ability to distinguish between positive and negative categories. As seen above, the suggested CNN is capable of making accurate level predictions and beats other traditional learning methods. We discovered that our model produces more accurate results and is more efficient.

5.2 Comparison of activation functions

In this section, we train users' characteristics to test the suggested classification algorithm using learning parameters. In this process, we use a smaller sample dataset and train the model using CNN architecture with and without a linear classifier at the final fully connected layer. We also compute the loss function using the local variable notion. Its purpose is to determine the loss gradient in terms of weight and bias. In CNN architecture, we examine several activation functions, such as SoftMax and RunMax, and tweak



a)Accuracy Score



b) Loss Score

Fig. 3 Performance result of the proposed CNN with fake profile features

Table 3 Comparison of precision, recall and F1-Score among algorithms

Algorithm	Precision (%)	Recall (%)	F1-Score (%)
Naïve bayes	86.91	86.95	87.02
GB (n estim = 50)	90.65	90.69	91.01
LR	90.48	90.58	90.60
SVM (rs = 31.6)	90.04	87.34	87.24
Proposed CNN	94.01	93.22	93.41

optimization techniques using hyperparameters. Choosing a hyperparameter value is critical for improving network training quality in regular deep learning. Before adding the nonlinear activation functions, we modify the same hyperparameter to retain the training result's dependability and stability and pick the network with the best performance.

In the training and testing process, we tune the CNN hyperparameters and set different linear classifiers at the end of the last hidden layer. We test CNN with and without the RunMax function to measure classifiers' performance in the same dataset. By adding the RunMax function, we gain better accuracy than another state-of-the-art method, SoftMax, in the same environment and hyperparameters. Table 4 displays the comparison performance among different classifiers to measure training accuracy and testing accuracy.

The training accuracy of the neural network with the RunMax is 0.8334, and the testing accuracy is 0.8172, which is greater than other typical CNN designs. The results suggest that by setting an adequate gradient descent in the training process, the proposed CNN with RunMax may deliver a promising outcome. The function may optimize the neural network's performance by demonstrating on the dataset. In the classification domain issue, training the model using an optimizer may improve accuracy. However, if a CNN with many layers and more neurons cannot increase predictive capacity, we identify a downside, and this approach remains a training time weakness.

As the standard activation function, Softmax converts all logits into probabilities. However, in several classification cases, the empty feature maps still produce a result. Softmax function can also suffer from a bottleneck of the representational capacity of neural networks in language modeling or the 1D dataset. The proposed nonlinear activation function is a new way to achieve accurate classification results to train features as the dataset to improve the Softmax drawbacks. The RunMax classifier becomes a useful technique if we train a model with a 1D dataset like OSN features.

Notably, adding the Gaussian function to develop the RunMax can be crucial to improving the classifier performance at the final layer of the neural network. Based on the experiment, we obtain that RunMax can achieve a better result than the conventional SoftMax activation function. In the fake account problem, this technique's significance relates to information security and rests on the ability of the model to distinguish fake profiles within similar contexts. To the best of our knowledge, no similar works presented the idea of binary classifying malicious profiles by constructing a novel learning model.

Table 4 Comparison among classifiers with the CNN architecture

Activation type	Training accuracy (%)	Testing accuracy (%)
CNN + No activation	80,51	79,54
CNN + SoftMax	82,31	81,33
CNN + RunMax	83,34	81,72

Therefore, this study can be a promising solution to achieve more accurate interpretability for phishing detection in the OSN.

6 Conclusion

Many researchers have proposed various techniques to deal with malicious account detection issues. Some critical aspects of malicious account classification are fake attributes, suspicious URLs, and toxic comments on the OSN wall. The conventional methods such as self-rules, statistical, or even common learning-based remain drawbacks. Instead of using a standard model to construct our method, we propose the malicious classification model with a novel learning algorithm.

