Hindawi
Security and Communication Networks
Volume 2023, Article ID 9820128, 1 page
https://doi.org/10.1155/2023/9820128

Retraction

Retracted: Effective Bots' Detection for Online Smartphone Game Using Multilayer Perceptron Neural Networks

Security and Communication Networks

Received 10 October 2023; Accepted 10 October 2023; Published 11 October 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] W. Tsaur, C. H. Tseng, and C. Chen, "Effective Bots' Detection for Online Smartphone Game Using Multilayer Perceptron Neural Networks," *Security and Communication Networks*, vol. 2022, Article ID 9429475, 10 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 9429475, 10 pages
https://doi.org/10.1155/2022/9429475

Research Article

Effective Bots' Detection for Online Smartphone Game Using Multilayer Perceptron Neural Networks

Woei-Jiunn Tsaur,[1,2] **Chinyang Henry Tseng**,[2] **and Chin-Ling Chen**[3,4,5]

[1]*Computer Center, National Taipei University, New Taipei City 237303, Taiwan*
[2]*Department of Computer Science and Information Engineering, National Taipei University, New Taipei City 237303, Taiwan*
[3]*School of Information Engineering, Changchun Sci-Tech University, Changchun 130600, Jilin Province, China*
[4]*School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, Fujian, China*
[5]*Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung City 413310, Taiwan*

Correspondence should be addressed to Chin-Ling Chen; clc@cyut.edu.tw

Received 3 November 2021; Revised 30 January 2022; Accepted 3 March 2022; Published 29 March 2022

Academic Editor: Mamoun Alazab

Online smartphone game bots can cause unfair behaviors and even shorten the game's life cycle. The random forest algorithm in machine learning is a widely used solution to identify game bots through behavioral features. Although the random forest algorithm can exactly detect more definite game bot players, some players belonging to the gray area cannot be detected accurately. Therefore, this study collects players' data and extracts the features to build the multilayer perceptron, neural network model, for effectively detecting online smartphone game bots. This approach calculates each player's abnormal rate to judge game bots and is evaluated on the famous mobile online game. Based on these abnormal rates, we then use K means to cluster players and further define the gray area. In the experimental evaluation, the results demonstrate the proposed learning model has better performance, not only increasing the accuracy but also reducing the error rate as compared with the random forest model in the same players' dataset. Accordingly, the proposed learning model can detect bot players more accurately and is feasible for real online smartphone games.

1. Introduction

With the popularity of smartphones, the market of mobile online games has become more and more prosperous in recent years. The mobile online game bot is an automated program playing games autonomously instead of human users. Without any touch from a human, bots can automatically move around in a virtual world. They behave in a programmed way and repetitively do patterned actions to collect game assets. Bots can gather game wealth faster than normal players because the bot program does not get tired [1, 2]. Consequently, bots will endanger the rights of mobile online game players. For a normal player, for example, after he/she practices continuously, his/her game score should grow slowly. However, if the player suddenly increases many thousand or even ten thousand scores within a short period of time, he/she may be abnormal to use game bots. When game bots are spread, the behaviors of using bots will break the rule of the whole game and lead to the dissatisfaction of normal players. Mobile online game bots have recently become an essential security issue that must be solved to protect game assets.

Game bots' detection is one of the defensive mechanisms against game bots [3]. However, it is frequently misjudged by players as game bots because some players may spend a lot of time playing the game, which makes their score data look higher than others, or a game bot may try to imitate a normal player to avoid being detected. Many machine learning methods [1, 2, 4–8] were developed to detect game bot attacks based on behavioral features. General machine

learning methods are to collect data from games' database first, and then, features will be extracted. These features are standardized to speed up the training process. After standardizing, all data are going to be separated into the training data and testing data. And the training data will be trained by some machine learning algorithms to build various models. Although these machine learning models can be used to exactly detect more definite game bot players, some players belonging to the gray area cannot be detected accurately. The existing works do not discuss the gray area of abnormal rate. These existing works use 50% to separate the normal and abnormal rates, but clusters of these two rates may overlap. So, the gray area is the cluster of detection rates between normal and abnormal rates. This work uses K means to generate the gray area cluster automatically. The rates in the gray area potentially result in most false positives and negatives. Therefore, this work will deal with these false positives and negatives in the gray area in order to increase detection accuracy.

The distribution of abnormal rates can be the clusters of high and low to predict the game bot and normal players. However, these two clusters can overlap to form a gray area such that we cannot use 50% to separate them easily. So, we need to find out those abnormal rates nearby 50%, and these rates form the gray area. Then, we can deal with these challenging rates that cause false positives and negatives. However, we require a method to find out these rates automatically instead of setting the manual border. Therefore, we adopt the K-means clustering to find out the gray area by setting $K = 3$ to cluster the rates into the normal, gray area, and abnormal clusters to generate the gray area cluster automatically. Most deep learning-based intrusion detection models do not identify and deal with the gray area directly.

In this study, we propose the multilayer perceptron (MLP) neural networks to effectively detect game bots, particularly in the gray area. As a neural network is a black-box model, it is extremely difficult for bot developers to recognize detection thresholds for neutralizing detection methods. In this study, we train the proposed deep neural network models on the famous mobile online baseball game named KANO. We find that the random forest algorithm is more frequently used in machine learning and thus choose it to compare with the proposed approach in experimental evaluation. In the experiments, we design and train four MLP models which have different optimizers, epochs, split of validation, and batch size, and we will choose the best model among them as the baseline of deep learning. We also employ the mobile online game players' dataset to train the machine learning's random forest model. According to the experimental results, the proposed best model increases the accuracy from 95.2% to 99.894% and reduces the error rate from 4.8% to 0.106%, as compared with the random forest model in the same players' dataset. Also, the proposed approach has lower FNR (False Negative Rate) and FPR (False Positive Rate) and therefore achieves better performance on detecting the gray area.

