Received February 10, 2020, accepted February 18, 2020, date of publication February 21, 2020, date of current version March 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2975630

Using Improved Conditional Generative Adversarial Networks to Detect Social Bots on Twitter

BIN WU[1], LE LIU[1], YANQING YANG[1], KANGFENG ZHENG[1], AND XIUJUAN WANG[2]
[1]School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

Corresponding author: Bin Wu (binwu@bupt.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802703, and in part by the Beijing Natural Science Foundation under Grant 4202002.

ABSTRACT The detection and removal of malicious social bots in social networks has become an area of interest in industry and academia. The widely used bot detection method based on machine learning leads to an imbalance in the number of samples in different categories. Classifier bias leads to a low detection rate of minority samples. Therefore, we propose an improved conditional generative adversarial network (improved CGAN) to extend imbalanced data sets before applying training classifiers to improve the detection accuracy of social bots. To generate an auxiliary condition, we propose a modified clustering algorithm, namely, the Gaussian kernel density peak clustering algorithm (GKDPCA), which avoids the generation of data-augmentation noise and eliminates imbalances between and within social bot class distributions. Furthermore, we improve the CGAN convergence judgment condition by introducing the Wasserstein distance with a gradient penalty, which addresses the model collapse and gradient disappearance in the traditional CGAN. Three common oversampling algorithms are compared in experiments. The effects of the imbalance degree and the expansion ratio of the original data on oversampling are studied, and the improved CGAN performs better than the others. Experimental results comparing with three common oversampling algorithms show that the improved CGAN achieves the higher evaluation scores in terms of F1-score, G-mean and AUC.

INDEX TERMS Social bot detection, conditional generative adversarial networks, data augmentation, supervised classification, imbalanced data.

I. INTRODUCTION

In recent years, online social networks (OSNs), in which people can conveniently share and promote news, information, opinions, links, and products, have grown widely. Notably, the increase in the number of mobile devices has contributed to an increase in the frequency of user interaction via online social networks. In the first quarter of 2009, Facebook had 197 million monthly active users. By the first quarter of 2019, the number of monthly active users had grown to 2.38 billion [1]. In the first quarter of 2010, Twitter had 30 million monthly active users [2]. By 2019, the number of monthly active Twitter users had grown to 330 million [3]. However, the typical features of openness and sharing in online social networks have also promoted malicious activity by attackers, spammers, and fraudsters. Social bots are one of the most high-level security threats in OSNs, as they make OSNs vulnerable to adversaries. A social bot is computer software that automatically produces content and interacts with humans on social media to emulate and possibly alter their behavior [4]. The primary objectives of these social bots are to create the illusion that a social network actively influences public opinion [5], to cause political penetration [6], and to spread malicious content. On popular social networks, these malicious social bots have had a negative impact on human users. According to a survey from The Washington Post from May 2018, Twitter identified more than 9.9 million suspicious accounts—triple the number in late 2017 [7]. Another study found that social bots were responsible for generating 35% of the content that is posted on Twitter [8].

As malicious social bots have increasingly used various social engineering methods to distribute unsolicited spam,

The associate editor coordinating the review of this manuscript and approving it for publication was Jihwan P. Choi.

advertise events and products of dubious legality, promote public figures, and steal sensitive personal information [4], many studies have examined how to detect fake accounts generated by social bots. Social bot detection aims to distinguish between bots and normal humans in social networks [9]. Machine learning has been used to detect social bots on social networks through various approaches, including by detecting fake content linked to the account, investigating the account profile itself, and using nonverbal indicators [10].

Malicious social bots affect information security and network environments by performing harmful operations and disseminating false information with malicious intent. Therefore, it is a crucial and urgent task to detect and remove malicious social bots in online social networks. However, in the real world the number of social bots is far less than the number of normal humans, leading to a serious sample imbalance problem in social network bot detection methods based on machine learning, and this conclusion has been verified in experimental environments [11]. In social bot classification detection using machine-learning methods, imbalances in training data can cause a difference in the proportion of positive and negative samples, and the final result may be misjudged to some extent [10]. This data imbalance can be addressed by both undersampling and oversampling techniques. In a social-bot-detection scenario, undersampling is not suitable because of the loss of data information in most categories. The data augmentation algorithm is an important method for solving the problem of data-set imbalance faced by the oversampling technique, and it has been applied to fields including computer vision, scene reconstruction, voice data augmentation, and natural language processing [12].

To overcome these difficulties, in this work, we propose a data-augmentation approach to address the imbalance in Twitter bot detection by adopting conditional generative adversarial networks (CGANs). Specifically, we propose using the power of the recently discovered density peak clustering to fulfill the task of the condition set of the CGAN as well as using the Wasserstein distance with a gradient penalty, which minimizes a different distribution divergence than the original CGAN, to achieve better performance in terms of convergence.

The main contributions of this paper are as follows:

1. We propose a data-augmentation approach by adopting improved CGAN generative methods to extend social bot samples.

2. We improve the CGAN model convergence judgment condition by incorporating the Wasserstein distance with a gradient penalty, which has achieved a better performance that minimizes the different distribution divergence.