Inspired by deep learning performance in various problems, we develop the classification model with the learning classifier concept. Our goal is to understand the state of the art in a malicious account problem in the first line of research, including novel techniques and approaches. Based on the experiments, the proposed method can harvest potential results to deal with malicious account detection using a deep learning classifier. We gain three research achievements with large user features using the neural network training process. Our proposed methods obtain several promising achievements in malicious classification as follows:

By developing a nonlinear activation function at the final layer to increase CNN's performance, we find the neural network architecture using the RunMax with a Gaussian distribution can improve SoftMax performance to address fake accounts issues with binary classification. This experiment can produce higher Precision, Recall, F1-Score, and AUC scores with user profile features as a dataset based on the two classes' separation. Based on the comparison table, the proposed model produces a higher accuracy score than other activation algorithms with the benchmark dataset. By implementing the generic function network, we obtain a promising improvement in the CNN graph's performance. Thus, it can be a real-time solution to address malicious account detection issues in practical areas.

Based on the experimental result, our technique can accurately detect the fake profile by producing Precision = 94.00, Recall = 93.21, and F1-Score = 93.42. By assessing the ranking based on the separation of the two classes, the suggested model may reach a better AUC score = 0.9547. To summarize, the model may be a valuable method for developing a classification model based on large datasets of profile data. Our findings may encourage individuals to work on social network data and develop solutions to help them make better judgments based on facts. This approach can increase the accuracy of predicting whether a profile is fake or real.

By implementing the network with the generic functions, we get better improvement in the CNN performance with supervised learning to classify fake and regular accounts. Notably, adding the Gaussian function to the proposed nonlinear classifier becomes essential for constructing a better neural network to achieve better performance. In future research, the subsequent investigation requires building a novel model using a more complex technique like ontology engineering or more sophisticated architecture including GAN and GCN.

Appendix

RunMax: complement material

To deal with the SoftMax bottleneck problem, we propose RunMax given as follows:

Definition 1 RunMax is defined as

$$\left[f(z)_i\right] = \frac{\exp(z_i)\varphi(z_i)}{\sum_{m=1}^{M}\exp(z_m)\varphi(z_m)}, \quad i = 1 \ldots M \tag{10}$$

where$\mathcal{G}(\cdot)$ represents a Gaussian function $f(z) = \exp\left(-z^2\right)$ with derivative $f'(z) = -2z\left(\exp\left(-z^2\right)\right)$

Theorem 1 *Let $z \in S$ as the input of RunMax $f(z)$ and SoftMax $f_s(z)$. Let S as a d-dimensional vector space $1 \in S$, thus the range of Softmax is a subset of RunMax.*

$$\left\{f_s(z)|z \in S \subseteq f(z)|z \in S\right\} \tag{11}$$

Proof If we have $1 \in S$, it can be described as $S = \left\{ \sum_{l=1}^{d-1} k'^{(l)}u'^{(l)} + k'^{(d)}1 | k'^{(l)} \in R \right\}$ where $u'^{(l)}$ is $(l = 1, \ldots, d-1)$ and 1 are linearly independent vectors. The arbitrary part of S can be represented as $\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)} + k'^{(d)}1$, and thus, we can write $z = \sum_{l=1}^{d-1} k'^{(l)}u'^{(l)} + k'^{(d)}1$. For the output of softmax,

$$\left[f_s(z)_i\right] = \frac{\exp\left(z_i\right)}{\sum_{m=1}^{M}\exp\left(z_m\right)} \tag{12}$$

By substituting $z = \sum_{l=1}^{d-1} k'^{(l)}u'^{(l)} + k'^{(d)}1$ to the $\left[f_s(z)_i\right]$ function, we have:

$$\begin{aligned}\left[f_s(z)_i\right] &= \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_i + k'^{(d)}\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_m + k'^{(d)}\right)} \\ &= \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_i\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_m\right)}\end{aligned} \tag{13}$$

Thus, the range of SoftMax become as follows:

$$\begin{aligned}&\left\{f_s\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right] + k'^{(d)}1\right)k'^{(d)} \in \mathbb{R}\right\} = \\ &\left\{\frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_i\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_m\right)} | k'^{(l)} \in \mathbb{R}\right\}\end{aligned} \tag{14}$$

