OPEN ACCESS
VOL. 7, NO. 1, PP.022-032, NOVEMBER 2024

ISSN : 2622-8106 (ONLINE)
DOI: 10.20895/INISTA.V7I1.1622

Journal of Informatics, Information System, Software Engineering and Applications (INISTA)

Deep Learning Model Based on Particle Swam Optimization for Buzzer Detection

Ika Kurniawati*[1]

#Department Information Technology, Nusa Mandiri University
Jakarta, Indonesia

[1] ika.iki@nusamandiri.ac.id

Received on 10-10-2024, revised on 20-11-2024, accepted on 28-11-2024

Abstract

Along with the development of the internet, the presence of buzzers is increasingly widespread on social platforms, especially on Twitter. Buzzers have been significant in shaping public opinion, disseminating misinformation, and engaging in harassment and intimidation of social media users. Therefore, an effective detection algorithm is needed to detect buzzer accounts that endanger social networks because they affect neutrality. This study experiments on particle swarm optimization (PSO) based on deep neural networks and Ada Boost based on deep neural networks, which were conducted on 1000 balanced datasets to obtain the best model for detecting buzzer accounts. In classification, PSO is used to find the optimal set of parameters or feature subsets for a classifier by simulating the movement of particles towards the best solution based on classification accuracy. Ada Boost is a classification technique that combines multiple weak classifiers by giving more weight to misclassified data points in each iteration, ultimately creating a strong classifier that improves accuracy through iterative correction. The results show that the performance of the PSO-based Deep Neural Network is the best, with 98.90% accuracy compared to the Ada Boost-based Deep Neural Network with 95.30% or without feature weight and boosting algorithms with 46.60%. This clearly shows the superiority of the proposed method. These results are expected to help maintain neutral information on social media and minimize noise in the data that will be used for sentiment analysis research.

Keywords: Ada Boost, Buzzer, Detection, Deep Learning, Particle Swarm Optimization

Corresponding Author:
*Ika Kurniawati
Department of Information Technology, Nusa Mandiri University
Jl. Kramat Raya No. 18, Senen, Jakarta Pusat, Indonesia
Email: ika.iki@nusamandiri.ac.id

I. INTRODUCTION

Social media is now not only used as a means of friendship and finding friends but has also been widely used for other activities. Twitter social media is currently a very popular communication medium among internet users. A buzzer is a phenomenon currently seen in social media, referring to a user who holds significant influence over others. Buzzers are often viewed as key figures in social connections within social media platforms [1]. Buzzers are social media users who have influence and can influence other users [1], absorbed from the term buzz marketing, which refers to a marketing strategy to increase visibility through fabricated marketing messages. Lately, the term buzzer has shifted its meaning on social media. Now, buzzers can be interpreted as social media accounts managed by individuals or companies that have a large number of followers and also help spread hoax information and negative statements in political campaigns.

In Indonesia, the effects of buzzers were felt during the Jakarta gubernatorial election event in 2012. Since then, the buzzer phenomenon in Indonesia has continued to increase every year. The peak can be said to have occurred in the 2019 presidential election. Politicians use buzzers to attack their political opponents by spreading false information and spreading their political opponents' personal information. The effects of

buzzers in Indonesia have reached a very bad stage that can divide society [2]. Research conducted by [3] shows that the role of buzzers on social media in Election and Regional Election activities has reached a severe stage. The emergence of buzzers in the world of politics also shows a trend that always increases every year [4]. Detecting buzzers can be called an important job. After all, the data generated by buzzer accounts can affect the validation of other research, such as sentiment analysis, because the data can be considered noise [5].

Previous research [6], which detected bot accounts on Twitter using a deep learning model, produced an F1 score of 0.999 in the first test and an F1 score of 0.995 in the second test. Several studies have been conducted previously by [1] detecting buzzer accounts using dynamic data from Twitter. Calculation of buzzer values using eigenvector centrality modifications that include dynamics that occur in social media, such as interactions within users, user activity levels, user influence levels, and user node values better represent the actual conditions of social media. The results of the study show that buzzer values indicate buzzers depending on their influence and activity values. Although a user has a large eigenvector centrality modification value, the user is not considered a buzzer if the user is inactive or has no influence in the observed social media graph [1].

The best accuracy and precision values in the Ada Boost, while the best recall value was obtained in Extreme Gradient Boosting in the study of detected buzzer accounts on Twitter by looking at the impact of the features used and selected based on Mutual Information. This study used boosting algorithms, including Ada Boost, Gradient Boosting, Extreme Gradient Boosting, and Histogram-based Gradient Boosting [5]. The classification of buzzer accounts carried out by [7] used the C4.5 Classification Algorithm. This study aims to design a model and build a buzzer account classification system. The results of the model test can be seen through the test results based on the Confusion Matrix in stages using 10,540 tweet data, and an accuracy value of 88.5% was obtained.