3. Based on the clustering algorithm, in conjunction with using a CGAN to rebalance skewed data sets, we propose an effective imbalanced data oversampling method that avoids the generation of data-augmentation noise and effectively overcomes the imbalances between and within class distribution.

4. We propose Gaussian kernel density peak clustering (GKDPCA) as a condition model to label the social bot data in the CGAN model. The GKDPCA improves the Euclidean distance calculation method of the density peak clustering algorithm. We use the Gaussian kernel function to project the original features into a high-dimensional kernel space.

The rest of this paper is organized as follows: Section II briefly reviews related research. Section III presents the method for the detection algorithms for malicious social bots, followed by the experiment and result analysis in section IV. Section V concludes this paper.

II. RELATED WORK

A. SOCIAL BOT DETECTION BASED ON FEATURE EXTRACTION

At present, popular technology involved in social bot detection is based on dynamic content sent by social bots and the social relationship diagram around social bots. This technology requires processing data sets that are acquired beforehand and then selecting some representative and discrimination features to achieve better classification results. For example, Alothali *et al.* [13] reviewed the various detection schemes that were currently in use, examined their common aspects, such as the classifier and data sets used, selected some of the features employed, and compared the evaluation techniques employed to validate the classifiers. Similarly, Chu *et al.* [14] 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 periodicity for a specific user. The machine-learning-based component detects spam according to the content of tweets, and the account-properties component detects account information. The decision-maker determines whether the input account is a social bot. Varol *et al.* [15] extracted 1150 features from public data and metadata regarding users, friends, tweet content and sentiments, network patterns, and activity time series and used the random forest, AdaBoost, logistic regression, and decision-tree classifiers to detect social bots. The best classifier in terms of the area under the 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, but some content and emotional features were redundant. Yang *et al.* [16] 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. Furthermore, Gilani *et al.* [17] 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, The Defense Advanced Research Projects Agency (DARPA) held a competition [10] in using machine learning to detect a social bot. Using the five features of tweet syntax, tweet semantics, temporal behavior features, user profile features, and network features, six teams found the most effective combinations of these features and machine-learning algorithms by judging the comprehensive results of the detection. Zafarani and Liu [18] proposed a method for identifying malicious users with minimal information. This method classified the features of malicious users into five categories, and the detection framework generated by machine learning demonstrated strong robustness in different algorithms and imbalanced data sets. Similarly, Clark *et al.* [19] maintained that excessive dependence on user metadata could make bots with strong imitation abilities difficult to detect. Therefore, the language attributes of tweets were used as the basis for the classification. These researchers calculated the mean and standard deviation of each dimension through the user data of humans and then calculated the distance between an unknown user and the attribute average to identify a social bot. This method can be used to dynamically prevent a bot account from manipulating user attributes and hiding its real identity. Moreover, Van Der Walt and Eloff [20] linked engineered features such as the "friend-to-followers ratio" to a set of fake human accounts in the hope of advancing the successful detection of fake identities created by humans on social media platforms. Loyola-González *et al.* [21] used a pattern-based classification mechanism for social bot detection specifically for Twitter and introduced a new feature model for social bot detection that partially extended an existing model with features based on Twitter account usage and a tweet content sentiment analysis.

B. SOCIAL BOT DETECTION WITHOUT FEATURE EXTRACTION

Social bot detection can also be achieved without feature extraction. Wang *et al.* [22] made use of a clickstream model to detect the real identity of social accounts on the server side. These authors input the clickstream sequence and then calculated the sequence distance to accurately classify social accounts. Shi *et al.* [23] presented a novel method of detecting malicious social bots, including both feature selection based on the transition probability of clickstream sequences and semi-supervised clustering. This method not only analyzed the transition probability of user behavior clickstreams but also considered the temporal features of behavior.

Cao *et al.* [24] developed a tool called the sybil rank, which ranked the user's probability of being impersonated by using social graph attributes. Similarly, Cai *et al.* [25] combined convolutional neural networks (CNNs) with a long short-term memory (LSTM) model to explore semantic information and a potential time model. This method utilized content information and behavior information and converted user content into temporal text data to reduce the workload for determining features. Chavoshi *et al.* [26] 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 bot detection. In addition, Kudugunta and Ferrara [27] judged whether a social media account was a social bot by analyzing a single tweet. They introduced the contextual LSTM deep neural network, which used content and metadata as input. This model can accurately be used to judge the category of a 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. Furthermore, Beutel *et al.* [28] detected lockstep page-like patterns on Facebook by analyzing only the social graph between users and pages during times at which the edges in the graph indicated malicious social bots. Finally, Cresci *et al.* [29] showed that a new wave of social spambots had emerged and efficient spambot detection could be achieved via an in-depth analysis of their collective behaviors by exploiting the digital DNA technique for modeling the behaviors of social network users. All of the detection methods based on machine learning inevitably require a large number of original data sets. Moreover, they do not explicitly address the imbalance in the ratio of the positive and negative samples in the original data set. An imbalance between positive and negative samples reduces the effectiveness of detection.

