Detecting Malicious Social Robots with Generative Adversarial Networks

Bin Wu[1], Le Liu[1*], Zhengge Dai[3], Xiujuan Wang[2], Kangfeng Zheng[1]
[1] School of Cyberspace Security, Beijing University of Posts and Telecommunications
Beijing, CN 100876
[e-mail: binwu@bupt.edu.cn]
[2] Faculty of Information Technology, Beijing University of Technology
Beijing, CN 100124
[e-mail: xjwang@bjut.edu.cn]
[3]Telecommunication Engineering with Management, Beijing University of Posts and Telecommunications
Beijing, CN 100876
[e-mail: zhenggedai@163.com]
*Corresponding author: Le Liu
[e-mail: liulejob@foxmail.com]

Received December 21, 2018; revised March 2, 2019; revised April 8, 2019; revised May 13, 2019; accepted June 25, 2019; published November 30, 2019

Abstract

Malicious social robots, which are disseminators of malicious information on social networks, seriously affect information security and network environments. The detection of malicious social robots is a hot topic and a significant concern for researchers. A method based on classification has been widely used for social robot detection. However, this method of classification is limited by an unbalanced data set in which legitimate, negative samples outnumber malicious robots (positive samples), which leads to unsatisfactory detection results. This paper proposes the use of generative adversarial networks (GANs) to extend the unbalanced data sets before training classifiers to improve the detection of social robots. Five popular oversampling algorithms were compared in the experiments, and the effects of imbalance degree and the expansion ratio of the original data on oversampling were studied. The experimental results showed that the proposed method achieved better detection performance compared with other algorithms in terms of the F1 measure. The GAN method also performed well when the imbalance degree was smaller than 15%.

Keywords: malicious robots, social robots detection, generative adversarial networks, supervised classification, unbalanced data

This work was supported part by National Key R&D Program of China and National Natural Science Foundation of China (No. 2017YFB0802703, 61602052)

http://doi.org/10.3837/tiis.2019.11.018 **ISSN : 1976-7277**

1. Introduction

With the high-speed development of the mobile Internet, online social networking has become an indispensable part of our daily lives. That being said, a large number of social robots participate in social networks. The primary objectives of these social robots are to create the illusion that a social network very actively to influence public opinion [1], to be employed as a means of political penetration [2], and to spread malicious content. On popular social networks, these malicious social robots have had a negative impact on human users.

As the influence of social robots on social networks has grown, malicious social robots have increasingly used various social engineering methods to encourage unsuspecting users of these networks to disclose personal and sensitive information. Therefore, social robot detection has become a hot research topic in recent years. Social robot detection aims to distinguish between robots and normal humans in social networks. Since the number of robots is far less than the number of normal humans in the real world, this problem has been considered in an experimental environment. In robot classification detection, an imbalance of training data is caused by an inconsistency in the number of normal humans and robots, and the difference in the proportion of the positive and negative samples leads to a final result that lacks credibility.

Based on the above analysis, this paper proposes the use of generative adversarial networks (GANs) to address the imbalance between positive and negative samples in robot detection. By generating samples of social robots through GANs, we mediated the imbalance between the social robot and normal human samples in the original data set, and this mediation was used to improve the accuracy of social robot detection. The main work of the paper is shown in **Fig. 1**.

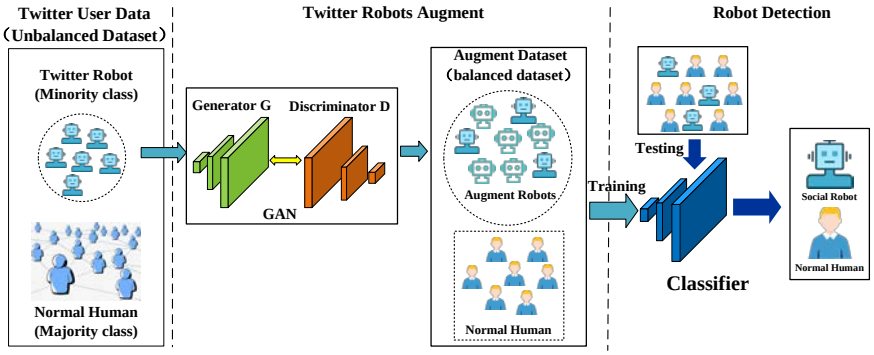


Fig. 1. We used generative adversarial networks to augment minority samples, constructed a balanced data set as the data input to train the classifier.

2. Related work

At present, popular technology involved in social robot detection is based on the dynamic content sent by social robots and the social relationship diagram around social robots. This technology requires processing data sets that are acquired ahead of schedule and then selecting some representativeness and discrimination features to achieve better classification results. Chu [3] used an entropy-based component, a machine-learning-based component,

and an account properties component to determine the likelihood that an unknown social media network user was human through the combination of features extracted from the user.

The entropy-based component detects the periodicity for the specific user. The machine-learning-based component detects spam by the content of tweets, and the account properties component detects account information. The decision maker determines whether the input account is a social robot. Varol and Ferrara [4] extracted 1150 features from public data and metadata regarding users, friends, tweet content and sentiments, network patterns, and activity time series and used random forest, AdaBoost, logistic regression, and decision tree classifiers to detect social robots. The best classifier for the area under curve (AUC) was the random forest, and the AUC for each type of feature was calculated separately. The most effective features were user metadata and content features, but some content features and emotional features were redundant. Yang and Wilson [5] used the support vector machine (SVM) classifier to make predictions from the average invitations sent over N hours using the ratio of accepted outgoing requests, the ratio of accepted incoming friend requests, and a clustering coefficient. This was the first time that Sybil graph topology was used on a major online social network. Gilani and Farahbakhsh [6] divided the collected accounts into four groups according to the number of account fans and then observed the specific relationships between their characteristics and the real identity of the accounts. These features included account age, content generation, content popularity, content consumption, account reciprocity, and tweet generation sources.

In work related to features, DARPA held a competition [7] using machine learning to detect a social robot. Among the five features of tweet syntax, tweet semantics, temporal behavior features, user profile features, and network features, six teams found the most effective combinations between features and machine learning algorithms by judging the comprehensive results of the detection. Zafarani and Liu [8] proposed a method for identifying malicious users with minimal information. This method had to classify the features of malicious users into five categories, and the detection framework generated by machine learning demonstrated strong robustness in the different algorithms and unbalanced data sets. Clark and Williams [9] maintained that the excessive dependence on user metadata could make robots with strong imitation abilities difficult to detect. Therefore, the language attributes of the tweets were used as the basis for the classification. These researchers calculated the mean and standard deviation of each dimension through user data of normal humans and then calculated the distance between the unknown user and the attribute average to classify a social robot. This method can be used to dynamically prevent a robot account from manipulating the user attributes and hiding its real identity.

Social robot detection can also be achieved without feature extraction. Wang and Konolige [10] made use of a clickstream model to detect the real identity of social accounts on the server side. These authors input the click stream sequence and then calculated the sequence distance to accurately classify social accounts. Cao and Sirivianos [11] developed a tool called the SybilRank that ranked the user's impersonation possibilities by using social graph attributes. Cai and Li [12] combined (convolutional neural networks) CNNs with (long short-term memory) LSTM model to explore the semantic information and a potential time model. This method utilized the content information and behavior information and converted the user content into temporal text data to reduce the workload for determining the features. Chavoshi [13] designed the DeBot system using an unsupervised learning approach and proposed a new hash mapping technique that could quickly group a large number of associated users. The accuracy of this method reached 94% in social robot detection. In addition, Kudugunta and Ferrara [14] judged whether the social media account was a social

robot by analyzing a single tweet. They introduced the contextual LSTM deep neural network that used content and metadata as input. This model can accurately be used to judge the category of the social media account. Additionally, these investigators proposed a method based on synthetic minority oversampling to enhance the existing data set and generated a minority sample to improve the classification performance. Beutel and Xu [15] detected lockstep page like patterns on Facebook by analyzing only the social graph between users and pages and the times at which the edges in the graph found malicious social robots.