The key contributions of the study are that this paper adopts K means to generate the gray area cluster of abnormal rates automatically. These rates can result in false positives and negatives because they are nearby the 50% rate. By generating the gray area cluster, we can deal with these rates to adjust the best border to distinguish the normal and abnormal players and increase the detection accuracy. Moreover, the novelty of this work is that this work identifies the gray area by clustering the abnormal rates based on K means. The root cause of false positives and negatives comes from the gray area. Therefore, this study deals with the abnormal rate in the gray area to resolve the false positives and negatives and increase the accuracy.

The remainder of this study is organized as follows. In Section 2, the related work will be surveyed. Section 3 proposes the system model and detection approach based on the deep neural networks, including players' data feature extraction and transformation, the architecture for MLP neural networks, and gray area prediction and response. Section 4 conducts the experiments on the dataset of the mobile online game and further compares it with the random forest method in machine learning. Finally, some concluding remarks are presented in Section 5.

2. Related Works

2.1. Game Bot Attacks and Countermeasures. With the popularity of mobile online games, game bots have become a critical security issue that must be solved effectively. The behavior of using bots will break the fairness of the whole game and lead to the fact that normal players quit the game. For a normal player, his/her game score should grow slowly after he/she practices continuously. If the player suddenly increases many scores within a short period of time, he/she may be abnormal by using game bots. Based on the principle of using a game bot and the degree of endangering games, game bots are classified into three types: auxiliary, modification, and crack [1–3, 9]. The auxiliary type of game bot imitates the human's pressing button actions, automatically doing some repetitive operations. The modification type of game bot uses the third-party mobile application to inject a malicious module into the game and then modify the game's internal data. This type of game bot seriously breaks the fairness of the game. Finally, the crack type of game bot has the most influence on game vendors; it not only breaks the game's fairness but also accelerates a game to be on the decline. This type of game bot has entirely known about the game and can unscrupulously modify the game's code.

The ways of defending game bots are based on three directions [3, 8, 10]. The first solution is the client side's defense by using a mobile application, protecting the game's code from being modified to reduce the possibility of trying to crack the game. However, this method excessively drops users' game experience, as they feel disturbed during the play. Second, the information transmitted between a client and the server is encrypted to prevent game bots from eavesdropping. This solution must spend more computation time to encrypt/decrypt the transmitted information. The third method for defeating game bots is back-end protection by timely monitoring or offline data analysis to defend known or unknown game bots. However, the previously proposed models performed well in the selected game, but

are hard to be utilized in other games. In addition, bot developers can neutralize detection methods if they recognize detection thresholds. Once bot developers figure out patterns of bot detection, they start to generate bots to imitate human users' behaviors [2]. For securing bot detection, it is necessary to develop an improved method, particularly in the gray area that cannot be detected accurately.

2.2. Game Bot Detection Based on Behavioral Features. Behavioral actions depict how a character performs physical activities such as moving, normal attack, or using skills [2]. Lee et al. [9] analyzed the full action sequence of players on a big data platform. Their approach was to set specific behavior sequences and apply the scoring algorithm and Naïve Bayesian algorithm to classify bots from players. In another method, Lee et al. [1] captured the similarity of character behavior to classify game bots. This method was to divide behavior sequences with a time window and embedded as a feature vector. Bots were shown they have similar behavior patterns during playtime by applying the logistic regression algorithm with self-similarity of characters. The aforementioned two methods [1, 9] demonstrated significant performance but had no sustainability. If bot developers change patterns of game bots, their detection methods are easily avoidable.

A lot of machine learning algorithms were applied to game bot defense. The most frequently used algorithms included J48, DecisionStump, HoeffdingTree, RandomForest, and REPTree [5, 11, 12]. A machine learning method is to collect data from games' databases first, and then, features will be extracted [13]. These features are standardized to speed up the training process. After standardizing, all data are going to be separated into training data and testing data. And the training data will be trained by some machine learning algorithms and built a model which is used to identify whether a player is a bot. Bernardi et al. [6] used many machine learning algorithms to detect game bots through behavioral features. They were first based on descriptive statistics to see what is the difference between a human and a bot and then utilized many machine learning algorithms to find the best algorithm that can accurately detect game bots. Kang et al. [4] detected multimodal game bots through user behavioral characteristics. They also first collected data from the game's database and then extracted users' features. They used four different machine learning algorithms to evaluate many metrics, including accuracy, precision, recall, and *F*-measure. Tao et al. [10] proposed a generalized game bot detection framework for massively multiplayer online role-playing games (MMORPGs) termed NGUARD, denoting NetEase Games' Guard. NGUARD took charge of automatically differentiating game bots from humans for MMORPGs. Xu et al. [8] employed a combination of supervised and unsupervised methods for game bot detection, where supervised methods were applied to discriminating bots and humans according to labeled data, and unsupervised methods were also applied to detecting game bots and can further help discover a new type of game bots. Park et al. [14] indicated one problem with the detection methodologies is that they do not provide rational explanations about their decisions. To resolve this problem, Park et al. [14] investigated the explainabilities of the game bot detection. They developed the XAI (Explainable Artificial Intelligence) model using a dataset from the Korean MMORPG, AION, which includes game logs of human players and game bots. Table 1 summarises the related works.

The learning models mentioned above can be used to exactly detect more definite game bot players. However, some players belonging to the gray area cannot be detected accurately. Hence, developing an improved method for effectively detecting game bots is exceedingly essential.

3. The Proposed Deep Neural Networks-Based Approach

3.1. System Model. In this study, we focus on players who use bots to modify internal data. In the game bot detection, it is hard to detect gray area players because normal or abnormal players' data look like each other in the gray area and thus reduce the accuracy of the detection system. Besides, the detection system needs more and more players' data to detect game bots; otherwise, it will only increase the false detection. This suggests the importance of players' historical data in the game detection system. Little players' data cannot increase the accuracy of the detection system.