The buzzer classifier carried out by [8] used four features: number of following, number of followers, sentiment value of the latest tweets, and account age. The Gaussian variant of the Naive Bayes algorithm will be used to classify buzzer and non-buzzer accounts. The results of this study show the performance of the buzzer account classification model, which has an accuracy value of 80%. In a study conducted by [9] detecting political buzzer accounts. The dataset was collected through the Instagram Graph API. The features used were based on posting time, images, hashtags, commonly followed accounts, and post frequency. The classification algorithms used were Naïve Bayes, Support Vector Machine, and Random Forest. This study provides a little insight into what happens on social media and the theory of how to overcome these problems. These arguments are expected to help identify buzzers in real life, thus helping to maintain neutral information with social media in Indonesia [9].

Data classification is needed before identifying buzzers. In this article, the data is first classified into what groups. This classification is done based on data features. Previous studies have used several methods to detect buzzers, such as Naive Bayes, Support Vector Machines, Random Forest, and feature-based detection [9]. Classification is the process of organizing and giving meaning to information that is useful for determining or establishing the suitability of events, ideas, goods, and people. Classification aims to group information based on similarities and characteristics into appropriate classes. In general, classification must go through several stages, namely data collection, data set preparation, data pre-processing, data set division, data training, algorithm use, model evaluation, and testing [10].

Based on previous research, it can be seen that deep learning algorithms can be used to detect an object. The author combines various findings from previous studies and combines several steps in the study in the hope of increasing the success rate of this study. The proposed model carries out the task of detecting buzzer accounts on Twitter with a deep neural network algorithm based on optimizing weight PSO, a deep neural network based on the Ada Boost enhancement algorithm, and a deep neural network without optimizing weight and boosting. This study compares the accuracy and ROC curves of all models. In summary, this paper is expected to contribute to providing a reference model for the buzzer detection framework to academics when researching sentiment analysis using Twitter data and to help ensure that information disseminated through social media remains neutral.

II. RESEARCH METHOD

In this section, the steps of the method used in this study will be explained. The method used consists of 4 steps, namely dataset collection, feature selection, experimentation, and evaluation. The steps can be seen in Figure 1.

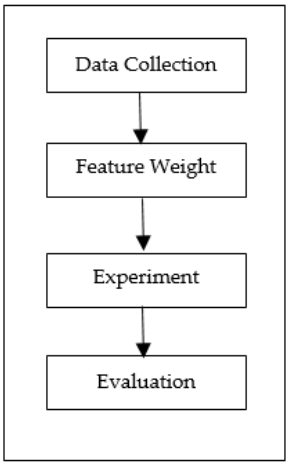


Fig. 1. Diagram of Research Methods

A. Dataset Collection

Datasets used in this study, each of which comes from the kaggle.com site [19], a site for finding and publishing datasets for machine learning developers. The dataset has as many as 1000 records consisting of 494 buzzers and 506 non-buzzers with the following variables in Table I.

TABLE I. FEATURE OF DATASETS

Tweet	The text content of the tweet
Retweet Count	The number of times the tweet has been retweeted
Mention Count	The number of mentions in the tweet
Follower Count	The number of followers the user has
Verified	A Boolean value indicating whether the user is verified or not
Location	The location associated with the user
Created At	The date and time when the tweet was created
Hashtags	The hashtags associated with the tweet
Label	A label indicating whether the user is a buzzer (1) or non-buzzer (0)

B. Feature Weight Particle Swarm Optimization (PSO)

Feature selection using the PSO algorithm is used to select the best attributes from the data and reduce irrelevant attributes to increase the effectiveness and efficiency of the classification algorithm's performance. PSO optimizes classification problems by moving particles based on the position and speed of each particle (swarm). The best fitness value influences the movement of particles in the swarm, which is carried out at each iteration until it reaches the maximum iteration. The maximum iteration is a test to determine the best fitness value at that iteration [10]. The use of PSO feature weight can improve attributes in the data set and make decisions based on optimal parameters from the previous classification, thereby increasing the accuracy of results [11]. PSO is a straightforward optimization method that highlights input features [12]. As a well-known bio-inspired algorithm, PSO draws on the social behaviours seen in bird flocking to address optimization challenges. To preserve the diversity of the swarms, some research has explored multi-swarm strategies [13].

PSO is a popular optimization technique inspired by the social behaviour of birds flocking or fish schooling. In the context of feature selection, PSO is used to search through the space of all possible feature subsets to identify the most relevant features that contribute to the accuracy of a predictive model. The goal is to reduce dimensionality by selecting a subset of features that leads to a simpler, more interpretable model while maintaining or improving its performance [21].