C. APPROACHES FOR IMBALANCED DATA CLASSIFICATION

Similar to the imbalanced data in social bot detection, data imbalance problem imposes challenges in performing data classification in realistic machine learning applications, such as fraud detection, text classification, face and image recognition, and medical diagnosis [30], [31]. The performance of classifiers leans to be partial toward majority classes in imbalanced data sets [32]. Various approaches have been proposed to overcome the problem of imbalanced data classification in the past decade [33], [34]. These approaches can be broadly divided into two main categories: data-driven approaches and algorithm-driven approaches [35], [36].

At the data level, data-driven approaches use various methods to adjust the number of samples in the original data set to achieve balanced class distribution. The typical methods include undersampling, oversampling or a combination of both [37]. Random undersampling is known as the primary under-sampling technique, although it reduces some useful information in the data set. Condensed nearest neighbor rule (CNN) [38], Tomek Links [39], one-sided selection (OSS) [40] are effective under-sampling approaches proposed earlier. Laurikkala [41] proposed the neighborhood cleaning rule (NCL) method, which improves the recognition accuracy of small samples. Wilson [42] proposed edited nearest neighbor rule (ENN) method, which uses a modified three-nearest-neighbor rule to edit the pre-classified samples and using the single-nearest-neighbor rule to make decisions.

The undersampling removes some set of data samples from the majority class, which has the possibility of losing informative samples. In contrast, the oversampling approach tends to increase the size of the minority class to obtain balanced classes, which results in overfitting and increases the time of the training phase [31].

SMOTE [43] is the pioneering oversampling algorithm proposed by Chawla et al. in which the synthetic samples are created in the feature space by interpolating between the considered minority sample and its random minority nearest neighbors. To overcome the overgeneralization problem of SMOTE, many creative methods have been developed. Barua *et al.* [44] proposed MWMOTE, in which the minority samples are partitioned into small clusters, and the synthetic samples can only be generated inside the clusters. Han *et al.* [45] proposed Borderline-SMOTE, in which only the minority examples near the borderline are oversampled. Bunkhumpornpat *et al.* [46] proposed safe-level SMOTE, which samples minority instances along the same line with safe level computed by using nearest neighbor minority instances. Douzas *et al.* [47] proposed k-means-SMOTE by combining k-means clustering and SMOTE, which avoids the generation of noise and effectively overcame imbalances between and within classes. Tuanfei Zhu et al. successively proposed synthetic minority oversampling for multiclass imbalance (SMOM) [48] and synthetic minority oversampling for imbalanced ordinal regression (SMOR) [49]. SMOM is a k-NN based synthetic minority oversampling algorithm which assigns a selection weight to each neighbor direction. SMOR further generalizes the idea of SMOM and calculates the selection weights to create synthetic samples to solve imbalanced ordinal regression problem. Zhang *et al.* [50] proposed the Abstention-SMOTE method, which constructs abstaining classifiers using ROC analysis and uses the abstaining classifiers to generate the abstention positive samples. Jiang *et al.* [51] proposed a genetic algorithm-based SMOTE algorithm (GASMOTE). GASMOTE uses different sampling rates for different minority class instances and finds a combination of optimal sampling rates. Prusty *et al.* [52] proposed weighted SMOTE (WSMOTE), where oversampling of each minority data sample is carried out based on the weight assigned to it. These weights are determined by using the Euclidean distance of a particular minority data sample relative to all remaining minority data samples. Mathew *et al.* [53] proposed a weighted kernel-based SMOTE (WK-SMOTE) to handle nonlinear separable imbalanced data. WK-SMOTE adopts the kernel method of SVM to enable original data to be linear separable in high-dimensional feature space. Cheng *et al.* [54] proposed a grouped SMOTE algorithm with noise filtering mechanism (GSMOTE-NFM) to deal with the propagation of the noisy information in the procedure of oversampling and the overlooking of the local distribution characteristics.

In addition to the series of SMOTE methods, researchers have contributed to various approaches for imbalanced data issues. He *et al.* [55] proposed the adaptive synthetic sampling approach (ADASYN) for assigning different weights to samples according to their level of complexity while learning and synthetic data are generated. In recent years, generative adversarial networks (GANs) [56] have achieved considerable success in generating convincing samples. Some researchers have focused on GANs as a new oversampling approach. Fiore *et al.* [57] trained a GAN to output mimicked minority class samples, which were then merged with training data into an augmented training set. Using the augmented training set the classification effectiveness can be improved in credit card fraud detection. Douzas and Bacao [58] used a conditional generative adversarial network (CGAN) to approximate the true data distribution and generate data for the minority classes of various imbalanced data sets. The experimental results show a significant improvement in the quality of the generated data.

At the algorithm level, researchers strive to create modern algorithms to facilitate the learning task specifically with respect to the minority class. These approaches include algorithmic centered approaches, cost-sensitive learning, and hybrid methods [35]. Chawla *et al.* [59] proposed the SMOTEBoost based on a combination of SMOTE algorithm and boosting procedure. SMOTEBoost creates synthetic examples from the minority class, indirectly changing the updating weights and compensating for skewed distributions. Joshi *et al.* [60] evaluated three existing categories of boosting algorithms for the task of mining rare classes and enhanced two of them that could achieve a better balance between recall and accuracy in mining rare classes. Tallo and Musdholifah [61] proposed a SMOTE-simple genetic algorithm (SMOTE-SGA), which is applied to determine instances to be generated and the number of synthetic instances to overcome the over-generalization problem in SMOTE. Ma and Fan [62] proposed the CURE-SMOTE algorithm, which clusters the samples of the minor class using CURE (Clustering Using Representatives) and generates artificial samples randomly between representative points and the center point.