Besides, by replacing $z = \sum_{l=1}^{d-1} k'^{(l)}u'^{(l)} + k'^{(d)}1$ to the $\left[f(z)_i\right]$ RunMax function, output of RunMax becomes as follows:

$$\left[f(z)_i\right] = \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_i\right)\varphi\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_i + k'^{(d)}\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_m\right)\varphi\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_m + k'^{(d)}\right)} \tag{15}$$

When $k'^{(l)}$ are fixed for $(l = 1, \ldots, d-1)$ and $k'^{(d)} \to +\infty$, we get the following equality:

$$\begin{aligned}&\lim_{k'^{(d)}\to+\infty} \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_i\right)\varphi\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_i + k'^{(d)}\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_m\right)\varphi\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_m + k'^{(d)}\right)} \\ &= \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_i\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_m\right)}\end{aligned} \tag{16}$$

hence $\lim_{k\to+\infty}\mathcal{G}(v,k) = 1$ when v is fixed. Considering Eq. 17, RunMax has following relation:

$$\begin{aligned}&\left\{\left(\frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_i\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)}\right]_m\right)}\right)k'^{(l)} \in \mathbb{R}\right\} \\ &= \left\{f(z)|z \in s'\right\} \subseteq f(z)|z \in S\}\end{aligned} \tag{17}$$

We can calculate $S' = \left\{\sum_{l=1}^{d-1} k'^{(l)}u'^{(l)} + k'^{(d)}1 | k'^{(l)} \in R\right.$ $\left. for\left(l = 1, \ldots, d-1, k'^{(d)} \to +\infty\right) \subset S\right\}$. Based on Eq. (16), we can look that the range of RunMax contains the range of SoftMax. Therefore, we get $\left\{f_s(z)|z \in S \subseteq f(z)|z \in S\right\}$. Theorem 1 describes that if $1 \in S$, so the range of RunMax is able to be larger than that of SoftMax. The assumption $1 \in S$ means that there exist inputs of which outputs are the equal probabilities for all labels as $p_\theta\left(y_i|x\right) = \frac{1}{M}$ for all i.

References

Abeer A-M, Maha H, Nada A-S, Hemalatha M (2016) Security issues in social networking sites. Int J Appl Eng Res 11(12):7672–7675

Al-Qurishi M, Al-Rakhami M, Alamri A, Alrubaian M, Rahman SMM, Hossain MS (2017) Sybil defense techniques in online social networks: a survey. IEEE Access 5:1200–1219

Al-Zoubi AM, Alqatawna J, Faris H, Hassonah MA (2021) Spam profiles detection on social networks using computational intelligence methods: the effect of the lingual context. J Inf Sci 47(1):58–81

Anglano C, Canonico M, Guazzone M (2017) Analysis of telegram messenger on android smartphones. Digital Investig 23:31–49

Baingana B, Giannakis GB (2016) Joint community and anomaly tracking in dynamic networks. IEEE Trans Signal Process 64(8):2013–2025

Bindu P, Thilagam PS (2016) Mining social networks for anomalies: Methods and challenges. J Netw Comput Appl 68:213–229

Castillo C, Mendoza M, Poblete B (2011) Information credibility on twitter. WWW Conference

Chiluka N, Andrade N, Pouwelse J, Sips H (2015) Social networks meet distributed systems: towards a robust sybil defense under churn. In: Proceedings of the 10th ACM symposium on information, computer and communications security, ASIA CCS '15, ACM: 505–518

Durst S, Zhu L (2016) The darpa twitter bot challenge

Egele M, Stringhini G, Kruegel C, Vigna G (2017) Towards detecting compromised accounts on social networks. IEEE Trans Dependable Secure Comput 14(4):447–460

He C et al (2020) CIFEF: combining implicit and explicit features for friendship inference in location-based social networks. KSEM, Texas

Jie HJ, Wanda P (2020) RunPool: a dynamic pooling layer for convolution neural network. Int J Comput Intell Syst 13(1):66–76

Kevin K, Alexander D, Matthias S (2021) Does my social media burn?—identify features for the early detection of company-related online firestorms on twitter. Online Soc Netw Media 25:100151