In PSO, each particle flies in the search space with a velocity adjusted by its flying memory and its companion's flying experience. Each particle has its objective function value, which is decided by a fitness function in Equation (1):

$$v_{id}^{t} = w \times v_{id}^{t-1} + c_1 \times r_1\left(p_{id}^{t} - x_{id}^{t}\right) + c_2 \times r_2\left(p_{gd}^{t} - x_{id}^{t}\right) \quad (1)$$

Where i represents the i-th particle and d is the dimension of the solution space, c_1 denotes the cognition learning factor and c_2 indicates the social learning factor, r_1 and r_2 are random numbers uniformly

distributed in (0,1), p_{id}^t and p_{gd}^t stand for the position with the best fitness found so far for the *i*th particle and the best position in the neighbourhood, v_{id}^t and v_{id}^{t-1} are the velocities at time t and time $t - 1$, respectively, and x_{id}^t is the position of i-th particle at time t. Each particle then moves to a new potential solution based on the Equation (2):

$$x_{id}^{t+1} = x_{id}^t + v_{id}^t \, , d = 1,2, \dots , D \tag{2}$$

Kennedy and Eberhart [21] proposed a binary PSO in which a particle moves in a state space restricted to 0 and 1 on each dimension in terms of the changes in probabilities that a bit will be in one state or the other. See in Equation (3) and (4).

$$x_{id} = \begin{cases} 1, rand() < S(v_{i,d}) \\ 0 \end{cases} \tag{3}$$

$$S(v) = \frac{1}{1 + e^{-v}} \tag{4}$$

The function $S(v)$ is a sigmoid limiting transformation, and $rand\ ()$ is a random number selected from a uniform distribution in [0.0, 1.0].

In PSO for feature selection, each particle in the swarm represents a candidate solution, which is a binary vector indicating which features are selected (1) or not selected (0). The swarm's population is initialized randomly, where each particle's position corresponds to a possible feature subset, and its velocity determines how the position might change over time. The initial position of the particles is often chosen based on random binary values (0 or 1) for each feature, representing whether a feature is included in the subset [21].

The fitness function plays a crucial role in evaluating the quality of each particle's solution. Typically, the fitness function is designed to assess the predictive performance of a model (such as a classifier) that uses the selected subset of features. Common fitness measures include accuracy, precision, recall, or the area under the ROC curve. The fitness function can also include a penalty for selecting too many features to prevent overfitting, which encourages feature subsets with lower dimensionality. This ensures that the selected feature set strikes a balance between simplicity (fewer features) and performance (better accuracy).

In summary, PSO in feature selection leverages the collective intelligence of a swarm of particles to explore and refine potential feature subsets. Through iterative adjustments based on individual and collective experiences, PSO is capable of selecting a subset of features that enhances model performance while reducing complexity and overfitting [21].

C. Ada Boost

The Ada Boost algorithm adjusts the weight of each misclassified data point to help construct the next classifier. The greater the weight of the classifier, the more influence it has on the overall accuracy [5]. By modifying the weights of misclassified data and integrating these weaker classifiers, Ada Boost enhances its performance and improves classification accuracy. Furthermore, boosting algorithms merge several weak classifiers to create a more robust classifier [5]. Ada Boost technique is used to handle class imbalance. The results of the study show that the proposed method can provide impressive improvements in comparison results. Ada Boost technique is also used to improve the accuracy value of the bank direct marketing dataset, resulting in an accuracy value of 92.25% [14].

D. Deep Neural Network

Deep learning (DL) is a branch of machine learning that is modelled after the architecture of the human brain. Just as humans make decisions by analyzing using a structured series of logical thoughts, deep learning imitates it with an algorithm called a neural network [20]. DL allows machines to learn from data. The Neural Network (NN) used in DL is a Deep Neural Network (DNN), namely an NN with more than one hidden layer between the input and output. The number of hidden layers and neurons used is determined according to needs. DNN, in general, is a feedforward network. That is, the flow of data from input to output is not repeated [15]. Deep learning involves neural networks that have numerous layers and parameters. Most deep learning techniques utilize neural network architectures, which is why they are often called deep neural networks [16]. Deep neural networks are relatively simple in structure, as they consist of feedforward neural networks with many layers, as illustrated in Figure 2.

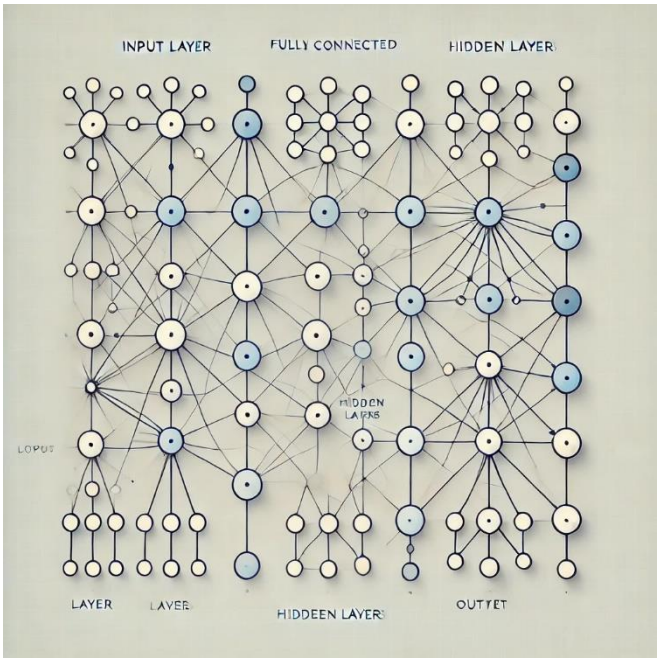


Fig. 2. A fully connected feedforward Deep Neural Network [17]

In DNN, the architecture is typically composed of multiple layers, each containing a certain number of neurons. The configuration of these layers is crucial for the network's ability to learn and generalize from data. DNN usually consists of three main types of layers: the input layer, hidden layers, and the output layer. The input layer takes in the raw features of the data, while the hidden layers transform these inputs into more abstract representations. The output layer produces the final prediction or classification [18].