At the same time, data augment could be applied some areas, such as computer vision, voice data augment and nature language. Charalambous and Bharath [16] proposed a method of generating synthetic video data for the data enhancement of gait sequences. This process allowed the generation of sequences using multiple confounding factors and ultimately synthesized large amounts of training and test data. Lemley and Bazrafkan [17] designed a network for data generation and a network for discriminating data. These researchers have performed many experiments using different data sets to verify that nontrivial cases, where two or more samples of a certain class were merged in nonlinear ways, resulting in the improved generalization of a target network. Antoniou and Storkey [18] proposed data augmentation generative adversarial networks (DAGAN) based on image conditions. The model fetches data from the source domain and learns to fetch any data items and generalize them to generate other within-class data items. Zafar and Ashraf [19] proposed a new image representation that combines spatial information with the bag-of-visual-words (BoVW) model. Spatial information is added by computing the global relative spatial orientation of the visual words. These researchers calculated the histogram of the visual word based on the size of these orthogonal vectors to improve the accuracy of the classification. The original input of gesture recognition are similar to picture recognition. Tran and Yin [20] proposed the disentangled representation learning-generative adversarial network (DR-GAN), the encoder-decoder structure of the generator allowed DR-GAN to learn a generative and discriminative representation. This representation is explicitly disentangled from other face variations, such as pose, through the pose code provided to the decoder and the pose estimation in the discriminator. Instead of directly manipulating the input image, Lenga and Yu [21] performed virtual sample generation at the feature level. First, the distribution of data features was estimated, and then the uniformly distributed random noise was taken as the input training sample, and finally the minority samples were generated. Some studies about data augment in the field of computer vision could also be extended to voice data. Hsu and Zhang [22] used the source and target domain data to train the variational autoencoder that learned the underlying laws of speech data, and modified the potential representation to convert attributes that are not related to recognition. This proposed method would have an absolute word error rate (WER) reduces by up to 35%. Cui and Goel [23] proposed a new data enhancement method based on random feature map (SFM) for speaker adaptive feature space. Improved recognition performance could be observed through experiments. Feature-based research methods had some similarities, so data enhancement methods could also be used in the natural language field. Fadaee and Bisazza [24] located low-frequency words by generating new sentence pairs containing rare words in the context of new synthetic creation. Experimental results simulating low resource settings show that our approach significantly improves translation quality.

All methods based on machine learning inevitably require a large number of original data sets. Existing detection methods do not explicitly solve the imbalance in the ratio of the positive and negative samples in the original data set. An imbalance between the positive and negative samples reduces the effectiveness of the final detection.

3. Social robot detection based on GANs

3.1 Social robot detection

The social robot detection problem is actually a binary classification problem. The formal definition is as follows: $A = \{a_1, a_2, \cdots, a_{|A|}\}$, which represents the collection of social accounts to be detected on the social network. $C = \{C_R, C_N\}$ is defined as the category collection, where C_R is a collection of social robot accounts, C_N is a collection of normal human accounts, and $C_R \ll C_N$. The essence of the social robot detection problem is to determine whether the account a_i belongs to the social robot collection C_N. The decision function is as follows:

$$\varphi(a_i, c_j): \mathrm{A} \times \mathrm{C} \rightarrow \{0,1\} (1 \leq \mathrm{i} \leq |\mathrm{A}|, \mathrm{j} \in \{\mathrm{R}, \mathrm{N}\}). \tag{1}$$

The result of $\varphi(a_i, c_j)$ can only be 0 or 1, which can be summarized as follows:

$$\varphi(a_i, c_j) = \begin{cases} 0, a_i \in C_R \\ 1, a_i \in C_N \end{cases}. \tag{2}$$

We generated new samples by oversampling methods, appended these new samples in C_A and mixed C_R in C_A, which made $C_A \approx C_N$. As a result, the classifier would facilitate more effective detection of social robots through a balanced data set.

3.2 Overall process

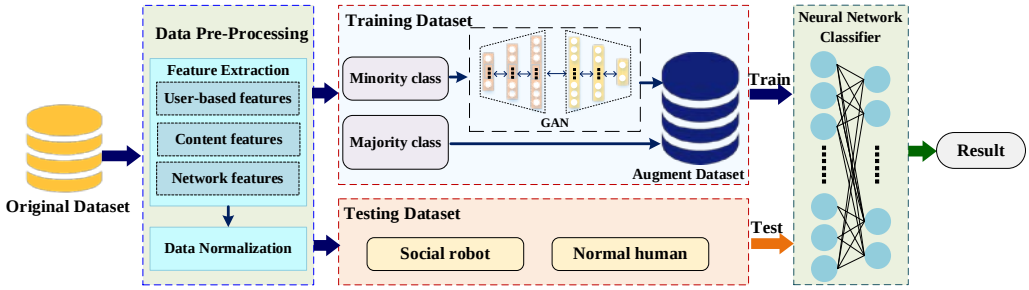


Fig. 2. Social robot detection framework

Fig. 2 shows the social robot detection framework proposed in this paper. By collecting tweet and user profile information, we generated the available data set from the original data set through feature extraction. We input the robot account from the training set into the GAN and then trained the GAN until it was stable. Then, we incorporated random noise into the stable GAN to generate fake robot accounts that were difficult to distinguish for the discriminator. Next, we mixed them with the normal human data sets to form an augmented training data set. Finally, we adopted the augmented training data set to train a classifier that could be used to classify an unlabeled account in the test set.

3.3 Feature extraction

The *i-th* social account a_i is quantified as $a_i = (a_i^1, a_i^2, \ldots, a_i^M)$. Here, a_i^j is the description of the *j-th* feature of the social account, and M is the number of features. Many available

features could be selected for the social robot detection process. We selected features with better classification abilities from the content of user tweets and behavioral features. We then used these features for social robot detection. Eleven types of features were involved:

(1) Average number of topic tags a_i^1: The average of the topic tags in the tweets. In the tweet, the "#" and other forms indicate that the tweet is highly correlated with a specific topic, and the social robot increases the speed of information dissemination by publishing tweets with multiple hot topics, so it can be defined as:

$$a_i^1 = \frac{n_t}{n_a}, \tag{3}$$

where n_t is the number of topic tags in all tweets of social account a_i and n_a is the number of tweets of the social account a_i.

(2) Average number of user mentions a_i^2: The average value mentioned by the user in the tweet. Specific users can be notified via "@username," and the social robot mentions users more frequently than normal users. The average number of user mentions can be formulated as:

$$a_i^2 = \frac{n_u}{n_a}, \tag{4}$$

and where n_u denotes the number of user mentions in all tweets of social account a_i.

(3) Number of links a_i^3: The average number of URL links in the tweet, where a_i^3 can be denoted as

$$a_i^3 = \frac{n_l}{n_a}, \tag{5}$$

and n_l is the number of URL links in all tweets of the specific social account. The tweet content in social networks supports multiple forms, including URL links, and social robots add more links to the tweet content posted to entice normal users to click and launch social engineering attacks.

(4) Number of retweets a_i^4: The ratio of the number of tweets that belong to retweets (i.e., forwards of tweets) of other users to the total number of tweets. Under normal circumstances, normal users will only retweet tweets in which they are interested, but social robots will retweet other users' tweets at a higher frequency under the control of an automated program. The number of retweets can be denoted as

$$a_i^4 = \frac{n_r}{n_a}, \tag{6}$$

where n_r is the number of the tweets that belong to retweets (i.e., forwards of tweets) of other users.

(5) Number of favorites a_i^5: The total number of users' favorites for other tweets. Normal users in social networks will express their concern and attitude towards tweet content by supporting it, but the main purpose of social robots is to increase their influence. Hence, their tweet content is favorited less frequently than that of normal human users. The final form of a_i^5 is an integer that is equal to the number of all favorites from social account a_i.