Most back-end game bot detection approaches are to use players' data, such as players' levels or their experience values. They collect and preprocess these data from the game's database. Then, these preprocessed data will be used in machine learning or data mining to build a model or define the range. If there are new players' data that need to be detected, they can use the previously defined range to judge whether is normal or not. Different from the general approach, we use the deep learning technique [15] to build the model and utilize this model to analyze the players in the famous game named KANO which is a mobile online game of playing baseball. Figure 1 illustrates the overall architecture of the proposed game bot detection.

We devise the deep learning approach to detect game bots and gray area players based on the multiplayer perceptron (MLP) neural networks [16, 17]. MLP is a class of feedforward artificial neural networks and a universal function approximator [18]. It can be used to create a mathematical model by regression analysis. MLP has broad application situations in diverse fields such as image recognition, speech recognition, and machine translation. The proposed deep neural networks-based approach utilizes a dataset of players' historical data. The raw data are preprocessed first, including data transformation, feature extraction, and standardization, to get better data formats for model learning. Then, the preprocessed data are fed into a neural network to build a model, where we must decide how many layers and neural units the model possesses. Then, we must set each neural layer's initialization function and activation function. Finally, we set the whole model's loss function, optimizer, and metrics. After setting the model, we

Table 1: Comparison of the related works.

Author	Year	Dataset	Approach
Kang et al. [3]	2013	AION	Rule set
Chung et al. [7]	2013	Yulgang online	Multiple SVM (support vector machine)
Lee et al. [1]	2016	Lineage, AION, blade & soul	Logistic regression
Tao et al. [10]	2018	NetEase MMORPGs	ABLSTM (attention-based bidirectional long short-term memory network)
Park et al. [2]	2019	AION	LSTM (long short-term memory network)
Xu et al. [8]	2020	NetEase MMORPGs	Combination of supervised and unsupervised methods
Park et al. [14]	2021	AION	XAI (explainable artificial intelligence) model

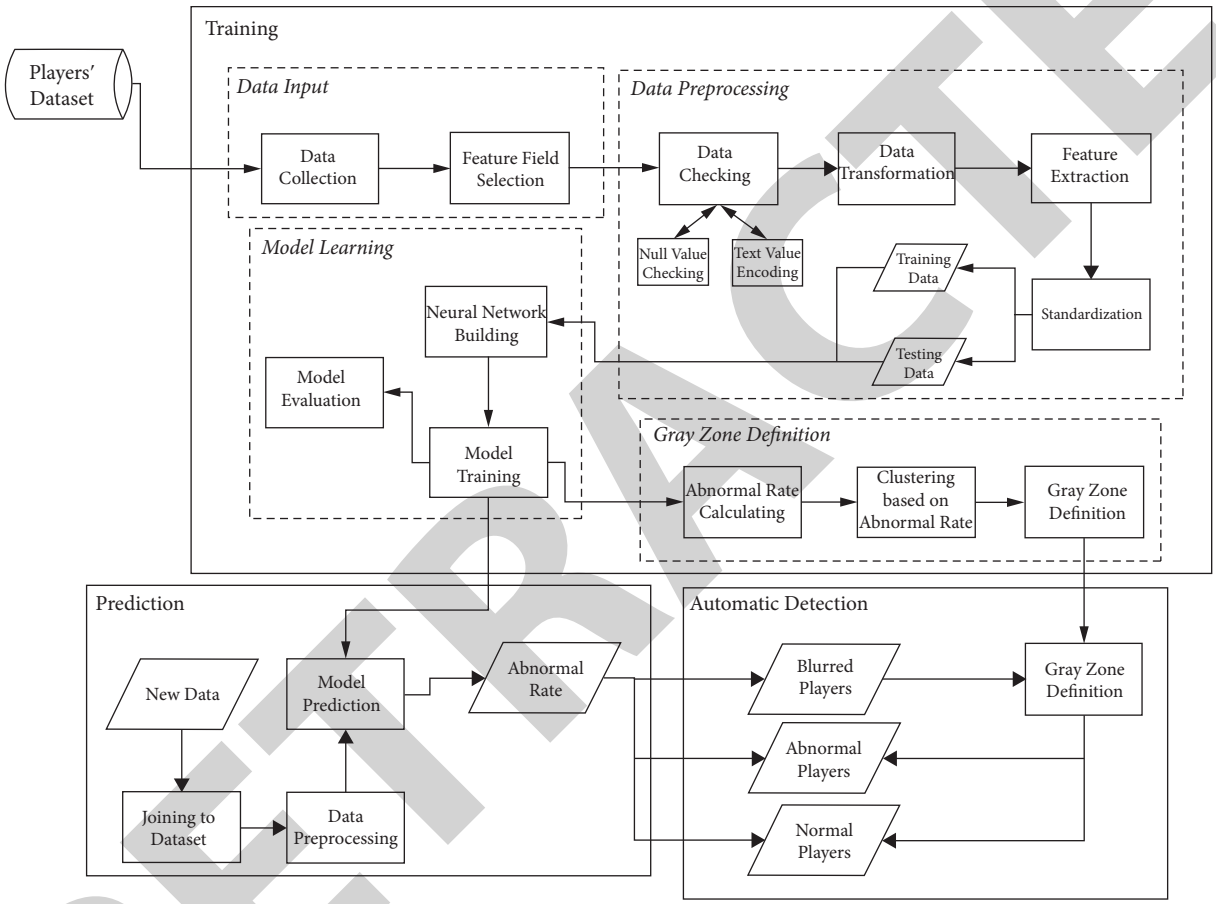

Figure 1: The system model of the proposed game bot detection.

can build the model, calculate each player's abnormal rate, cluster players based on their abnormal rate, and finally define the gray area. When there are new players whose data have been updated, the system preprocesses the raw data first and joins the players' data into the historical dataset afterward. Next, the trained model is used to predict and calculate the abnormal rate. We distinguish players into three types: normal, abnormal, and blurred player based on their abnormal rate. We present four MLP models in this study. They consist of different parameters and settings, such as kernel initialization, activation function, and epoch.