Each neuron in a hidden layer performs a weighted sum of the inputs from the previous layer, followed by an activation function. The number of neurons in each layer is a key design choice that impacts the model's capacity. If the number of neurons is too small, the network might not be able to capture the complexity of the data, leading to underfitting. On the other hand, having too many neurons can lead to overfitting, where the network memorizes the training data rather than learning general patterns. This is why selecting the right number of neurons and layers is an important task in network design [18].

Hyperparameter tuning plays a critical role in optimizing the performance of a DNN. Hyperparameters are values set before training, such as the number of layers, the number of neurons in each layer, the learning rate, and the batch size. These hyperparameters can significantly affect the training process and the final model's performance. The design of a DNN involves careful choices regarding the number of layers, neurons per layer, activation functions, and optimization strategies. Hyperparameter tuning further refines the model to achieve optimal performance, while regularization techniques are used to ensure that the model generalizes well to unseen data [18].

DNN can be viewed as a series of functions where each neuron in a layer transforms its input using a mathematical operation, typically involving a weighted sum followed by a non-linear activation function (like ReLU or sigmoid). These layers work together to learn complex patterns in data by adjusting the weights through a process called backpropagation, where the model minimizes a loss function using optimization algorithms like gradient descent. In essence, DNN are composed of mathematical operations that iteratively refine the network's ability to make accurate predictions or classifications, learning from large amounts of labelled data [18].

1. Linear Transformation (Weights and Biases):

Each neuron in a DNN performs a linear transformation of its inputs. Suppose a neuron receives inputs $x_1, x_2, \ldots, x_n$ from the previous layer. The neuron computes a weighted sum of these inputs, where each input x_1 is multiplied by a corresponding weight w_i, and a bias term b is added. See in Equation (5).

$$z = w_1\, x_1 + w_2\, x_2 + \ldots + w_n\, x_n + \; b \tag{5}$$

Here, z is the output before applying any activation function.

2. Activation Function (Non-linear Transformation):

After z is calculated, a non-linear activation function $\sigma(z)$ (such as ReLU or sigmoid) is applied to t transform the linear z values into non-linear values. This gives the model the flexibility to learn more complex patterns in the data. The activation function helps the neural network to approximate complex functions that linear models cannot handle. See in Equation (6).

$$a = \sigma(z) \tag{6}$$

3. Layer-wise Computation:

DNN consist of multiple layers. In each layer, the output a from one layer becomes the input to the next layer. The forward pass through the network involves performing these weighted sums and activations layer by layer until the output layer is reached, which provides the model's final prediction.

4. Backpropagation (Gradient Descent):

During training, the goal is to minimize the difference between the predicted output and the actual label (the loss function). Backpropagation computes the gradients (partial derivatives) of the loss with respect to each weight, and these gradients are used to update the weights using an optimization algorithm like gradient descent. For a weight w_i , the update rule is:

$$w_i = w_i - \mathrm{n} \cdot \frac{\partial L}{\partial wi} . \tag{7}$$

Where n is the learning rate, and $\frac{\partial L}{\partial wi}$, is the gradient of the loss L with respect to the weight w_i.

5. Iterative Learning:

This process of forward propagation, loss calculation, and backpropagation continues iteratively, refining the weights to improve the network's predictions. As the weights are adjusted, the network gradually "learns" from the data and improves its performance.

E. Evaluation

The test results were validated using 10-fold cross-validation. This evaluation used the accuracy value and ROC (Receiver Operating Characteristics) parameters. ROC plots two metrics: True Positive Rate and False Positive Rate. AUC (Area Under the Curve) offers a comprehensive assessment of performance across all potential classification thresholds. AUC values range from 0 to 1, with a higher AUC indicating a more effective classifier [12].

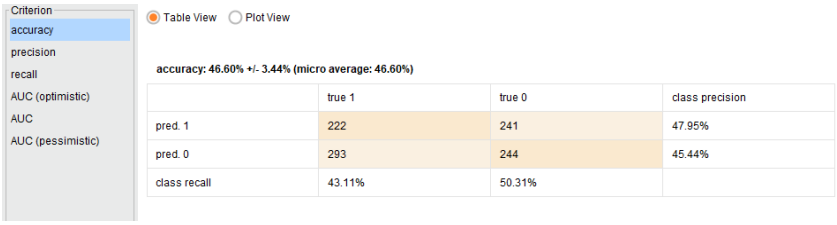
III. RESULTS AND DISCUSSION

A. Proposed Method

In this study, several experiments were conducted to find out how to get the most optimal model to detect buzzer accounts. Several experiments were conducted to test data using Deep Neural Network, Deep Neural Network based on PSO, and Deep Neural Network based on Ada Boost.