(6) Ratio of followers to the number followed a_i^6: The specific definition can be expressed as follows:

$$a_i^6 = \frac{n_e}{n_d}, \tag{7}$$

where n_e denotes the number of followers and n_d denotes the number of friends of social account a_i. Social robots focus on a large number of normal human users on social networks to improve their influence in virtual networks. However, due to the lack of realistic friends and dynamic content, their fan base is small.

(7) Tweet source a_i^7: The number of tweet sources belonging to the official source. Normal human users use a variety of different platforms to send tweets, but these platforms use the interface provided by Twitter's official platform to share tweets. The source of tweets sent by social robots is unofficial. a_i^7 is an integer that is equal to the type of tweet sources belonging to the official source.

(8) The similarity of content a_i^8: The latent semantic text content similarity of the original tweet. Latent semantic analysis (LSA) [25] extracts the "concepts" of documents and words through "vector semantic space" to analyze the potential connection between documents and words. The basic assumption of an LSA is that if multiple different words appear in the same document multiple times, the words are semantically similar. An LSA constructs a text collection matrix. The rows of this matrix represent words, the columns represent documents, the specific values of the matrix elements represent the number of times that a word appears in the document, and then the matrix is subjected to singular value decomposition (SVD). We convert the data text into the matrix A,

$$A = U\Sigma V^T \tag{8}$$

where $\Sigma = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}$ and $S = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$.The SVD can reduce the number of rows in the matrix while retaining as much column information as possible, and then the similarity of each two words can be quantified by the cosine similarity of the two row vectors a, b.

$$\varepsilon = \frac{\sum_{j=1}^{n}(a_j \times b_j)}{\sqrt{\sum_{j=1}^{n}(a_j)^2} \times \sqrt{\sum_{j=1}^{n}(b_j)^2}} \tag{9}$$

where $a = (a_1 \cdots a_n)$ and $b = (b_1 \cdots b_n)$. The closer ε is to one, the more similar the two words are. The closer ε is to zero, the more dissimilar the descriptions are. a_i^8 is the mean of all ε. The semantic similarity of original tweets from a normal user is higher than that of original tweets from social robots. Furthermore, the interests of normal human users are relatively stable, but social robots often need to have considerable interest to expand their influence.

(9) The similarity of the tweet length a_i^9: The variance in the number of tweet words. The length of the tweet sent by the social robot is steady, while that of the normal human user varies greatly, so it can be defined as follows:

$$a_i^9 = \frac{\Sigma(x_j - \mu_9)^2}{n_a} \tag{10}$$

$$\mu_9 = \frac{\sum x_j}{n_a} \tag{11}$$

where μ_9 denote the mean of the similarity of the tweet length, and x_j is the number of the tweet length about the *j-th* tweet.

(10) The similarity of punctuation usage a_i^{10}: The variance in the number of punctuation marks in the original tweet. Normal human users have distinct punctuation usage habits, while the diversity of social robot tweet sources results in punctuation without a fixed style.

$$a_i^{10} = \sum_{j=1}^{n} p_j \tag{12}$$

where p_j is the variance of the number of occurrences of a particular punctuation.

(11) The similarity of stop words a_i^{11}: The variance in the number of stop words in the original tweet. The stop word is the most frequent word in the tweet, and it represents the writing style of the tweet sender. Normal human users employ more consistent usage of stop words than do social robots. With the same definition as a_i^9, this variable a_i^{11} indicates the degree of change in the number of stop words.

3.4 GAN

GAN is an oversampling approach that was originally proposed in 2014. In terms of deep learning, the GAN method can generate more realistic pictures to make the deep neural network model develop in the desired direction. The GAN method can also automatically learn the potential distribution laws in the original image samples. Unlike the method of machine learning that defines the model in advance, a GAN can obtain the model that conforms to the data set distribution through iterative learning.

Due to its characteristics, classical GANs were combined in this paper to generate social robot samples and ultimately improved the imbalance between positive and negative samples in the original data set. **Fig. 3** shows the main structure of a GAN.

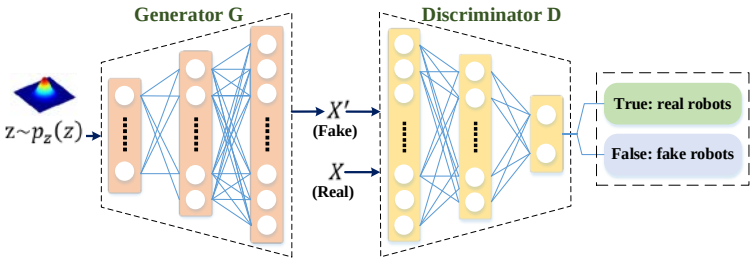


Fig. 3. The trained generator *G* produces a new sample by using random noise z, and its output is merged with the original training set. The discriminator *D* cannot distinguish the source of the input samples.

The ultimate goal of a GAN was to learn the correct distribution p_{data} of the social robot in the training data C_R. To learn the generator's distribution p_g over data x, random noise z conforming to $p_z(z)$ was taken as input of the generator *G*, and we defined a neural network $G(z;\theta_g)$ to map, where θ_g was a parameter of the neural network. We also defined another neural network, $D(x;\theta_d)$, and the output *D(x)* of this neural network represented the probability that x came from the data rather than p_g. We trained the neural

network of the generator G to optimize θ_g by minimizing $log(1 - D(G(z)))$, while θ_d remained unchanged. The data set mixed between G(z) and x was used as the input of discriminator D to train the discriminator to optimize θ_d while leaving θ_g unchanged. After some intervals of training, both G and D attempted to optimize their network parameters to form a competitive confrontation until the two sides reached a dynamic balance that was represented as $p_g = p_{\text{data}}$. Finally, discriminator D could not accurately determine the source of the social robot samples, which indicated that generator G could generate new samples that matched the distribution p_{data} of the social robots in the training data C_R as much as possible. That is, the optimization process of G and D was a binary minimax problem.

The optimization problem consisted of two parts. The first part was that the discriminator D judged whether some samples were from x or $G(z)$ so that $E_{x\sim p_{\text{data}}(x)} \log(D(x))$. Maximizing this part was the equivalent of enabling discriminator D to output $D(x) = 1$ when x conformed to p_{data}. Another part of the problem was that generator G tried to deceive discriminator D, so that there was $E_{z\sim p_{z(z)}}\left[log\left(1 - D\left(G(z)\right)\right)\right]$. The objective function [26] could be expressed as follows:

$$\min_G \max_D V(D,G) = E_{x-p_{\text{data}(x)}}[\log D(x)] + E_{z\sim p_{z(z)}}\left[\log\left(1 - D\left(G(z)\right)\right)\right]. \quad (13)$$

The essence of the generator and discriminator was the neural network, and the loss function was their essential part. The loss function of the generator and the discriminator in this paper was a logarithmic loss function that increased the sparse classification. It was defined as follows:

$$L\left(Y, P(Y|X)\right) = -log\, P(Y|X) \quad (14)$$

The goal of the G and D interval training was to update the parameters of the two networks to make them closer to each other or to reach the optimal value, thus minimizing the damage function.

3.5 Detection process

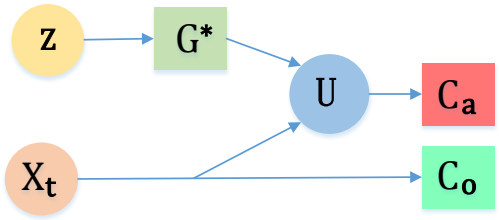


Fig. 4. The trained generator G^* is fed with random noise z, and its output is merged with the original training set X_t. The different classifiers are trained on the augmented dataset (C_a) and the original training set (C_o).