3.2. Players' Feature Extraction and Transformation. We first export players' data to a csv file named "Players.csv" from the mobile online game's database. Because we use Python to deal with the data, we choose the needed columns and transform these selected columns' data to Pandas data frame format so that we can handle the data easily. We use these fields as the training features. Table 2 shows the descriptions and names of these fields. After processing, we remove the name's column temporarily because names will not be used in the training phase and are going to be used in the prediction phase. Then, we check if there is any field being the null value and transform all data frames to arrays.

After transformation, we get a $n \times m$ matrix, which is n rows and m columns, and this is a tuple of integers indicating the size of the array in each dimension. Then, we extract the label and features from the array. Next, we are going to do standardizing because every numerical characteristic field has a different unit; for example, the unit of some players is person, and gold level's unit is level and does not have a

Table 2: Descriptions and names of player data's 9 fields.

Name	Description
Gold	The number of gold that a player holds
Gold level	The level of the number of holding gold
Player level	Player level is to depend on experience
Player experience	Player's experience
Number of players	The coach can set the number of players in the game
Player status	0: idle; 1: playing
Endurance	Endurance of a player
Fighting	Fighting of a player
Reputation	The reputation of a player

common standard. Let all values be between 0 and 1 by standardization. Standardization can raise the accuracy of the model after training. Finally, we randomly separate the processed data into training and testing data, where training and testing data are 80% and 20% of all processed data, respectively. Here, training data are used for training a model, and testing data are used for calculating the accuracy of finishing model training.

3.3. Multilayer Perceptron Neural Network Model. In this section, we design a multilayer perceptron (MLP) neural network model. MLP consists of at least one hidden layer (not including input and output layers), and it can learn not only linear function but also nonlinear function. We use backpropagation to train the model, which is a method used in an artificial neural network. Backpropagation calculates a gradient that is needed in the calculation of the weights, and it is a supervised learning method. In brief, backpropagation is just like "learning from mistakes," and it can help adjust the optimum weights.

For MLP, the multilayer network can get better convergence, but too many hidden layers just let network complexity be bigger, making more local minimum points. Cybenko [19] had proved that, in the case of a hidden layer having enough neural units, only one hidden layer can approach any continuous function. Hush and Horne [20] also presented that a neural network using two hidden layers, each layer just possessing a few neural units, can replace the neural network that uses one hidden layer and a lot of neural units. Eventually, it depends on how complex the problem is. In this study, we set the MLP as a 2-layer architecture, one input layer, two hidden layers, and one output layer [21, 22].

As for the number of neural units of the hidden layer, too few neural units may make large errors. On the contrary, although a big number of neural units can reduce deviation values, the convergence rate will slow down. Even after more than a certain number, it will increase the time of training a neural network. Taud and Mas [23] pointed out that too many neural units in the neural network generate the phenomenon of overfitting. Overfitting means overtraining; that is, the hypothesis of machine learning is too close to the training data, making the error larger when the hypothesis is going to match the testing data. Dawson and Wilby [24] presented two ways to decide the number of neural units: pruning algorithm and constructive algorithm. The pruning algorithm is a technique in machine learning. It first sets a huge number of neural units of the hidden layer to start training and then reduces the number of neural units one by one. On the contrary, a constructive algorithm sets a small number of neural units first and then adds the number of network units one by one. Because the previous processed features have 9 fields, we set the input layer as 9 neural units and concatenate the output to the hidden layer 1. The hidden layer 1 adopts a constructive algorithm and sets 20 neural units first. Finally, we find that 50 neural units possess the best effect. After we finish setting the number of neural units, we then set the initialization function. Because the essence of the training procedure of the deep learning model is to update the weight, this process requires every parameter to have its corresponding initial value, and the initialization function has a great impact on model convergence. Thus, we must set the initialization function. We set the initialization function of the hidden layer 1 as uniform distribution to initialize weight and bias:

$$f(x) = \begin{cases} \dfrac{1}{b-a} & a \le x \le b, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Here, random variable x is restricted to a finite interval $[a, b]$. The activation function's main effect is to broach the nonlinearity. In the artificial neural network, if we do not use the activation function, the linear combination of the front layer input is the output of this layer (input and output still keep a linear relationship), which makes the deep artificial neural network becomes meaningless. So, we set the activation function as rectified linear units (ReLu function):

$$f(x) = \text{ReLu}(x) = \max(0, x), \tag{2}$$

where x is the input to a neuron. Compared with traditional neural network activation functions, e.g., tanh and ReLu function has more advantages. ReLu function has more efficient gradient propagation, avoiding vanishing or exploding gradient problems.

Next is the hidden layer 2 which adopts the pruning algorithm. We first set 50 neural units and then find 40 neural units that have the best effect. And its initialization function and activation function are the same as hidden layer 1 [25]. In addition, because we are going to predict abnormal rates afterward, the output layer has 1 neural unit, and we set its activation function as the Sigmoid function:

$$S(t) = \frac{1}{1 + e^{-t}}, \tag{3}$$

where t is an independent variable and $S(t)$ is an exponential function. The output of the sigmoid function can range from 0 to 1.