1. Test results using Deep Neural Network

According to the results obtained from testing the dataset with the Deep Neural Network model, as shown in Figure 3, the accuracy value is 46.60%, which shows that the classification for the buzzer prediction contains 222 true buzzers according to the prediction and 241 buzzer predictions are included in the true non-buzzer. The data that predicts non-buzzers is 293 true buzzers, and 244 are included in the true non-buzzer prediction.



	true 1	true 0	class precision
pred. 1	222	241	47.95%
pred. 0	293	244	45.44%
class recall	43.11%	50.31%	

Fig. 3. Accuracy Results of Deep Neural Network

The ROC curve results for the deep neural network are depicted in the image, showing an AUC (Area Under the Curve) value of 0.464, indicating a failure in classification performance.

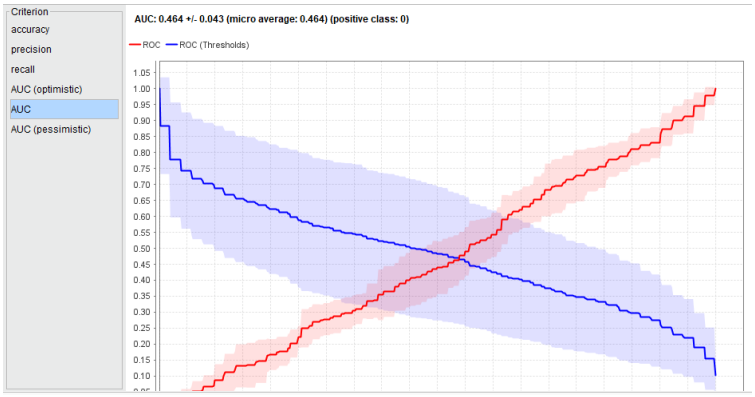
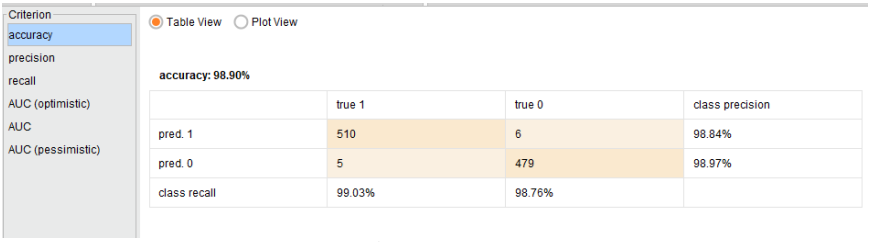


Fig. 4. ROC Curve Results of Deep Neural Network

2. *Test result using Deep Neural Network Based on PSO*

According to the results obtained from testing the dataset with the deep neural network model based on PSO, as shown in Figure 5, the accuracy value is 98.90%, which shows that the classification for buzzer predictions contains 510 true buzzers according to the prediction, and 6 buzzer predictions are included in true non-buzzers. Non-buzzer prediction data contains 5 true buzzers and 479 negative non-buzzers included in true non-buzzers.



	true 1	true 0	class precision
pred. 1	510	6	98.84%
pred. 0	5	479	98.97%
class recall	99.03%	98.76%	

Fig. 5. Accuracy Results of Deep Neural Network Based on PSO

The ROC curve results for the deep neural network based on PSO are displayed in Figure 7, showing an AUC (Area Under the Curve) value of 0.999, indicating excellent classification performance.

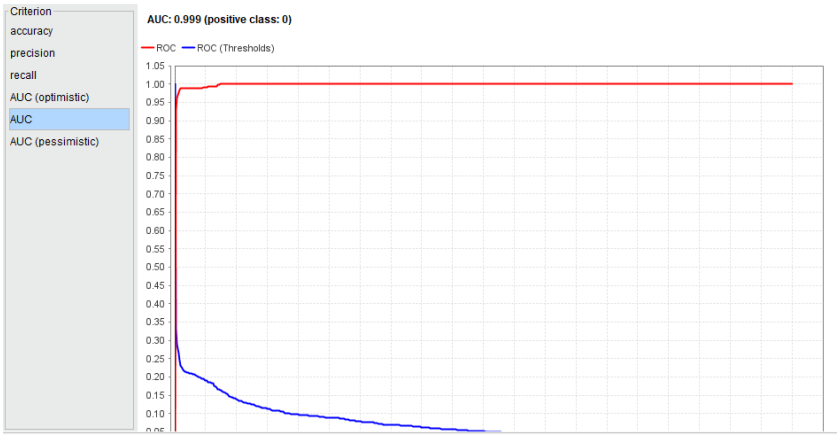
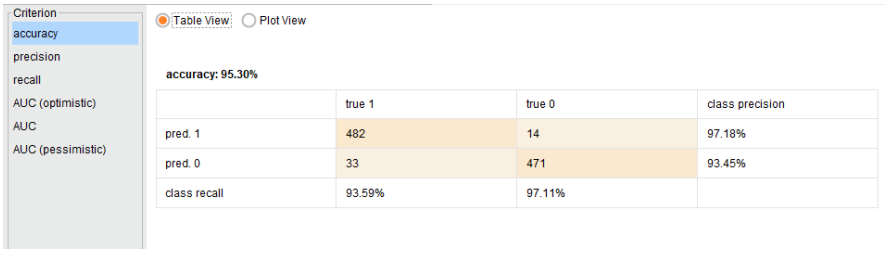