We referred to the flow diagram proposed by Fiore [27] and combined it with our work. The framework of the entire detection process was shown in **Fig. 4**, and it was divided into the following steps:

Step 1: The original data were divided into a training set T and a test set S, and the neural network classifier C_o was trained with the training set T.
Step 2: The data set F, which consisted of all the minority samples in the training set T, was used as a training set, and the continual training adjusted the hyperparameters of the GAN.
Step 3: The trained stable generator G^* could transform the input random noise z into a minority class sample set F', which was difficult to distinguish from the discriminator D.
Step 4: We mixed F' with the training set T and trained the neural network classifier C_a under the same parameters. We then compared the performance of C_o and C_a with that of the test set.

4. Experiments and analyses

4.1 Experimental preparation

The experiment used 1971 normal human accounts and 462 social robot accounts as original samples, of which 891 normal users were from the data set used in [4], and the 1080 normal users and all the social robot users were from the data set used in [28]. Social robot accounts made up 18% of the total original samples. All the data that were original tweet content crawled from the Twitter website were converted into a raw data set that could be used directly through feature extraction. The first part of the data was the normal user ID crawled in 2014; the official Twitter API was used to crawl all the relevant content. The second part of the data was the Twitter account crawled by the Twitter API in 2015 as well as related content. The overall data distribution was shown in **Table 1**.

Table 1. Data set composition

Category	Number of accounts
Normal user	1971
Social bot	462
Total	2433

Traditional evaluation metrics for the two classification problems were adopted, namely, accuracy rate, accuracy, recall, and F-measure. In a two-category problem, there were four cases in the final test results. If an instance was a positive class and was predicted to be a positive class, then it was considered to be a true positive (*TP*). If the instance was a negative class sample and was predicted to be a positive class, then it was called a false positive class (*FP*). Correspondingly, if the instance was a negative class and was predicted to be a negative class, then it was called a true negative (*TN*). If a positive class instance was predicted to be a negative class, then it was a false negative (*FN*). This definition was as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{15}$$

$$Precision = \frac{TP}{TP+FP} \tag{16}$$

$$Recall = \frac{TP}{TP+FN} \tag{17}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{18}$$

4.2 Compared Algorithms

To analyse the effectiveness of the GAN method in detecting social robots, we used four types of synthetic minority over-sampling technique (SMOTE) algorithms and an adaptive synthetic (ADASYN) algorithm as the comparison group. These five algorithms were very common in the field of data augmentation, and the improvement of the effect on the classifier was obvious.

（1）SMOTE [29]: The SMOTE algorithm randomly selected positions on the line of two minority class samples as a new minority class sample. This method improved the accuracy of classifiers for minority classes by increasing the number of minority classes. The SMOTE provided more relevant minority samples for learning so that the classifier could accurately learn the differences between different types of samples and correctly identify more samples.

（2）SMOTE-Borderline1 [30]: The SMOTE used all the minority samples to generate new samples. However, some samples located on the borderline were more likely to be misclassified than other samples away from borderline. This method generated new samples through dangerous samples that at least half of the nearest neighbour samples were from the same category.

（3）SMOTE-Borderline2 [30]: SMOTE-Borderline2 was not limited to the selection of neighbour samples for source samples, which was different from the SMOTE-Borderline1 method. The minority samples generated by the SMOTE-Borderline2 method could make the classifier better distinguish between the minority class and the majority class.

（4）SMOTE-SVM [31] [32]: The SMOTE-SVM algorithm was an efficient active learning method that generated more samples belonging to a minority class. Active learning with early stopping achieved a satisfactory solution without sacrificing classification performance. The SMOTE-SVM provided an efficient SVM-based active learning selection strategy that queried a small part of the data set at each step instead of querying the entire data set.

（5）ADASYN [33]: The ADASYN algorithm could adaptively generate bias by reducing the data imbalance for synthetic data samples of the minority classes. At the same time, the ADASYN algorithm could be extended to handle imbalances in different scenarios, and this algorithm was equally applicable to various categories of data imbalance problems.

The implementation of these five algorithms was provided by the scikit-learn library [34]. We executed these five algorithms by calling the appropriate modules in the library. All of the above algorithms needed to use the training data set as data input, and these five algorithms contained some parameters that achieved the best performance. The use of these five oversampling algorithms was similar to the GAN in the experiment. The five oversampling algorithms used the training set as input data to generate more samples. We used the augmented data set to train the neural network classifier and used the test set to verify the performance of the classifier.

These five algorithms needed to set some parameters to ensure that the generated data could conform to the laws of minority samples. We used the augmented data as input to train the classifier and choose the important parameters based on the performance of the classifier. The random state was a random number seed that guaranteed the same random sequence

during different oversampling processes. The value tested for the random state were in the range from 5 to 20. After the experimental comparison, the random state of the five oversampling algorithms was set to 10. In addition, the number of nearest neighbours used to construct synthetic samples was set to 5 empirically. For the SMOTE-Borderline1 algorithm and the SMOTE-Borderline2 algorithm, the number of nearest neighbours used to determine if a minority sample was in danger was set to 10 for optimization reason. The ratio represented the number ratio of the minority class to the majority class in the augmented data set, and this parameter was dynamically set according to subsequent experiments.

4.3 Experimental process and results analyses

For the original data set, we randomly allocated two thirds of the data as the training set and one third of the data as the test set of the positive and negative samples, respectively.

4.3.1 Parameter selection

The hyperparameters in the neural network affected the performance of the entire network, making it necessary to continuously adjust the hyperparameters for the neural network classifier until the classifier performance was optimal.

Too few layers of the neural network could cause the network to fail to satisfactorily learn the features of the data. Too many layers, however, may result in the phenomenon of overfitting. In many experiments, networks with two, three, and four layers were tested in the generator. Both the discriminator and classifier were used to find the best number of layers. One of the hyperparameters was the type of activation function in each layer. Some activation functions included the logistic sigmoid, which was defined as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{19}$$

where the ReLU function was defined as

$$f(x) = max(0, x). \tag{20}$$

In addition, the tanh function was defined as follows:

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}. \tag{21}$$

The output layer result of the discriminator was the input sample. Because the sigmoid function mapped any range to (0, 1), the activation function selected the sigmoid function.
When the input entered the neuron, it was multiplied by the weight, and we randomly initialized the weights and updated them during the model training. In addition to the weights, another linear component applied to the input was called the bias. It was added to the result of multiplying the weight by the input. For the initialization of the weights and biases, weight values were drawn from the normal distribution *N(0,1)* and biases were drawn from the uniform distribution *U(–1,1)*. The neural network generally consisted of input layer, hidden layer and output layer. Since the data dimension and the amount of data used in the experiment were small, we chose three layers network as the default implementation of the neural network.

Different optimization algorithms had different principles and were applicable to different scenarios. After many investigations, we selected the SGD and Adam algorithms as

the optimizers for testing the optimization algorithms. As one of the hyperparameters, the learning rate was the speed at which the neural network reached the optimal status. If the learning rate was too small, then the loss of the network would decline very slowly. If the learning rate was too large, then the range of the parameter updated would be too large, resulting in the network convergence to local optimum. We selected the values experimentally and observed that the performance was less dramatic with the change in hyperparameters.