When all functions have finished their set, we can build a sequential stack model. Figure 2 illustrates the overall MLP neural networks architecture. Before training, we have to define the training method. We first set the loss function which is used to estimate the model's discordant degree of prediction and real value. The smaller the loss function is, the greater the model's robustness is. Because the problem is binary classification, we set it as binary cross-entropy (log loss), using logistic regression, to get the following formula:

$$-\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log\left(\widehat{y}_i\right) + (1 - y_i)\log\left(1 - \widehat{y}_i\right)\right]. \tag{4}$$

It utilizes a known sample distribution to find the parameter values that are most likely to cause this distribution (that is maximum probability). In deep learning, it is useful to use cross-entropy to obtain a better training effect. And it is common to optimize the gradient; that is, an optimizer is used to optimize every gradient descent algorithm. We set optimizer as adaptive moment estimation (Adam) function:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\sqrt{\widehat{v}_t} + \varepsilon}}. \tag{5}$$

Adam optimizer is a first-order optimization algorithm that can replace the traditional random gradient descent process. It is based on training data iteration to update neural network weights. Using Adam optimizer, it can make training convergence quicker and raise the accuracy. The final parameter is the metric function. Metric is a norm of evaluating neural network performance and does not participate in the optimization process. The model is binary classification, so it usually sets the metric as an accuracy function.

Regarding the computational complexity of the proposed MLP model, we will describe it in the following. As shown in Figure 2, we define the number of neural units in the input layer, hidden layer 1, hidden layer 2, and output layer as N_i, N_1, N_2, and N_o, respectively. Because the major computational complexity in MLP involves the fully connected networks between the layers, the computational complexity is $N_i^* N_1 + N_1^* N_2 + N_2^* N_o$ [26].

3.4. Predicting Gray Area and Response. When we have finished training the model, we can further use this model to predict whether is a game bot. We write the previous data preprocessing method to a function so that when there is new or updated data needing to be detected, we can use this function to preprocess the raw data first. Then, we use the trained model to predict, and the prediction will calculate the abnormal rate to know whether a player is normal or not.

Although a player can be judged based on his/her abnormal rate, there is still some misjudgment in the gray area. In this study, the gray area means a normal player data looking abnormal or an abnormal player whose data looks normal. These two types of players make judgment go wrong easily, and therefore, we must devise a method to detect the gray area player for effectively identifying game bots.

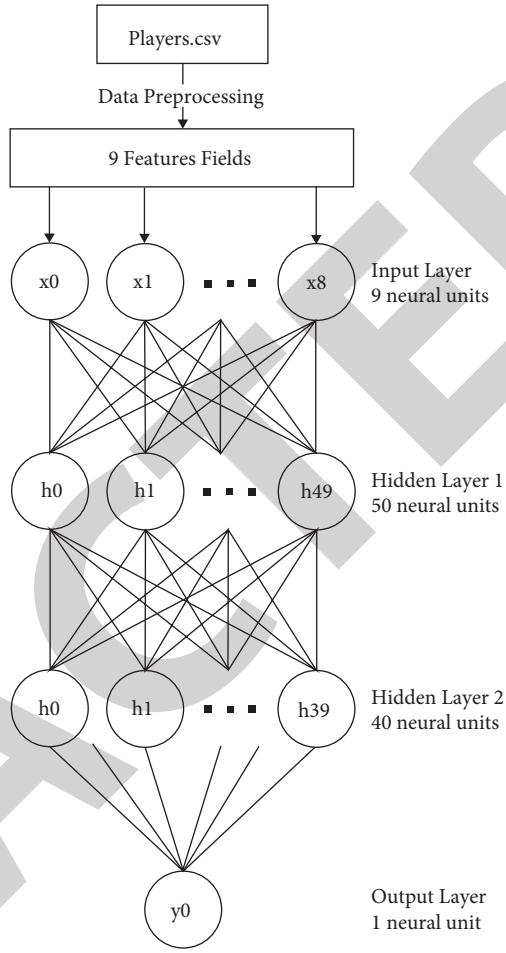

Figure 2: The proposed MLP neural networks.

We adopt the K-means clustering method [27] to cluster players based on their abnormal rates. For the abnormal rates, we want to separate them into three clusters: normal, gray area, and game bot. K means can find the best clusters with center points having minimal distances to the group members and automatically calculate the border of the cluster without manual thresholding. By K means, we can easily find out the gray area cluster between normal and game bot clusters. The rates in the gray area cluster are far away from the center points of normal and game bot clusters, so K means can help us to determine the best size of the gray area cluster and its border. The K-means clustering was presented by Hartigan and Wong [27], which is a method of vector quantization, originally from signal processing. It aims to partition n observation into K clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. Through a set of observations, the general procedure is to search for a K-partition with a locally optimal within-cluster sum of

squares by moving points from one cluster to another, and the following is the formula:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2. \tag{6}$$

After clustering, we obtain three clusters, including normal, gray area, and game bot clusters. To exactly define whether a player in the gray area is normal, we check each player's data to see if there is a misjudgment. Finally, we get a range to judge the player and then evaluate whether the proposed model is accurate and further compare with the random forest model in machine learning by using the following metrics [28, 29]: error rate, accuracy, precision, recall, $F1$ score, false-positive rate (FPR), and false-negative rate (FNR), where FN, FP, TN, and TP denote the number of false negatives, false positives, true negatives, and true positives, respectively:

$$\begin{aligned}
\text{Error rate} &= \frac{\text{FN} + \text{FP}}{\text{FN} + \text{FP} + \text{TN} + \text{TP}} \times 100\%, \\
\text{Accuracy} &= \frac{\text{TN} + \text{TP}}{\text{FN} + \text{FP} + \text{TN} + \text{TP}} \times 100\%, \\
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100\%, \\
\text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100\%, \\
F1 \text{ score} &= 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \times 100\%, \\
\text{FPR} &= \frac{\text{FP}}{\text{FP} + \text{TN}} \times 100\%, \\
\text{FNR} &= \frac{\text{FN}}{\text{FN} + \text{TP}} \times 100\%.
\end{aligned} \tag{7}$$

4. Experimental Evaluation

4.1. Dataset Description and Preprocessing. We evaluate the proposed bot detection in the mobile online KANO game's dataset. The dataset recorded the players' data from 2019 to 2021, including normal and abnormal behaviors. We extract these three years' player data which comprise about 210,000 players (130,000 normal and 80,000 abnormal players).