Fig. 6. ROC Curve Results of Deep Neural Network based on PSO

3. Test result using Deep Neural Network based on Ada Boost

According to the results obtained from testing the dataset with the deep neural network model based on Ada Boost, as shown in Figure 7, the accuracy value of 95.30% is obtained, indicating that the classification for buzzer predictions contains 482 true buzzers according to the prediction, and 14 predicted buzzers are included in true non-buzzers. Non-buzzer prediction data contains 33 true buzzers, and 471 non-buzzers are included in true non-buzzers.



	true 1	true 0	class precision
pred. 1	482	14	97.18%
pred. 0	33	471	93.45%
class recall	93.59%	97.11%	

Fig. 7. Accuracy Results of Deep Neural Network based on Ada Boost

The ROC curve results for the deep neural network based on PSO are displayed in Figure 8, showing an AUC (Area Under the Curve) value of 0.994, indicating excellent classification performance.

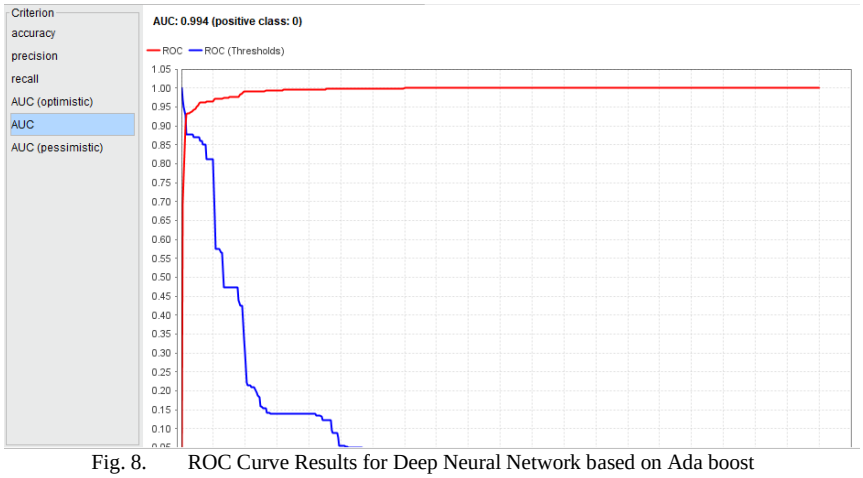


Fig. 8. ROC Curve Results for Deep Neural Network based on Ada boost

In Figure 9, the results show what attributes affect the classification results. The attributes that most influence in detecting buzzer or non-buzzer accounts include the top 3 most influential, namely created at (@) and hashtag, then the attributes that have the least influence based on the classification results are tweets and follower count.

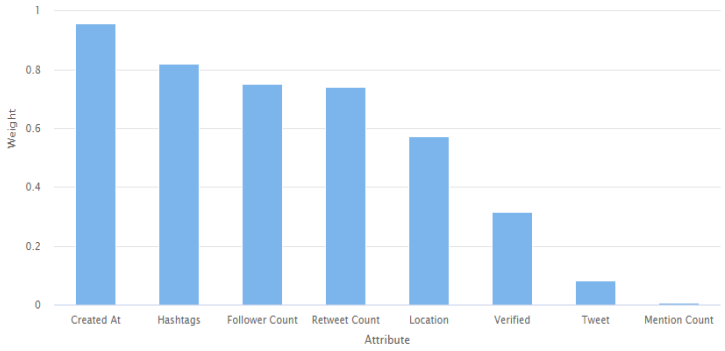


Fig. 9. Weight Attribute Distribution

B. Evaluation

This study used 10-fold cross-validation to avoid overfitting. After that, the accuracy and AUC scores for each proposed model will be compared. Table 2 is a comparison of accuracy and AUC against the proposed methods of Deep Neural Network + PSO, Deep Neural Network + Ada Boost, and Deep Neural Network without optimization and boosting algorithms. Based on the performance results of all algorithms, the proposed method, namely the PSO-based Deep Neural Network, increased accuracy by a 52.30% absolute increase. In comparison, the Deep Neural Network with Ada Boost increased by 48.70% compared to the Deep Neural Network without optimization and boosting algorithms, with an accuracy of 46.60%. Table II shows the superiority of the model proposed in this study.