Kökciyan N, Yolum P (2016) ProGuard: a semantic approach to detect privacy violations in online social networks. IEEE Trans Knowl Data Eng 28(10):2724–2737

Liu B-H, Hsu Y-P, Ke W-C (2014) Virus infection control in online social networks based on probabilistic communities. Int J Commun Syst 27:4481–4491

Meier A, Johnson BK (2022) Social comparison and envy on social media: a critical review. Curr Opin Psychol 45:101302. https://doi.org/10.1016/j.copsyc.2022.101302

Mohaisen A, Hollenbeck S 2013 Improving social network-based sybil defenses by rewiring and augmenting social graphs. In: Revised selected papers of the 14th international workshop on information security applications, WISA: 8267

Muftic S, Abdullah N, Kounelis I (2016) Business information exchange system with security, privacy, and anonymity. J Electr Comput Eng 251:7093642

Nadav V, Gal-Oz N, Ehud G (2021) A trust based privacy providing model for online social networks. Online Soc Netw Media 24:100138

Ni X, Luo J, Zhang B, Teng J, Bai X, Liu Bo, Xuan D (2016) A mobile phone-based physical-social location proof system for mobile social network service. Sec Commun Netw 9:1890–1904

Prabhu Kavin B, Sagar Karki S, Hemalatha DS, Vijayalakshmi R, Thangamani M, Haleem SLA, Jose D, Tirth V, Kshirsagar PR, Adigo AG (2022) Machine learning-based secure data acquisition for fake accounts detection in future mobile communication networks. Wireless Commun Mobile Comput 2022:1

Ruan X, Wu Z, Wang H, Jajodia S (2016) Profiling online social behaviors for compromised account detection. IEEE Trans Inf Forensics Secur 11(1):176–187

Satish Kumar A, Revathy S (2022) A hybrid soft computing with big data analytics based protection and recovery strategy for security enhancement in large scale real world online social networks. Theor Comput Sci 927:15–30. https://doi.org/10.1016/j.tcs.2022.05.018

Sharma V, You I, Kumar R (2017) ISMA: intelligent sensing model for anomalies detection in cross platform OSNs with a case study on IoT. IEEE Access 5:3284–3301

Shu K, Wang S, Tang J, Zafarani R, Liu H (2017) User identity linkage across online social networks: a review. ACM SIGKDD Explor Newsl 18(2):5–17

Simon WJ, Krupnik TJ, Aguilar-Gallegos N, Halbherr L, Groot JCJ (2021) Putting social networks to practical use: improving last-mile dissemination systems for climate and market information services in developing countries. Climate Serv 23:215

Takahashi T, Panta B, Kadobayashi Y, Nakao K (2018) Web of cybersecurity Linking, locating, and discovering structured cybersecurity information. Int J Commun Syst 31:3470

Tan Z, Ning J, Liu Y, Wang X, Yang G, Yang W (2016) ECRModel: An elastic collision-based rumor-propagation model in online social networks. IEEE Access 4:6105–6120

Uppada SK, Manasa K, Vidhathri B et al (2022) Novel approaches to fake news and fake account detection in OSNs: user social engagement and visual content centric model. Soc Netw Anal Min 12:52

Vigliotti MG, Hankin C (2015) Discovery of anomalous behaviour in temporal networks. Soc Netw 41:18–25

Wanda P, Jie HJ (2020) DeepProfile: finding fake profile in online social network using dynamic CNN. J Inf Secur Appl 52:102465

Wanda P, Jie HJ (2021) DeepFriend: finding abnormal nodes in online social networks using dynamic deep learning. Soc Netw Anal Min 11:34

Wanda P, Endah MH, Jie HJ (2020) DeepOSN: Bringing deep learning as malicious detection scheme in online social network. IAES Int J Artif Intell (IJ-AI) 9(1):146

Yang Z, Dai Z, Salakhutdinov R, Cohen WW (2018) Breaking the softmax bottleneck: a high-rank RNN language model. In: International conference on learning representations. https://arxiv.org/abs/1711.03953.