The dataset also contains labels that are used to mark whether a player is normal. Labels are either 0 or 1, which means a human and a bot, respectively. Because enough data can make detection more accurate, all 210,000 data are used to train the model. The data preprocessing function is used to process the raw data, and then, the proposed MLP model will be built.

4.2. Experimental Setting and Results. In this section, we present the experimental results acquired from different MLP neural network models and a widely used random forest machine learning model with the same dataset. The training model environment is on NVIDIA GTX 1050Ti GPUs with 4 GB memory.

We set training parameters to separate the dataset into training and testing data with different ratios, where different ratios will have different results. As discussed in Section 3, Table 3 describes the specifications of models used in experiments. To simplify the expression, we name each model with a different name listed in Table 3 to explain the experimental results.

We contrast among different MLP models to choose the best model by estimating the results with the same dataset. We also contrast between the MLP model and random forest algorithm. The random forest algorithm was developed by Breiman [30], which has many merits, e.g., being able to process large input variables, and is extensively used in machine learning. Accordingly, we select the random forest method as the baseline of the general machine learning model in the experiments. We evaluate the experimental results using five metrics, including the error rate, accuracy, precision, recall, and $F1$ score. In the following, we first list the comparisons among the proposed four MLP neural network models, then conduct the comparisons between the random forest machine learning and the proposed best MLP neural network from the same KANO game players' dataset, and further make comparisons between Park et al.'s method [14] and the proposed best MLP model.

4.2.1. Comparisons among the Proposed Four MLP Models. We compare the performance of four different MLP neural network models with the same amount of data in this experiment. The differences among the four models are the optimizer, split of validation, epochs, and batch size. To strengthen the accuracy or reduce the error rate, we will increase the number of epochs and use more times of training periods to improve them. However, it still needs to consider other variables, including a split of validation or batch size; if we do not consider these variables, it may increase the training time, decrease the accuracy, and therefore raise the error rate. Table 4 demonstrates the results of the performance comparisons among these four different MLP models. We find that these four MLP neural network models reach a similar performance in the experimental results. Especially, model_3 has the best accuracy, precision, recall, $F1$ score, and the most reduced error rate. The accuracy in model_3 is extremely high, 99.894%. The $F1$ score is the combination of precision and recall. Unlike accuracy, F1 does not involve TN, true normal users, which are usually more than true abnormal users. F1 in model_3 is also very high, 96.163%. Precision shows the influences of FP against TP, and recall reflects FN. Both of them in model_3 are the best, and recall is relatively higher than precision. So, model_3 can provide extremely low FP, which is more critical than FN due to the large among of normal users, and still provide very low FN to detect abnormal users.

Moreover, Figures 3 and 4 separately illustrate accuracy and loss for training iterations of model_3 in the experiment. Chollet [31] indicated that if a proposed model has a large

Table 3: Specifications of different MLP models.

Items	Model_1	Model_2	Model_3	Model_4
Layer	2	2	2	2
Neuron	50, 40	50, 40	50, 40	50, 40
Activation function	ReLu, sigmod	ReLu, sigmod	ReLu, sigmoid	ReLu, sigmoid
Loss function	Binary cross-entropy	Binary cross-entropy	Binary cross-entropy	Binary cross-entropy
Optimizer	Adam	Adadelta	Adamax	Nadam
Split of validation	0.1	0.3	0.2	0.4
Epochs	30	20	12	15
Batch_size	1000	5000	9500	7000

Table 4: Performance comparisons among different MLP models.

Metrics	Model_1	Model_2	Model_3	Model_4
Error rate	3.818%	1.382%	0.106%	1.659%
Accuracy	96.182%	98.618%	99.894%	98.341%
Precision	96.58%	97.013%	97.352%	97.116%
Recall	98.057%	97.219%	99.319%	98.127%
*F*1 score	96.123%	96.142%	96.163%	96.125%
Total training time (second)	22 s	16 s	1 s	7 s

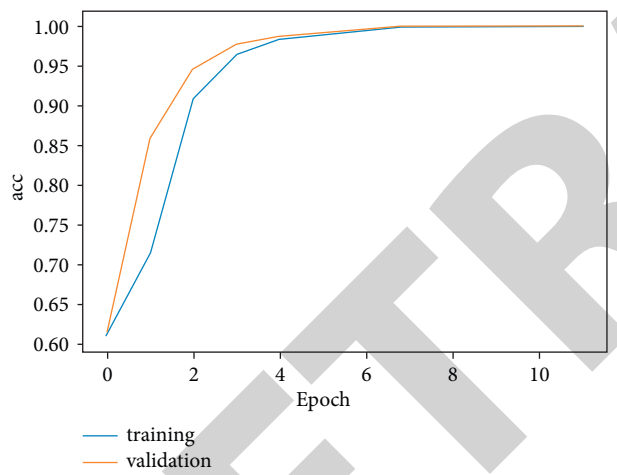

Figure 3: Accuracy for training iterations of model_3.

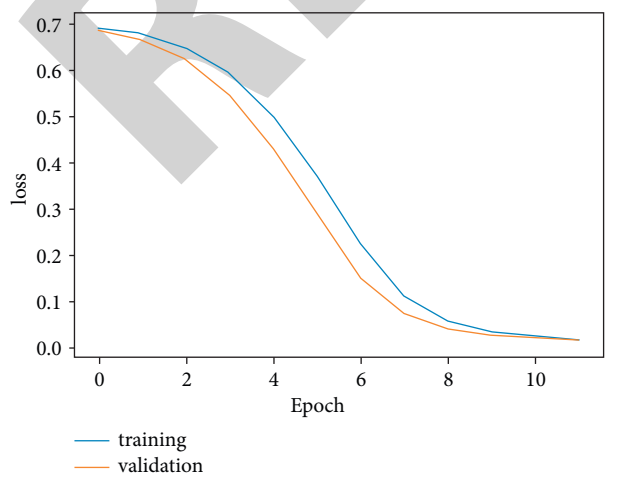

Figure 4: Loss for training iterations of model_3.

gap between its training error curve and validation error curve, it shows the proposed model may have a big problem of overfitting. In the experimental result of model_3, the gaps between the training and validation sets in accuracy and loss are both very small, which means there is not too much phenomenon of overfitting or underfitting in the training iterations.