TABLE II. COMPARISON OF PROPOSED MODELS RESULTS

Model	Accuracy (%)	AUC
Deep Neural Network	46.60	0.464
Deep Neural Network + PSO	98.90	0.999
Deep Neural Network + Ada Boost	95.30	0.994

Comparison of ROC curves through the AUC area value for the proposed model, namely PSO-based Deep Neural Network, increased by 0.479. In contrast, in Ada Boost-based Deep Neural Network, it increased by 0.476 compared to using only Deep Neural Network. PSO functions not only to select features but also to give weight to features that are considered important, helping to reduce the loss of information that can occur due to the removal of certain features. In addition, the proposed method achieves the highest AUC compared to other approaches.

While DNN are powerful tools for learning complex patterns in data, their performance can be significantly hindered by irrelevant or redundant features, especially in the case of noisy data like that in buzzer detection. Traditional DNN models often rely on manual feature engineering or can overfit if too many features are used. By integrating PSO, the proposed model iteratively searches for an optimal subset of features that maximizes classification accuracy while minimizing dimensionality and avoiding overfitting. This dual-purpose use of PSO enhances the learning process, enabling the model to focus on the most informative features and adjust its parameters more effectively [22].

The effectiveness of this method is validated through rigorous experimentation, demonstrating that the combination of PSO and DNN offers a more robust solution for buzzer detection. Such hybrid approaches hold the potential to address real-world challenges by improving the accuracy and efficiency of predictive models, particularly in fields involving complex, noisy, or high-dimensional data. In this study, the main contribution of this research lies in leveraging the strengths of PSO not only for optimizing the network's hyperparameters but also for selecting the most relevant features, thereby addressing both feature selection and model optimization simultaneously.

IV. CONCLUSION

This paper detects buzzer and non-buzzer accounts on the Twitter platform as the most popular social media application. There are 9 features used to classify buzzers and non-buzzers: Tweet, Retweet Count, Mention Count, Follower Count, Verified, Location, Created At, Hashtags, and Label. The proposed classification algorithm is Deep Neural Network integrated with PSO feature selection and Ada Boost. The effectiveness of the classification system is evaluated through a test dataset. The evaluation of the proposed method uses 10-fold cross-validation on 1000 datasets. The results of this experiment show that PSO-based DNN achieves the best performance with an accuracy of 98.90%, compared to Ada Boost-based DNN with an accuracy of 95.30% and 46.60% for DNN without feature selection and boosting algorithms.

A significant increase in accuracy is achieved by integrating PSO with DNN, compared to using DNN alone or with other boosting methods such as Ada Boost. This is because the hybrid model optimizes neural network training through the PSO algorithm, which improves the convergence speed and performance of the model. In addition, the integration of Ada Boost with DNN can improve the predictive ability of DNN by combining weak learners to minimize bias and variance. This study contributes to the field of classification by proposing an innovative framework that significantly improves the effectiveness of buzzer detection. This framework has the potential to be applied in both academic and practical fields, such as in sentiment analysis and efforts to maintain the neutrality of information dissemination in discussions on social media.

Further research can explore the refinement of the buzzer detection framework by incorporating advanced machine learning algorithms, such as deep learning or ensemble methods, to improve the accuracy in identifying bias in sentiment analysis in Twitter data. Additionally, investigating the role of context and user behaviour in sentiment analysis can provide deeper insights into how bias emerges and propagates in social media discussions. Exploring multimodal data integration (e.g., combining text, images, and hashtags) can also improve the robustness of sentiment detection systems.

REFERENCES

[1] M. T. Juzar and S. Akbar, "Buzzer Detection on Twitter Using Modified Eigenvector Centrality," *Proc. 2018 5th Int. Conf. Data Softw. Eng. ICoDSE 2018*, pp. 1–5, 2018, doi: 10.1109/ICODSE.2018.8705788.

[2] F. N. Yudianto and Y. Sibaroni, "Klasifikasi Hashtag Buzzer/Bot Menggunakan Algoritma Random Forest dengan Atribut Komunitas untuk Mengurangi Disinformasi pada Twitter," *e-Proceeding Eng.* , vol. 9, no. 3, pp. 1892–1901, 2022, [Online]. Available: www.trends24.in/indonesia/.

[3] C. Juditha, "Buzzer di Media Sosial pada Pilkada dan Pemilu Indonesia," *Semin. Nas. Komun. dan Inform.*, vol. 2019, pp. 199–212, 2019.

[4] B. Arianto, "Salah Kaprah Ihwal Buzzer: Analisis Percakapan Warganet di Media Sosial," *JIIP J. Ilm. Ilmu Pemerintah.*, vol. 5, no. 1, pp. 1–20, 2020, doi: 10.14710/jiip.v5i1.7287.