Cost-sensitive learning is a cost-specific technique that assigns different costs to the samples in different classes and can be applied to many existing algorithms to turn them into imbalance recovery methods. Doroshenko *et al.* [63] proposed a cost-sensitive classification method for imbalanced data sets based on the neural structure of successive geometric transformation models using a piecewise-linear approach for classification. Khan *et al.* [64] proposed a cost-sensitive neural network to handle imbalanced data. Dhar and Cherkassky [65] extended the universum-SVM (U-SVM) formulation to problems with different misclassification costs and presented practical conditions for the effectiveness of the cost-sensitive U-SVM. Qiu *et al.* [66] proposed a new test-cost sensitive decision tree learning algorithm and constructed the randomly selected decision tree (RSDT),

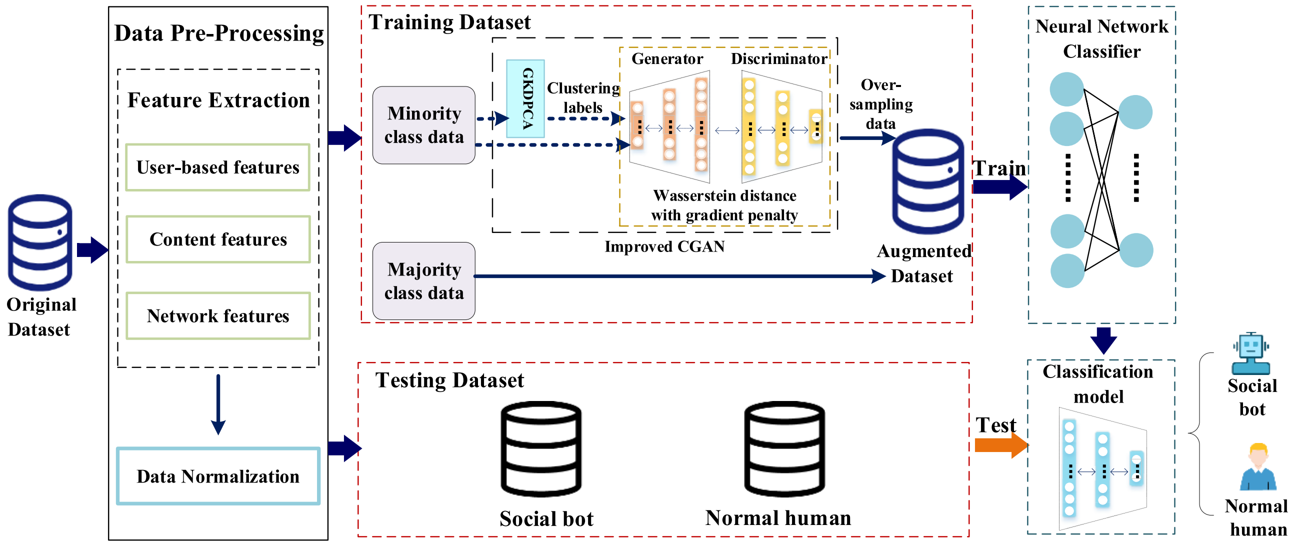


FIGURE 1. Social bot detection framework.

which significantly reduces the total test cost and maintains higher classification accuracy. Palacios *et al.* [67] proposed a cost-sensitive FURIA (fuzzy unordered rule induction algorithm) method, which verifies that cost-sensitive learning is a competitive approach for learning fuzzy rules in imbalanced classification problems.

Hybrid methods combine of multiple classifiers and pre-processing approaches to efficiently handle the problem of imbalanced data. Sun *et al.* [68] proposed a novel ensemble method for classification imbalance problem, which firstly converts an imbalanced data set into multiple balanced data sets, and then builds a number of classifiers on these data sets, finally combines the classifiers by a specific ensemble rule. Benchaji *et al.* [69] proposed a sampling method based on the k-means clustering and the genetic algorithm to improve classification of imbalanced data sets for credit card fraud detection. Awasare and Gupta [70] proposed a SVM with cluster partitioning method, where multiple sub-datasets are built from majority class samples by cluster partitioning method, and then SVM classifiers are trained with the sub datasets of majority class and the minority data sets. Barstuğan and Ceylan [71] combined dictionary learning and the ensemble classifier AdaBoost algorithm to create a hybrid structure, which uses sparse coefficients in the weight update formula of AdaBoost.

III. SOCIAL BOT DETECTION BASED ON IMPROVED CGAN

A. SOCIAL BOT DETECTION

The formal definition of the social bot detection problem 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 of the social accounts, where C_R is a collection of social-bot accounts, C_N is a collection of normal human accounts, and $|C_R| \ll |C_N|$. The aim of the social bot detection problem is to determine whether account a_i belongs to the social bot collection C_N. The decision function is as follows:

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

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

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

Since the number of bots is not of the same order as the number of people, the classification accuracy is reduced. We generate minority samples by oversampling methods, append these new samples in C_A, and mix C_R into C_A to make $|C_A| \approx |C_N|$. As a result, the classifier facilitates more effective detection of social bots through a balanced data set.