4.2.2. Comparisons between the Random Forest Machine Learning and Proposed Best MLP Model from the Same KANO Game Players' Dataset. Here, we contrast the performance of the MLP neural network and random forest models. In the next section, we will compare the performance of detecting the gray area between the two models. Based on the above analysis, we select model_3 as the representative of various MLP neural networks and compare it with the random forest method. Table 5 shows the contrast between model_3 and the random forest model with the same dataset. The contrast result shows that the best model_3 achieves the error rate of 0.106%, accuracy of 99.894%, and *F*1 score of 99.163%, while the random forest model (with the same dataset) achieves the error rate of 4.8%, accuracy of 95.2%, and *F*1 score of 95.0%. That is, the proposed MLP neural network approach reduces the error rate from 4.8% to 0.106% and increases the accuracy from 95.2% to 99.894%.

4.2.3. Comparisons between Park et al.'s Method [14] and the Proposed Best MLP Model. In 2021, Park et al. [14] developed the random forest model and MLP model with XAI (Explainable Artificial Intelligence) modules using a dataset from the Korean MMORPG, AION, which includes game logs of human players and game bots. At first, Park et al. trained the random forest classifier and MLP classifier, respectively. The dataset contained two classes: Bot and Human, and it has been through the oversampling process due

Table 5: Performance comparisons between the MLP model_3 and random forest model.

Model name	Error rate	Accuracy	Precision	Recall	$F1$ score	Time
Model_3	0.106%	99.894%	97.352%	99.319%	96.163%	1 s
Random forest	4.8%	95.2%	95.0%	95.3%	95.0%	34 s

to the low number of data in the "Bot" class. 90% of the dataset was used to train the models, and the rest of the data were for validation. The validation accuracy of the random forest model is 88.51%, while the MLP model reached over 90.10%. Therefore, the proposed best MLP model, model_3, with an accuracy of 99.894% is superior to Park et al.'s random forest model and MLP model. Table 6 shows the accuracy comparison between Park et al.'s method and the proposed MLP model_3.

4.3. Gray Area Processing. Since model_3 has the best performance as discussed in Table 4, we use model_3 to calculate every player's abnormal rate, then cluster players based on their abnormal rate, and further define the gray area. Meanwhile, we generate data named Data_1 (20 players) and Data_2 (50 players) to detect whether the gray area is accurate or not. We also use this Data_1 and Data_2 to test the random forest method to see whether it can be accurate in detecting the gray area. In these players, each data is very close to the threshold limit value (TLV) of its field so that we can simulate players who are usually judged incorrectly in the game. We are going to use two metrics, false-positive rate (FPR) and false-negative rate (FNR), to evaluate the performance of the best MLP neural network model_3 and the random forest model. In the following, we first measure model_3 on Data_1, and the experimental result shows that FPR and FNR are both 0.0%. This means no player is misjudged, and all players are correctly detected. Then, we measure model_3 using Data_2, and the experimental result shows that FPR and FNR are 0.196% and 0.184%, respectively. On the contrary, the random forest model's FPR is 0.29% and FNR is 0.23% on Data_1 and 0.305% FPR and 0.350% FNR on Data_2. Therefore, model_3 has better detection performance, which can be more accurate to detect the gray area's players and reduce the possibilities of misjudgment.

4.4. Summary. The above experimental results show that the proposed MLP neural network model has better performance than the random forest model under the same dataset. The best MLP neural network model, model_3, increases accuracy from 95.2% to 99.894%, increases $F1$ score from 95.0% to 99.163%, and reduces error rate from 4.8% to 0.106%. In summary, our proposed model can be more accurate to detect game bots.

In terms of the gray area, the MLP neural network model can be almost fully accurate to judge the blurred players on Data_1, and it has 0.0% FPR and FNR. However, the random forest model still has a slight error; that is, it has 0.29% FPR and 0.23% FNR on Data_1. Although model_3 has a slight error on Data_2, it is a drop in the bucket as compared with the random forest model. The random forest model has a large error on judging players, and it will make players unsatisfied about it. Accordingly, model_3 is superior to the random forest model.

Table 6: Accuracy comparison between Park et al.'s method [14] and the proposed MLP model_3.

Model Name	Accuracy
Model_3	99.894%
Park et al.'s random forest model [14]	88.51%
Park et al.'s MLP model [14]	90.10%

5. Conclusions

This study proposes an effective bot detection approach for a mobile online game based on the MLP neural networks, which cannot only help enhance the performance of detecting game bots but also identify gray area players. We first collect the dataset of the mobile online game and extract its features, and then by preprocessing raw data and training the models, the proposed game bot detection approach uses four models to evaluate and further selects the best model as the baseline of the approach. The experimental results show that the proposed game bot detection model increases the accuracy from 95.2% to 99.894% and reduces the error rate from 4.8% to 0.106%, as compared with the widely used random forest algorithm in machine learning with the same players' dataset. In addition, the proposed deep neural network-based approach has better performance in detecting game bots, especially in the gray area, as compared with the widely used machine learning techniques. Furthermore, the proposed approach uses an automatic mechanism to reduce the human resources on detecting game bots and is also suitable for other games that use a back-end database to record players' data.