[5] A. Suciati, A. Wibisono, and P. Mursanto, "Twitter Buzzer Detection for Indonesian Presidential Election," *ICICOS 2019 - 3rd Int. Conf. Informatics Comput. Sci. Accel. Informatics Comput. Res. Smarter Soc. Era Ind. 4.0, Proc.*, 2019, doi: 10.1109/ICICoS48119.2019.8982529.

[6] H. Ping and S. Qin, "A social bots detection model based on deep learning algorithm," *Int. Conf. Commun. Technol. Proceedings, ICCT*, vol. 2019-Octob, pp. 1435–1439, 2019, doi: 10.1109/ICCT.2018.8600029.

[7] A. Maulana and S. Kuswayati, "Klasifikasi Akun Buzzer Pemilu Pada Media Sosial Twitter Berdasarkan Data Tweet Menggunakan Algoritma C4.5," *Naratif J. Nas. Ris. Apl. dan Tek. Inform.*, vol. 3, no. 02, pp. 30–35, 2021, doi: 10.53580/naratif.v3i02.132.

[8] Catur Arpal Perkasa, Amalia Andjani Arifiyanti, and Agus Salim, "Klasifikasi Akun Buzzer Pada Twitter Menggunakan Algoritma Naive Bayes," *J. Ilm. Tek. Inform. dan Komun.*, vol. 3, no. 1, pp. 01–12, 2023, doi: 10.55606/juitik.v3i1.363.

[9] A. J. Panatra, F. B. Chandra, W. Darmawan, H. L. H. S. Warnars, W. H. Utomo, and T. Matsuo, "Buzzer Detection to Maintain Information Neutrality in 2019 Indonesia Presidential Election," *Proc. - 2019 8th Int. Congr. Adv. Appl. Informatics, IIAI-AAI 2019*, pp. 873–876, 2019, doi: 10.1109/IIAI-AAI.2019.00177.

[10] T. M. Khalil and Sahid, "Klasifikasi Informasi Hoaks pada Media Sosial Twitter menggunakan Algoritma Random Forest berbasis Particle Swarm Optimization," *J. Kaji. dan Terap. Mat.*, vol. 8, no. November, pp. 199–209, 2022.

[11] R. F. Amir, I. A. Sobari, and R. Rousyati, "Penerapan PSO Over Sampling Dan Adaboost Random Forest Untuk Memprediksi Cacat Software," *Indones. J. Softw. Eng.*, vol. 6, no. 2, pp. 230–239, 2020, doi: 10.31294/ijse.v6i2.9258.

[12] I. Kurniawati and H. F. Pardede, "Hybrid Method of Information Gain and Particle Swarm Optimization for Selection of Features of SVM-Based Sentiment Analysis," *2018 Int. Conf. Inf. Technol. Syst. Innov. ICITSI 2018 - Proc.*, pp. 1–5, 2018, doi: 10.1109/ICITSI.2018.8695953.

[13] Y. Liu, G. Wang, H. Chen, H. Dong, X. Zhu, and S. Wang, "An improved particle swarm optimization for feature selection," *J. Bionic Eng.*, vol. 8, no. 2, pp. 191–200, 2011, doi: 10.1016/S1672-6529(11)60020-6.

[14] A. N. Rais, "Integrasi SMOTE Dan Ensemble AdaBoost Untuk Mengatasi Imbalance Class Pada Data Bank Direct Marketing," *J. Inform.*, vol. 6, no. 2, pp. 278–285, 2019, doi:

10.31311/ji.v6i2.6186.
[15] M. N. Baay, A. N. Irfansyah, and M. Attamimi, “Sistem Otomatis Pendeteksi Wajah Bermasker Menggunakan Deep Learning,” *J. Tek. ITS*, vol. 10, no. 1, 2021, doi: 10.12962/j23373539.v10i1.59790.
[16] P. P. Shinde and S. Shah, "A Review of Machine Learning and Deep Learning Applications," *Proc. - 2018 4th Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2018*, pp. 1–6, 2018, doi: 10.1109/ICCUBEA.2018.8697857.
[17] J. Ahmad, H. Farman, and Z. Jan, *Deep Learning Methods and Applications BT - Deep Learning: Convergence to Big Data Analytics*. Springer Singapore, 2019. doi: 10.1007/978-981-13-3459-7.
[18] Goodfellow. I, Bengio.Y, Courville, A. *Deep-learning*. MIT Press, 2016. doi: 10.1038/s41566-018-0231-3.
[19] https://www.kaggle.com/datasets/goyaladi/twitter-bot-detection-dataset (accessed Jun. 27, 2024).
[20] A. Oppermann, "What is Deep Learning and How does it work? | Towards Data Science." https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac (accessed Sep. 14, 2024).
[21] Kennedy, J., & Eberhart, R. C. *Particle swarm optimization*. Proceedings of the IEEE International Conference on Neural Networks, 1942-1948, 1995.
[22] Yang, X.-S., & He, X. *Particle Swarm Optimization*: *Basic Concepts, Variants, and Applications. Springer Handbook of Computational Intelligence*. Springer, 2013.