B. OVERALL PROCESS

Fig. 1 shows the social bot detection framework proposed in this paper. By collecting tweet and user-profile information, we generate the available data set from the original data set through feature extraction and data normalization.

Then, we formulate a data-augmentation approach by adopting generative adversarial network method. To overcome the shortcomings of the original CGAN—it easily causes mode collapse, and it is unable to effectively control the category of generated samples—we create an improved CGAN by introducing the Wasserstein distance with a gradient penalty. We also improve the conditional model of CGAN to control the generated samples, and we use the modified density peak clustering algorithm to generate conditions as a part of the input.

We input the bot accounts from the training set and clustering labels into the improved CGAN and then train the improved CGAN until it is stable. Then, we incorporate random noise and random labels into the stable improved

CGAN to generate fake bot accounts that are difficult for the discriminator to distinguish. Next, we mix them with the normal human data sets to form an augmented training data set. Finally, we adopt the augmented training data set to train a classifier that will be used to classify an unlabeled account in the test set.

C. FEATURE EXTRACTION

Six types of features are selected for the social-bot-detection process [15]: user meta-data, sentiment, friends, content, network, and timing. We select the following 11 types of features with better classification abilities for social bot detection:

1) Average number of topic tags: The average of the topic tags in the tweets. The "#" and other forms indicate that the tweet is highly correlated with a specific topic, and social bots want to expand influence quickly by publishing tweets on popular topics.
2) Average number of user mentions: The average number of users mentioned by the user in the tweet. Social accounts can be notified via "@username" for some tweets, and a social bot speeds up information diffusion by mentioning users more frequently than normal users do.
3) Number of links: The average number of URLs in the tweet. The tweet content in social networks can include URLs, images, and other files. The social bots are more likely to add more links than normal users do, which induces normal users to click and redirects them to malware websites for social-engineering attacks.
4) Number of retweets: 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. Normal users only retweet tweets in which they are interested, whereas social bots always retweet other users' tweets to stay active in the social network.
5) Number of favorites: The total number of the users' favorites. Normal users can express their concern by favoriting some tweets about topics of interest, but social bots also use this tactic to increase their influence. Generally the tweet content of social bots is favorited less frequently than that of normal human users.
6) Ratio of followers to the number followed: Online relationships for normal humans are with people they know in real life, whereas social bots have a small number of fans and always follow a large number of strangers to improve their influence in social networks.
7) Tweet source: The number of tweet sources that are official. Normal human users send tweets through a variety of platforms recommended by Twitter. The source of tweets sent by social bots is unofficial because social bots are controlled by automatic programs.
8) Similarity of content: The latent semantic text content similarity of the original tweet. Latent semantic analysis (LSA) [72] analyzes potential connections between documents and words by extracting the words from documents into a vector-semantic space. If different words appear in the same document multiple times, the words are semantically similar. LSA extracts the text-collection matrix from the data set. The rows of this matrix represent words, the columns represent documents, and the specific values of the matrix elements represent the number of times that a word appears in the document; then the matrix is subjected to singular value decomposition (SVD). We convert a data text into matrix A as follows:
$$A = U\Sigma V^T, \tag{3}$$
where $\Sigma = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}$ and $S = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, and $\sigma_1 \geq \ldots \geq \sigma_r > 0$. The SVD reduces the dimension of the matrix while maintaining as much column information as possible. Then, the similarity of each pair of words can be quantified by the cosine similarity of the two row vectors a and b as follows:
$$\varepsilon = \frac{\sum_{j=1}^{n} (a_j \times b_j)}{\sqrt{\sum_{j=1}^{n} (a_j)^2} \times \sqrt{\sum_{j=1}^{n} (b_j)^2}}, \tag{4}$$
where $a = (a_1 \cdots a_n)$, and $b = (b_1 \cdots b_n)$, and the closer ε is to one, the more similar the two words are. Normal human beings change their interests infrequently, so the similarity of two words should be high, but social bots often need to have considerable interests to expand their influence. Furthermore, the semantic similarity of original tweets from a normal user is higher than that of original tweets from social bots.
9) Similarity of the tweet length: The variance in the number of tweet words. The length of a tweet sent by a social bot is consistently controlled by an automatic program, whereas that of a normal human user varies greatly.
10) Similarity of punctuation usage: The variance in the number of punctuation marks in the original tweet. Different normal human users have distinct punctuation usage habits, but the diversity of social bot tweet sources results in punctuation without a fixed style.
11) Similarity of stop words: The variance in the number of stop words in the original tweet. The usage of stop words is part of an individual's writing style, so a normal user always uses more stop words in tweet content affected by the growth process. In contrast, social bots send tweets by automatic programs, which makes their writing style less predictable.