4.3.2 Parameter setting

The optimal performance of GAN was that the accuracy of the discriminator D should be close to 50% as far as possible. There are several hyperparameters in GAN network which would affect the performance of GAN, including network parameters about generator G, discriminator D and the parameters about random noise. So we found the optimal combination of parameters by the grid search method. According to the experience about neural network parameter setting and GAN training, the structure of the discriminator D and the generator G was set to three layers. The dimensions of the input data and output data were determined by our features, so the neuron numbers about the first layer and the third layer of the discriminator D was fixed. For the same reason the neuron numbers about the third layer of the generator G was fixed. The learning rate of the generator and discriminator should be selected. The dimension of the noise, denoted as z_d, ranged from 1 to 3. The g_1 that ranged from 4 to 7 was the number of neurons in the first layer of generator, and g_2 was the number of neurons in the second layer of generator G, its range was from 8 to 11. The number of neurons in first layer of the discriminator D should be equal to the dimension of the input. The number of neurons in the second layer of the discriminator D, denoted as d_2, ranged from 4 to 9. Learning rate in a logarithmic grid $(5 \times 10^{-4}, 3 \times 10^{-4}, 1 \times 10^{-4})$ were experimented, g_{lr} was the learning rate of the generator G and d_{lr} was the learning rate of the discriminator D. In the training process of GAN, the discriminator D could be trained multiple times, denoted as d_t, and the generator G trained once in order to achieve balance faster. d_t ranged from 1 to 3. The stable condition of discriminator D was that the discriminator accuracy was maintained stable over 500 epochs. t_g was the training time of the generator G when GAN reached stable. t_d was the training time of the discriminator D and d_{acc} was the accuracy when GAN reached stable state. So t was the running time of GAN. All types of hyperparameters combinations exceeded 8,000, so the top 20 results with the accuracy rate closest to 50% were showed in **Table 2**. The hyperparameters combination with an accuracy of 49.04% was selected as the optimal result for training the GAN to generate an augment robot dataset.

Table 2. After the GAN reaches stability, the accuracy, training time and testing time of the GAN changed in different hyperparameters combinations.

Hyperparameters of GAN							**Evaluation metrics**			
z_d	g_1	g_2	g_{lr}	d_2	d_{lr}	d_t	d_{acc}	t_d(s)	t_g(s)	t(s)
2	4	10	0.0005	7	0.0003	3	64.70%	33	12	45
2	6	10	0.0001	6	0.0005	3	63.60%	95	31	126
3	6	9	0.0003	6	0.0005	1	62.64%	21	19	40
2	4	10	0.0003	5	0.0001	1	62.50%	28	25	53
1	5	10	0.0005	3	0.0003	1	62.36%	26	29	55
3	5	9	0.0005	6	0.0001	1	61.26%	94	92	186
1	4	8	0.0003	4	0.0001	1	60.85%	32	11	43

2	4	10	0.0005	7	0.0003	1	60.71%	43	35	78
2	4	10	0.0005	3	0.0001	1	59.62%	31	30	61
1	4	9	0.0003	5	0.0001	1	59.34%	98	87	185
2	6	9	0.0003	7	0.0001	2	57.28%	124	59	183
2	4	9	0.0005	3	0.0001	1	52.75%	28	25	53
2	4	11	0.0001	6	0.0003	1	51.79%	36	22	58
3	**4**	**9**	**0.0001**	**3**	**0.0001**	**1**	**49.04%**	**27**	**19**	**46**
3	4	10	0.0003	5	0.0003	1	42.58%	74	59	133
2	4	10	0.0003	3	0.0003	2	42.45%	62	30	92
3	7	9	0.0001	3	0.0001	1	41.48%	51	53	104
3	4	10	0.0001	3	0.0003	1	41.35%	34	34	68
2	4	9	0.0003	7	0.0001	1	41.07%	22	12	34
3	4	9	0.0003	3	0.0005	1	40.66%	22	16	38

As shown in **Table 2**, the optimal performance of GAN was determined by the hyperparameters combination. The relationship between optimal performance of GAN and running time for GAN was weak. The GAN running time represented as t in **Table 2** lasted from tens of seconds to more than one hundred seconds which is reasonable in the GAN training process, because the performance of the GAN also changed greatly when the hyperparameters combination changed slightly due to the random generation of the input noise for the generator G. The time of the training generator G and the time of training discriminator D had certain relationship with the respective network parameters. When d_t was larger, the proportion of the training time for discriminator D in the total time was larger. The increase of d_t led to reduction in the training epoch, but the increase in the time of training discriminators resulted in increase in the time of one epoch.

After a lot of experiments for the most optimal hyperparameters combination, the random noise *z* conformed to the random distribution *N(0,10)*. The dimension of random noise was setting as 3. The generator *G* in the GAN had three layers, which contained 3 ReLU units, 4 ReLU units, and 9 tanh units. The discriminator also had three layers composed of 11 ReLU units, 3 ReLU units, and 2 Sigmoid units, respectively. The learning rate of the generator was 1×10^{-4} and discriminator was selected as 1×10^{-4}. In addition, the optimizer of the generator was the Adam algorithm, and the optimizer of the discriminator was the SGD algorithm. The loss function of the generator, classifier, and discriminator in this paper was a logarithmic loss function that increased the sparse classification. The change in loss under the optimal hyperparameters combinations is presented in **Fig. 5**.

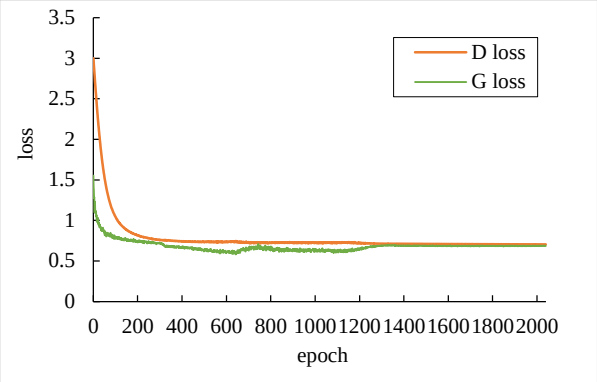


Fig. 5. The loss of generator *G* and discriminator *D* fluctuated with a change in the training epoch.

At the beginning of the training, the loss of generator G and discriminator D changed dramatically, because the generator that has not finished multiple epochs of training did not convert noise into some samples that conformed to the laws of social robots. After four hundred epochs, the loss of the GAN method tended to become flat, indicating that the samples generated by the current generator have been able to conform the data laws of real robots. Whereas **Fig. 5** indicated that the loss of generator G and the loss of discriminator D were synchronous at most epochs. After the 1400th epoch, the loss of the generator and discriminator remained stable, and the entire network reached equilibrium. The loss of generator G changed more rapidly than does the loss of discriminator D, which indicated that the initial statement of generator G may be far from the final state.

The accuracy of discriminator D with epoch changed was shown in **Fig. 6**.

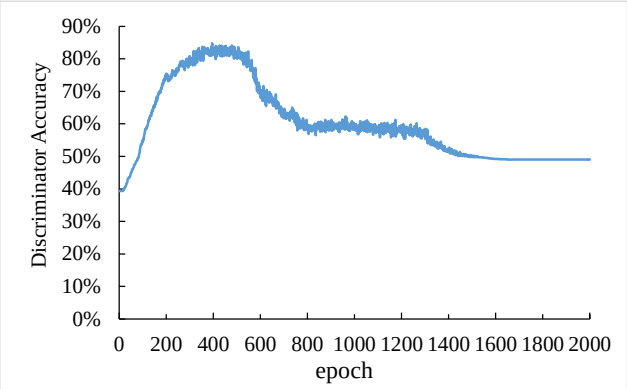


Fig. 6. The accuracy of discriminator *D* fluctuated, and finally stabilized with the change in the training epoch.

From this figure, we could see that the tendency of the accuracy was almost the same as that of the loss of the discriminator D. It was well-known that the optimal accuracy of discriminator D was fifty percent, but it was difficult to reach this value because the classical GAN had some disadvantages. After many attempts, we selected specific network parameters in order to make the accuracy of the discriminator close to 50%. Accuracy was the correct rate that the discriminator D discriminated whether the source of the sample was the generator or the real environment. Therefore, when the loss of the discriminator slightly reduced, the accuracy of the discriminator increased accordingly. Under the selected combination of hyperparameters, the generator G and the discriminator D were alternately trained, and the accuracy of the discriminator changed gradually from high to low. The discriminator D could not accurately determine the source of the input sample, so the accuracy of the discriminator was low during the initial epoch. The accuracy of the discriminator gradually increased with the continuous training. The sample generated by the generator G gradually conformed to the distribution of the social robot samples, and the accuracy rate gradually decreased. As the discriminator D continuously affected the generator G, the generator G gradually adjusted the internal parameters of the network, and generated samples that the discriminator could not distinguish, so the accuracy of the discriminator D was reduced, and finally stabilized.