To extend the accomplishment of this work, we will further apply the proposed method of this work to explore the gray area in the other games. Although the other games may have different data distributions, they may have gray areas that cause false positives and negatives. This work currently utilizes the KANO game as an example to demonstrate the advantage of the proposed method. In the future, we will use the proposed method to distinguish the gray areas of the other games and check the players in the gray areas. In such a way, we will be able to show this work can also increase the game bot detection rate in the other games.

Data Availability

The data supporting this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the Ministry of Science and Technology in Taiwan, under Contract MOST 110-2218-E-305-001–MBK.

References

[1] E. Lee, J. Woo, H. Kim, A. Mohaisen, and H. K. Kim, "You are a game bot!: uncovering game bots in MMORPGs via self-similarity in the wild," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, United States, December 2016.

[2] K. H. Park, E. Lee, and H. K. Kim, "Show me your account: detecting MMORPG game bot leveraging financial analysis with LSTM," in *Proceedings of the 20th World Conference on Information Security and Applications*, Jeju Island, South Korea, August 2019.

[3] A. R. Kang, J. Woo, J. Park, and H. K. Kim, "Online game bot detection based on party-play log analysis," *Computers & Mathematics with Applications*, vol. 65, no. 9, pp. 1384–1395, 2013.

[4] A. R. Kang, S. H. Jeong, A. Mohaisen, and H. K. Kim, "Multimodal game bot detection using user behavioral characteristics," *SpringerPlus*, vol. 5, no. 1, pp. 523–619, 2016.

[5] K. T. Chen, H. Kenneth Pao, and H. C. Chang, "Game bot identification based on manifold learning," in *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, 21 October 2008.

[6] M. L. Bernardi, M. Cimitile, F. Martinelli, and F. Mercaldo, "Game bot detection in online role player game through behavioral features," in *Proceeding 12th International Conference on Software Technologies*, IEEE-Press, Buenos Aires Argenti, June 25 - 29, 2007.

[7] Y. Chung, C. Park, N. Kim et al., "Game bot detection approach based on behavior analysis and consideration of various play styles," *ETRI Journal*, vol. 35, no. 6, pp. 1058–1067, 2013.

[8] J. Xu, Y. Luo, J. Tao, C. Fan, Z. Zhao, and J. Lu, "NGUARD+: an attention-based game bot detection framework via player behavior sequences," *ACM Transactions on Knowledge Discovery from Data*, vol. 14, no. 6, p. 24, 2020.

[9] J. Lee, J. Lim, W. Cho, and H. K. Kim, "In-game action sequence analysis for game bot detection on the big data analysis platform," *Proceedings in Adaptation, Learning and Optimization*, vol. 2, pp. 403–414, 2015.

[10] J. Tao, J. Xu, L. Gong, Y. Li, C. Fan, and Z. Zhao, "NGUARD: a game bot detection framework for NetEase MMORPGs," in *Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining*, pp. 811–820, SIGKDD, London, United Kingdom, 19 July 2018.

[11] P. Torres, C. Catania, S. Garcia, and C. G. Garino, "An analysis of recurrent neural networks for botnet detection behavior," in *Proceedings of the 2016 IEEE Biennial Congress of Argentina*, 15-17 June 2016.

[12] R. Thawonmas, Y. Kashifuji, and K. T. Chen, "Detection of MMORPG bots based on behavior analysis," in *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, pp. 91–94, ACE, Yokohama, Japan, December 2008.

[13] S. Mitterhofer, C. Kruegel, E. Kirda, and C. Platzer, "Server-side bot detection in massively multiplayer online games," *IEEE Security and Privacy Magazine*, vol. 7, no. 3, pp. 29–36, 2009.

[14] E. Park, K. Park, and H. Kim, "Understand watchdogs: discover how game bot get discovered,"vol. 2, pp. 924–931, in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART)*, vol. 2, PublisherSciTePress, Vienna, Austria, 2021 Feb 6.

[15] X. Yuan, C. Li, and X. Li, "DeepDefense: identifying DDoS attack via deep learning," in *Proceedings of the 2017 IEEE International Conference on Smart Computing*, 29-31 May 2017.

[16] K. Prasetya and W. Zheng, "Artificial neural network for bot detection system in MMOGs," in *Proceedings of the 9th Annual Workshop on Network and Systems Support for Games*, 16-17 Nov. 2010.

[17] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Information Sciences*, vol. 467, pp. 312–322, 2018.

[18] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809–821, 2016.

[19] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[20] D. R. Hush and B. G. Horne, "Progress in supervised neural networks," *IEEE Signal Processing Magazine*, vol. 10, no. 1, pp. 8–39, 1993.

[21] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 683–697, 1992.

[22] W. Guo, H. Wei, T. Liu, A. Song, and K. Zhang, "An adaptive natural gradient method with adaptive step size in multilayer perceptron," in *Proceedings of the 2017 Chinese Automation Congress*, 20-22 Oct. 2017.

[23] H. Taud and J. Mas, "Multilayer perceptron (MLP)," in *Geomatic Approaches for Modeling Land Change Scenarios*-Springer, Cham, 2018.

[24] C. W. Dawson and R. L. Wilby, "Hydrological modelling using artificial neural networks," *Progress in Physical Geography: Earth and Environment*, vol. 25, no. 1, pp. 80–108, 2001.

[25] J. Y. Choi and C. H. Choi, "Sensitivity analysis of multilayer perceptron with differentiable activation functions," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 101–107, 1992.

[26] P. J. Freire, Y. Osadchuk, B. Spinnler et al., "Performance versus complexity study of neural network equalizers in coherent optical systems," *Journal of Lightwave Technology*, vol. 39, no. 19, pp. 6085–6096, 2021.

[27] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: a K-means clustering algorithm," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[28] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[29] L. Pereira and N. Nunes, "An experimental comparison of performance metrics for event detection algorithms in NILM," in *Proceedings of the 4th International NILM Workshop*, USA, June 2018.

[30] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[31] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, IEEE, Honolulu, HI, USA, 21-26 July 2017.