D. SOCIAL BOT AUGMENTATION WITH AN IMPROVED CGAN

In the original GAN model, we cannot control the specific types of samples being generated by generator G. Sometimes, we are interested in the "conditional on" class of social

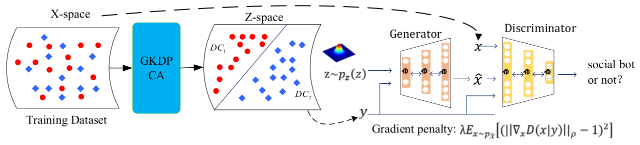


FIGURE 2. Structure of the improved CGAN with the Gaussian kernel density peak cluster algorithm.

bots. Conditional generation is performed by incorporating more information into the training procedure of GANs so that the generated samples conform to the same attributes as certain training sample categories [73]. Considering the inherent association of social bots, if the social bot samples are clustered first, then the clustering results can be used as the conditional model of the CGAN, and the augmented social bot data set can achieve a more balanced class distribution and effectively avoid the generation of noisy samples. The use of clustering enables the CGAN model to more accurately identify and target areas of the input social bot space where the generation of social bot data is most effective.

However, there are some problems with the traditional CGAN. The better the discriminator is trained, the more easily the vanishing gradient problem occurs in the generator of the traditional CGAN. The lack of diversity in the data generated by the generator leads to the collapse of the traditional CGAN. The Wasserstein distance with a gradient penalty can solve these problems perfectly.

Therefore, in this study, we improve the CGAN by introducing the Wasserstein distance with a gradient penalty and a condition-generation method through the modified density peak clustering algorithm. The structure of the improved CGAN is shown in Fig.2. The discriminator has to correctly label real samples that come from the training data set as "real," and it has to correctly label generated samples from the generator as "fake." The generator is fed with random noise, z, and the labels from the GKDPCA. The discriminator determines whether each input sample comes from the generator or the real environment.

We first describe how to incorporate clustering information into the CGAN model training process and then formulate the loss function of the improved CGAN.

1) CONDITION MODEL IN THE IMPROVED CGAN

In an unconditioned generative model, there is no control of the modes of the data being generated. However, by conditioning the model with additional information, it is possible to direct the data-generation process. Such conditioning may be based on class labels, on some part of the data for inpainting, or even on data from different modalities [73]. Thus, the GAN can be extended into a conditional model if both the generator and discriminator are conditioned with some extra information, y, which can be any kind of auxiliary information, such as class labels or data from other modalities. We perform conditioning by feeding y into both the discriminator and generator as an additional input layer.

Tweets and features of social bots are objective and immutable and are therefore not appropriate to modify and supplement. However, there are many similar factors in a social bot's characteristics, such as occupations, hobbies, and behaviors. In the existing research data on social bots, there is no recognized typical classification standard with a functional meaning division. Therefore, clustering social bots by their features is a feasible method that can result in a more reasonable classification label. The sample data of a bot may be generated by different bots, and the spatial distribution rules are different. Therefore, the category of the bot is used as auxiliary information to help the CGAN generate more realistic samples. We cluster the bot samples to obtain the category information through algorithms.

Density peak clustering is a new clustering algorithm that is different from the traditional clustering method. First, it needs to calculate the local density of each data point and the shortest distance from the data points with a higher density to each data point. Second, the cluster center is manually determined through a sorting graph. Finally, the remaining data points are assigned to the category in which the data points are higher in density and shorter in distance.

The original DPCA adopts the default Euclidean distance to compute the distance between two data points. However, when the data set is complex and linearly inseparable, the Euclidean distance can cause severe misclassification. We improve the DPCA with the GKDPCA, which measures the distance by implicitly mapping the raw data into a high-dimensional feature space. Furthermore, to improve the performance of the proposed method, we introduce the Gaussian kernel instead of the cut-off kernel to calculate the local density of the data points. The GKDPCA can detect non-spherical clustering, which cannot be detected by traditional distance-clustering methods, such as K-means and K-medoids clustering.

Given a data set S $= \{\vec{x_i}\}_{i=1}^{N}$, where $\vec{x_i} \in \mathbb{R}^d$, we use a nonlinear kernel function to map the raw data from the input space Rd to the high-dimensional feature space H, as follows:

$$\varphi : \mathbb{R}^d \to \mathbb{Z}, \vec{x_i} \to \varphi\left(\vec{x_i}\right). \tag{5}$$

where

$$\vec{x_i} = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}], \tag{6}$$

and

$$\varphi\left(\vec{x_i}\right) = [\varphi_1\left(\vec{x_i}\right), \varphi_2\left(\vec{x_i}\right) \cdots, \varphi_z\left(\vec{x_i}\right)]. \tag{7}$$

Hence, the kernel distance between two data points, $\vec{x_i}$ and $\vec{x_j}$, is calculated as follows:

$$\begin{aligned}
&\| \varphi\left(\vec{x_i}\right) - \varphi\left(\vec{x_j}\right) \|^2 \\
&= \left(\varphi\left(\vec{x_i}\right) - \varphi\left(\vec{x_j}\right)\right)^T \left(\varphi\left(\vec{x_i}\right) - \varphi\left(\vec{x_j}\right)\right) \\
&= \varphi^T\left(\vec{x_i}\right) \cdot \varphi\left(\vec{x_i}\right) - 2\varphi^T\left(\vec{x_i}\right) \cdot \varphi\left(\vec{x_i}\right) + \varphi^T\left(\vec{x_j}\right) \cdot \varphi\left(\vec{x_j}\right) \\
&= \mathrm{K}\left(\vec{x_i}, \vec{x_i}\right) - 2\mathrm{K}\left(\vec{x_i}, \vec{x_j}\right) + \mathrm{K}\left(\vec{x_j}, \vec{x_j}\right)
\end{aligned} \tag{8}$$