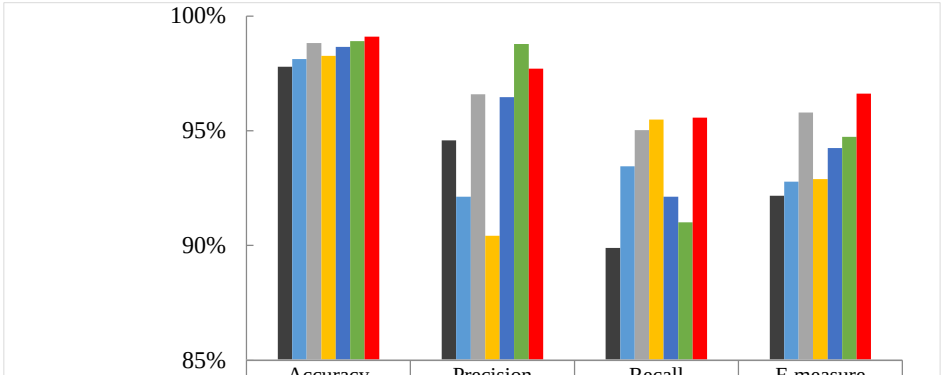
The classifier had a three-layer network composed of 20 ReLU units, 50 ReLU units, and 2 Softmax units. A simplified Nesterov momentum was also adopted, where momentum was equal to 0.5. The selected learning rate of the classifier was 5×10^{-2}, and the SGD algorithm was found to be the optimal algorithm.

4.3.3 Experimental results and analysis

To analyze the effectiveness of the GAN method in detecting social robots, we performed 3 experiments with five other traditional oversampling algorithms. At present, the most popular method of data oversampling was synthetic minority over-sampling technique (SMOTE). Therefore, we used four types of SMOTE algorithms and an adaptive synthetic (ADASYN) algorithm as the comparison group experiment. We used the same dataset and different oversampling methods to generate a balanced dataset. The same hyperparameters and balanced datasets from different sources were used in the training process of the classifier, and the performance of the classifier was detected through the same test set. The experiments were conducted with the aim of answering the following questions:

(1) Does the GAN oversampling method improve the accuracy of robot detection?
(2) Is the GAN oversampling method more advantageous than other oversampling methods?
(3) How many samples are generated for the most obvious improvement in robot detection?
(4) When the degree of data imbalance in the original data set was different, what was the difference in the degree of improvement of the detection effect by different oversampling methods?

Experiment 1: To verify the effectiveness of the GAN method in generating minority samples, five traditional oversampling algorithms were compared. The ADASYN, GAN, and four different types of SMOTE algorithms were considered in the experiment. Minority class samples were generated with the oversampling algorithms until the quantity of positive and negative samples was the same, and then the neural network was adopted to detect the social bots. The detection results are shown in **Fig. 7**.



	Accuracy	Precision	Recall	F-measure
Original	0.9779	0.9458	0.8989	0.9217
SMOTE-Regular	0.9812	0.9213	0.9345	0.9278
SMOTE-Borderline1	0.9882	0.9659	0.9503	0.958
SMOTE-Borderline2	0.9826	0.9043	0.9549	0.9289
SMOTE-SVM	0.9866	0.9647	0.9213	0.9425
ADASYN	0.989	0.9878	0.9101	0.9474
GAN	0.991	0.977	0.9557	0.9662

Fig. 7. We took as many small samples in the class as large samples in the class. The variation trend of the four evaluation indexes with the change in oversampling mode is shown.

The oversampling method based on a GAN model worked best in six of the oversampling methods in **Fig. 7**. Detailed detection based on a GAN achieved 96% in terms of the F-measure, which was 7 percentage points higher than that of the original data set. The accuracy of all classifiers was relatively close, and the accuracy of the classifier generated by GAN was one percent higher than that of the classifier generated by the original dataset. The recall of GAN methods, SMOTE-borderline1 and SMOTE-borderline2 were much higher than other methods, while GAN was better than the other two methods. High recall indicated that the classifier generated by the GAN could detect more social robots than others. The GAN method exhibited obvious advantages in the scenario with high accuracy requirement and high recall requirement. The result that GAN was superior to other oversampling methods was because GAN could learn the spatial distribution characteristics of social robots in the process of iterative training. The SMOTE algorithms randomly selected a new minority sample from the connection line between the nearest neighbor samples and a specific sample. ADASYN automatically determined how many minority samples are synthesized, rather than synthesizing the same number of samples for each minority sample like SMOTE.

Experiment 2: We had to determine how many generated examples were suitable for training the classifier. For this purpose, the expansion multiple was used. It can be defined as follows:

$$\varphi = a{:}\, b \tag{22}$$

where b was the number of social robots in the test data, and a was the number of samples generated by the oversampling method.

The main purpose of this experiment was to examine the sensitivity of the six oversampling methods in relation to the expansion multiple. In this experiment, the number of minority samples accounted for 12% of the total number of samples. Minority samples were generated with five different φ, i.e., 1:1, 2:1, 4:1, 6:1, and 8:1. Thus, five different training sets were generated, the classification of the data was performed, and the results were shown in **Table 3**. Observing the accuracy trend graph of the classifier, as φ changed, the accuracy of the classifier generated by SMOTE-Borderline2 fluctuated significantly. The accuracy of the SMOTE-Borderline2 algorithm and the SMOTE-Regular algorithm were always lower than that of the original data set. The accuracy of the GAN methods and the SMOTE-Regular algorithm was stable, and their fluctuations were gentle, which proved that the GAN method and the SMOTE-SVM algorithm were less affected by the changes in the sampling proportion.

The precision rate of the classifiers generated by the SMOTE-Borderline2 algorithm was worse than that of other methods, which indicated that the social robot samples generated by this method did not fully conform to the inherent laws of existing robot samples. The precision of the classifier generated by the SMOTE-SVM algorithm and the GAN method were stable, and it remained at 96%. Among the six oversampling methods, the recall rate of the SMOTE-Borderline2 algorithm fluctuated greatly. As φ changed, the recall of the classifier generated by the two oversampling methods of the GAN and SMOTE-Regular algorithm were always superior to the classifier produced by the original data set. The F-Measure represented the weighted average of precision and recall. In the classifiers generated by the six oversampling methods, the effect of the SMOTE-Regular algorithm and the GAN method were more stable as φ changed, and the GAN was the oversampling method that was least affected by the sampling proportion.

Table 3. The trend diagram of the evaluation indexes pertaining to the classifier generated by different oversampling methods with changes in φ.