The Gaussian kernel can be expressed as follows:

$$\mathrm{K}\left(\vec{x_i}, \vec{x_j}\right) = \exp\left(-\frac{\| \vec{x_i} - \vec{x_j} \|^2}{2\sigma^2}\right), \quad \sigma > 0 \tag{9}$$

Clearly, $\mathrm{K}\left(\vec{x_i}, \vec{x_i}\right) = \mathrm{K}\left(\vec{x_j}, \vec{x_j}\right) = 1$, thus (8) reduces to:

$$\| \varphi\left(\vec{x_i}\right) - \varphi\left(\vec{x_j}\right) \|^2 = 2\left(1 - \mathrm{K}\left(\vec{x_i}, \vec{x_j}\right)\right). \tag{10}$$

We use this measurement to compute the distance between two data points $\vec{x_i}$ and $\vec{x_j}$, as follows:

$$d_{i,j} = \| \varphi\left(\vec{x_i}\right) - \varphi\left(\vec{x_j}\right) \| = \sqrt{2\left(1 - \mathrm{K}\left(\vec{x_i}, \vec{x_j}\right)\right)} \tag{11}$$

A data point used as a clustering center needs to have two characteristics: 1) the point should be surrounded by other points with a relatively low local density and 2) the distance between this point and other data points with higher densities should be great. Thus, the algorithm needs to calculate the local density ρ_i of each data point i and the shortest distance δ_i from that point i to a point with a higher local density. The distance can be quantified by the cutoff distance [74] or the Gaussian kernel distance [75].

The cut-off kernel distance is:

$$\rho_i = \sum\nolimits_{i \neq j} X(d_{ij} - d_c), \quad and\ X(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{12}$$

where d_{ij} is the distance between the data points i and j, and d_c is the cut-off distance set in advance.

The Gaussian kernel distance is:

$$\rho_i = \sum\nolimits_{i \neq j} e^{-\left(\frac{d_{ij}}{d_c}\right)^2} \tag{13}$$

δ_i is defined as the shortest distance between the data point i and the other data point with a higher local density, and this definition is as follows:

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}) \tag{14}$$

but $\delta_i = \max(d_{ij})$ for the data point with the highest density.

The algorithm treats the points with higher δ_i and relatively higher ρ_i at the same time as the clustering centers. We define the γ_i to find suitable cluster centers, and γ_i is defined as follows:

$$\gamma_i = \rho_i \delta_i. \tag{15}$$

According to the previous cluster center selection criterion, the larger the γ_i value of a point, the more likely it is to be a cluster center. After the cluster center is manually selected according to the number of the clusters k, the category of the remaining points is the same as that of the data point with higher density and closer distance than those of points. The GKDPCA algorithm is described below.

Algorithm 1 GKDPCA Algorithm

```
Input: Training data set $S = \{\vec{x_1}, \vec{x_2}, \cdots \vec{x_n}\}$, the number of clusters $K$, the Gaussian Kernel parameter $\sigma$.
Output: the K subsets $DC_1, DC_2, \cdots, DC_K$.
Calculate the distance $d_{i,j}$ between data points $\vec{x_i}$ and $\vec{x_j}$ according to (9).
Assign the cut-off distance $d_c$.
Calculate the local density $\rho_i$ of each data point $\vec{x_i}$ according to (13).
Calculate the distance $\delta_i$ of each data point $\vec{x_i}$ according to (14).
Calculate $\gamma_i = \rho_i \delta_i, i \in I_s$, select the data points corresponding to the K maximum values of $\{\gamma_i\}_{i=1}^N$ as the cluster centers $C = \{C_i\}_{i=1}^K$.
Finally, assign each remaining point to the same cluster as its nearest neighbor of higher density;
the training data set S is divided into K subsets $DC_1, DC_2, \cdots, DC_K$
Return the K subsets $DC_1, DC_2, \cdots, DC_K$.
```

2) CGAN WITH WASSERSTEIN DISTANCE AND A GRADIENT PENALTY

The essential loss function of a CGAN aims to optimize the maximum and minimum problems, consisting of discriminator losses and generator losses. The goal of the discriminator D is to correctly distinguish whether the source of the sample is fake or real based on whether the expected output $D(x|y)$ of real data x approaches 1 and the expected output $D(G(z|y))$ of fake data $\hat{x}$ approaches 0. Thus, $E_{x\sim p_{\text{data}}(x)} \log(D(x|y))$ and $E_{z\sim p_{z(z)}}\left[\log(1 - D(G(z|y)))\right]$ take the maximum value. The goal of the generator G is that the generated sample $G(z|y)$ is judged by the discriminator D as real data, that is, the output $D(G(z|y))$ of the discriminator approaches 1. Thus, $E_{z\sim p_{z(z)}}\left[\log(1 - D(G(z|y)))\right]$ takes the minimum value.