φ	Category (%)	Original	SMOTE -Regular	SMOTE -Borderline1	SMOTE -Borderline2	SMOTE -SVM	ADA SYN	GAN
1:1	Accuracy	98.79	98.66	98.93	98.39	98.79	98.93	**99.06**
	Precision	100.0	93.41	96.55	95.29	95.45	95.51	**98.81**
	Recall	89.89	**95.51**	94.38	91.01	94.38	**95.51**	93.26
	F-measure	94.68	94.45	95.45	93.10	94.91	95.51	**95.95**
2:1	Accuracy	98.79	98.26	**98.93**	97.32	98.12	98.53	**98.93**
	Precision	100.0	93.18	96.55	92.59	96.30	**100.0**	95.51
	Recall	89.89	92.13	94.38	84.27	87.64	87.64	**95.51**
	F-measure	94.68	92.65	95.45	88.23	91.77	93.41	**95.51**
4:1	Accuracy	98.79	98.66	98.39	97.32	**98.93**	98.39	98.79
	Precision	100.0	97.59	95.29	82.86	95.51	91.40	**97.62**
	Recall	89.89	91.01	91.01	**97.75**	95.51	95.51	92.13
	F-measure	94.68	94.19	93.10	89.69	**95.51**	93.41	94.80
6:1	Accuracy	98.79	98.66	98.26	98.53	99.06	98.39	**99.20**
	Precision	100.0	97.59	98.72	91.49	96.59	98.73	**98.82**
	Recall	89.89	91.01	86.52	**96.63**	95.51	87.64	94.38
	F-measure	94.68	94.19	92.22	93.99	96.05	92.86	**96.55**
8:1	Accuracy	98.79	98.53	98.66	97.86	98.79	**99.20**	98.79
	Precision	100.0	94.32	92.47	87.63	**100.0**	97.70	94.44
	Recall	89.89	93.26	**96.63**	95.51	89.89	95.51	95.51
	F-measure	94.68	93.79	94.50	91.40	94.68	**96.59**	94.97

Experiment 3: Considering the lack of proportion between minority and majority samples in the original data, we extended the number of the minority samples to be equal to the number of the majority samples in order to influence the performance of the oversampling methods. Therefore, this experiment was designed to check the sensitivity of the six methods relative to the imbalance degree δ, which was defined as:

$$\delta = \frac{x_r}{x}, \tag{23}$$

where x was the number of accounts in the original data set, and x_r is the number of social robots in the original data set.

We let δ take the values of 6%, 9%, 12%, 15%, and 18%, individually. In each case, specific numbers of minority samples were sampled randomly and then placed into data set P together with all the normal samples. Thus, we obtained five different data sets with different δ values. Then, P was oversampled using the six methods previously mentioned, and classifications were performed. The experimental results were shown in **Fig. 8**. As δ increased, the accuracy of all the classifiers generated by all the oversampling methods gradually decreased. This showed that with a gradual increase of δ, the effect of the oversampling methods gradually weakened. When δ rose to 18%, the classification accuracy (factoring in the original data set) was optimal because the difference between the generated minority sample and the original sample had gradually increased. As δ increased, the accuracy of the SMOTE-Borderline2 algorithm dropped more quickly. Before δ was more than 9%, the GAN oversampling method was shown to perform better than all the other methods, and the detection effect of the classifiers produced by all the oversampling methods decreased slightly with the rise in the imbalance.

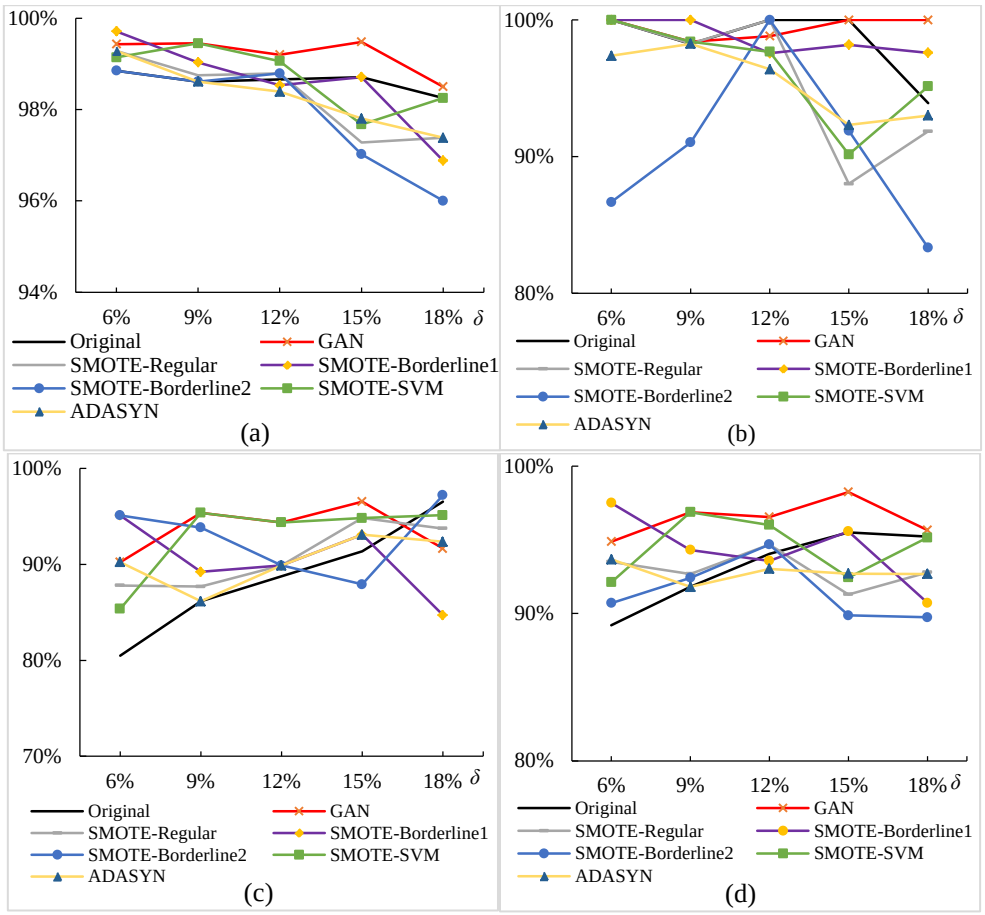


Fig. 8. The trend diagram of the evaluation indexes pertaining to the classifier generated by different oversampling methods with changes in δ: (a) accuracy, (b) precision, (c) recall, and (d) F-measure.

5. Conclusion

To resolve the problem of the imbalance between normal users and social robot accounts in terms of social robot detection, we proposed a new oversampling method that could be used to generate a minority sample through GANs. We also introduced specific processes for how to use the generated data set to detect social robots. First, we defined three different variables, and then analyzed the effects of these three variables on the experimental results through a large number of experiments. We verified that the GAN is a more effective method of data oversampling. The experimental results showed that the GAN method improved the accuracy of social robot detection.

There is a series of questions that need to be explored and answered in our future work. One potential problem to resolve is the difficulty of training a stable network into a GAN to reach a Nash equilibrium. Moreover, we need to choose more effective features for social robot detection to create more realistic social robot samples.

References

[1] Christopher A. Cassa, Rumi Chunara, Kenneth Mandl, and John S. Brownstein, "Twitter as a sentinel in emergency situations: lessons from the Boston marathon explosions," *PLoS Current*, vol. 5, July, 2013. Article (CrossRef Link)

[2] Michael Conover, Jacob Ratkiewicz, Matthew Francisco, Bruno Gonc¸alves, Filippo Menczer, and Alessandro Flammini, "Political polarization on Twitter," in *Proc. of the 5th International AAAI Conference on Weblogs and Social Media*, pp. 89-96, July 17-21, 2011. Article (CrossRef Link)

[3] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on twitter: Human, bot, or cyborg?" in *Proc. of the 26th Annual Computer Security Applications Conference*, pp.21-30, December 6-10, 2010. Article (CrossRef Link)

[4] Onur Varol, Emilio Ferrara, Clayton A. Davis, Filippo Menczer and Alessandro Flammini, "Online human-bot interactions: detection, estimation, and characterization," in *Proc. of the Eleventh International AAAI Conference on Web and Social Media*, pp.280-289, May 15-18, 2017. Article (CrossRef Link)

[5] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y. Zhao, and Yafei Dai, "Uncovering social network sybils in the wild," in *Proc. of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp.259-268, November 2-4, 2011. Article (CrossRef Link)

[6] Zafar Gilani, Reza Farahbakhsh, Gareth Tyson, Liang Wang and Jon Crowcroft, "Of Bots and Humans (on Twitter)," in *Proc. of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp.349-354, July 31- August 3, 2017. Article (CrossRef Link)

[7] V. S. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, K. Lerman, L. Zhu, E. Ferrara, A. Flammini, and F. Menczer, "The darpa twitter bot challenge," *Computer*, vol.49, no. 6, pp.38-46, June, 2016. Article (CrossRef Link)

[8] R. Zafarani and H. Liu, "10 Bits of Surprise: Detecting Malicious Users with Minimum Information," in *Proc. of the 24th ACM International on Conference on Information and Knowledge Management*, pp.423-431, October 18-23, 2015. Article (CrossRef Link)

[9] E. M. Clark, J. R. Williams, R. A. Galbraith, C. A. Jones, C. M. Danforth, and P. S. Dodds, "Sifting robotic from organic text: a natural language approach for detecting automation on twitter," *Journal of Computational Science*, vol. 16, pp.1-7, September 2016. Article (CrossRef Link)

[10] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y Zhao, "You Are How You Click: Clickstream Analysis for Sybil Detection," in *Proc. of the 22nd USENIX Security Symposium*, pp.241-256, August 14-16, 2013. Article (CrossRef Link)

[11] Qiang Cao, Michael Sirivianos, Xiaowei Yang and Tiago Pregueiro, "Aiding the Detection of Fake Accounts in Large Scale Social Online Services," in *Proc. of the 10th USENIX Symposium on Networked Systems Design and Implementation*, pp.15-15, April 25-27, 2012. Article (CrossRef Link)

[12] C Cai , L Li and D Zengi, "Behavior Enhanced Deep Bot Detection in Social Media," in *Proc. of IEEE International Conference on Intelligence and Security Informatics*, pp.128-130, July 22-24, 2017. Article (CrossRef Link)

[13] Nikan Chavoshi, Hossein Hamooni and Abdullah Mueen, "DeBot: Twitter bot detection via warped correlation," in *Proc. of the 16th IEEE International Conference on Data Mining*, pp.817-822, December 12-15, 2016. Article (CrossRef Link)

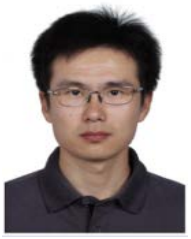
[14] Sneha Kudugunta and Emilio Ferrara, "Deep Neural Networks for Bot Detection," *Information Sciences*, vol. 467, pp.312-322, October, 2018. Article (CrossRef Link)

[15] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow and Christos Faloutsos, "CopyCatch: stopping group attacks by spotting lockstep behavior in social networks," in *Proc. of the 22nd international conference on World Wide Web*, pp.119-130, May 13-17, 2013. Article (CrossRef Link)

[16] Christoforos C. Charalambous and Anil A. Bharath, "A data augmentation methodology for training machine/deep learning gait recognition algorithms," in *Proc. of the British Machine Vision Conference (BMVC)*, pp. 110.1-110.12, September 19-22, 2016. Article (CrossRef Link)
[17] Joseph Lemley, Shabab Bazrafkan and Peter Corcoran, "Smart Augmentation Learning an Optimal Data Augmentation Strategy," *IEEE Access*, vol. 5, pp. 5858-5869, March 2017. Article (CrossRef Link)
[18] Antreas Antoniou, Amos Storkey and Harrison Edwards, "Data Augmentation Generative Adversarial Networks," *arXiv:1711.04340*, 2017. Article (CrossRef Link)
[19] Bushra Zafar, Rehan Ashraf, Nouman Ali, Mudassar Ahmad, Sohail Jabbar and Savvas A. Chatzichristofis, "Image classification by addition of spatial information based on histograms of orthogonal vectors," *PLoS ONE*, vol. 13, no. 6, June 2018.e01198175. Article (CrossRef Link)
[20] Luan Quoc Tran, Xi Yin and Xiaoming Liu, "Disentangled Representation Learning GAN for Pose-Invariant Face Recognition," in *Proc. of IEEE Computer Vision and Pattern Recognition*, pp. 1415-1424, July 22-25, 2017. Article (CrossRef Link)
[21] Biao Lenga, Kai Yua and Jingyan Qin, "Data augmentation for unbalanced face recognition training sets," *Neurocomputing*, vol. 235, pp. 10-14, December 2016. Article (CrossRef Link)
[22] Wei-Ning Hsu, Yu Zhang and James Glass, "Unsupervised Domain Adaptation for Robust Speech Recognition via Variational Autoencoder-Based Data Augmentation," in *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec 16-20, 2017. Article (CrossRef Link)
[23] Xiaodong Cui, Vaibhava Goel and Brian Kingsbury, "Data Augmentation for deep neural network acoustic modeling," in *Proc. Of IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 9, pp. 1469-1477, Sept 2015. Article (CrossRef Link)
[24] Marzieh Fadaee, Arianna Bisazza, and Christof Monz, "Data Augmentation for Low-Resource Neural Machine Translation," in *Proc. of Association for Computational Linguistics*, pp. 567–573, July 30-August 4, 2017. Article (CrossRef Link)
[25] Thomas K Landauer, Peter W. Foltz and Darrell Laham, "An introduction to latent semantic analysis," *Discourse Processes*, vol. 25, no. 2-3, pp. 259-284, November, 2009. Article (CrossRef Link)
[26] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio, "Generative adversarial nets," in *Proc. of Annual Conference on Neural Information Processing Systems*, December 8-13, 2014. Article (CrossRef Link)
[27] Ugo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti and Francesco Palmieri, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Information Sciences*, vol. 479, pp. 448-455, April, 2019. Article (CrossRef Link)
[28] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi and Maurizio Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in *Proc. of the 26th International Conference on World Wide Web Companion*, pp.963-972, April 3-7, 2017. Article (CrossRef Link)
[29] Nitesh V. Chawla, Kevin W. Bowyer and Lawrence O. Hall, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp.321-357, January, 2002. Article (CrossRef Link)
[30] Hui Han，Wen Yuan Wang，Bing Huan Mao, "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning," in *Proc. of International Conference on Intelligent Computing*, pp.878-887, August 23-26, 2005. Article (CrossRef Link)
[31] Seyda Ertekin，Jian Huang，Léon Bottou and C. Lee Giles, "Learning on the Border: Active Learning in Imbalanced Data Classification," in *Proc. of the Sixteenth ACM Conference on Information and Knowledge Management*, pp. 127-136, November 6-10, 2007. Article (CrossRef Link)
[32] Seyda Ertekin，Jian Huang and C. Lee Giles, "Active Learning for Class Imbalance Problem," in *Proc. of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp.823-824, July 23-27, 2007. Article (CrossRef Link)

[33] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li, "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning," in *Proc. of International Joint Conference on Neural Networks*, pp.1322-1328, July 1-6, 2008. Article (CrossRef Link)

[34] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel，"Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol.12, pp.2825-2830，February 2011. Article (CrossRef Link)



Bin Wu: He received his Ph.D. degree in signal and information processing from Beijing University of Posts and Telecommunications. He is currently a lecture in the National Disaster Recovery Technology Engineering Laboratory, Beijing University of Posts and Telecommunications. His research interests include network security, intrusion detection, social engineering, and artificial intelligence security.

Le Liu: He is now pursuing his master degree in School of Cyberspace Security, Beijing University of Posts and Telecommunications (BUPT), he received the bachelor's degree in cyberspace security from BUPT. His research interests include network security, social engineering, and artificial intelligence security.

Zhengge Dai: He is now pursing the bachelor degree with International school, Beijing University of Posts and Telecommunications, Beijing, China. His main research interests lie in recurrent neural network, variational auto-encoder and generative adversarial network.

Kangfeng Zheng: He received his Ph.D degree in Information and Signal Processing in July 2006 from Beijing University of Posts and Telecommunications. He is currently an associate professor at School of Cyberspace Security, Beijing University of Posts and Telecommunications. His research interests include network security and network data analysis.

Xiujuan Wang: She received her PhD degree of Information and Signal Processing in July 2006 at Beijing University of Posts and Telecommunications. She is currently an instructor lecturer at College of Computer Sciences, Beijing University of Technology. Her research interests include information and signal processing, network security and network coding.