The objective of the discriminator D is to judge whether some samples are from x or $G(z|y)$, such that $E_{x\sim p_{\text{data}}(x)} \log(D(x|y))$. Maximizing this part also enables the discriminator D to output $D(x|y) = 1$ when x conforms to p_{data}. Another part of the problem is that generator G wants to deceive discriminator D, such that $E_{z\sim p_{z(z)}}\left[\log(1 - D(G(z|y)))\right]$. Equation (16) shows the objective function of a CGAN:

$$\min_G \max_D V(D, G) = E_{x\sim p_{\text{data}(x)}}\left[\log D(x|y)\right] + E_{z\sim p_{z(z)}}\left[\log(1 - D(G(z|y)))\right]. \tag{16}$$

There are two main problems with the traditional CGAN algorithm. When the discriminator is too good, generator training may fail as the gradient disappears. Moreover, if the sample diversity generated by the generator during training is insufficient, the generator fails to learn to represent the complex real data distribution and becomes stuck in a small space with extremely low variety, causing the model to collapse.

The Wasserstein distance can perfectly solve the above two problems.

The Wasserstein distance is also called the earth-mover (EM) distance [76] and is defined as follows:

$$W\left(\rho_{data}, \rho_g\right) = \inf_{\gamma \sim \prod(\rho_{data}, \rho_g)} E_{(x,y)\sim\gamma}\left[||\text{x-y}||\right] \tag{17}$$

where ρ_{data} is the correct distribution of the social bots, ρ_g is the distribution learned by the generator G, and $\prod(\rho_{data}, \rho_g)$ is a collection of all joint distributions of ρ_{data} and ρ_g combined. For every possible joint distribution γ, we can obtain real social bot samples x and y generated by the generator G, and calculate the distance $||x-y||$ of the pair of samples so that the expected value of the distance of the samples under the joint distribution $E_{(x,y)\sim\gamma}[||x-y||]$ can be calculated. The lower bound of this expected value in all of the possible joint distributions is defined as the Wasserstein distance.

In the actual training process, the GAN with Wasserstein distance uses weight clipping to ensure Lipschitz continuity in the whole sample space. After updating the discriminator parameters, it checks whether the absolute value of all of the parameters of the discriminator exceeds a threshold. To ensure that all parameters of the discriminator are bounded during the training process, we ensure that the discriminator can not distinguish two slightly different samples from each other, thereby indirectly meeting the Lipschitz limit. The optimal strategy in this case is to make all of the parameters as extreme as possible. The specific approach we take is to increase the gradient penalty term on the discriminator's loss function because there is no need to have Lipschitz continuity in the whole sample space, only to focus on generated samples, real samples, and the middle area [76].

In the following equation, $\hat{x}$ represents samples that are linearly interpolated by the real data x_r and the fake data x_g generated from the generator G(x):

$$\hat{x} = \varepsilon x_r + (1-\varepsilon)\, x_g \tag{18}$$

where the ε is a random number that obeys the uniform distribution U (0, 1).

The discriminator D judges whether some samples are from x or $G(z)$, and generator G tries to deceive discriminator D so that the loss function of the generator and discriminator is as follows:

$$L(G) = -E_{x\sim p_g}\left[D(x \mid y)\right] \tag{19}$$

$$L(D) = E_{x\sim p_g}\left[D(x|y)\right] - E_{x\sim p_{\text{data}}}\left[D(x \mid y)\right] + \lambda E_{x\sim p_{\hat{x}}}\left[(||\nabla_x D(x|y)||_p - 1)^2\right] \tag{20}$$

E. DETECTION PROCESS

When the generator G generates fake bot samples that are not distinguished by discriminator D, the fake samples will be mixed with the real samples as input for the network classifier. The label for real bot samples is obtained through Gaussian kernel density peak clustering, and this label is used as the auxiliary information for the improved CGAN. The training process for the improved CGAN is described in Algorithm 2.

Algorithm 2 The Process of the Improved CGAN

```
Input: The gradient penalty coefficient $\lambda$, the number of critic iterations per generation iteration $n_{\text{critic}}$, the batch size m, $Rmsprop$ hyperparameters $\alpha, \rho, \epsilon, \sigma$, initial critic parameters $\omega_0$, initial generator parameters $\theta_0$.
Output: the stable improved CGAN.
While $\theta$ has not converged do
  for $t = 1, \ldots\ldots, n_{\text{critic}}$ do
    for $i = 1, \ldots\ldots$, m do
      Sample real data $x \sim P_r$, $z \sim p(z)$, select a random number $\varepsilon \sim U[0, 1]$.
      Obtain the bot category y
      $\tilde{x} \longleftarrow G_\theta(z|y)$
      $\hat{x} \longleftarrow \varepsilon x + (1-\varepsilon)\tilde{x}$
      $L^{(i)} \longleftarrow D_w(\tilde{x}|y) - D_w(x|y) + \lambda(\left|\left|\nabla_{\hat{x}} D_w(\hat{x}|y)\right|\right|_2 - 1)^2$
    end for
    $\omega \longleftarrow Rmsprop(\alpha, \rho, \epsilon, \sigma)$
  end for
  Sample a batch of latent variables $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$
  $\theta \longleftarrow Rmsprop(\alpha, \rho, \epsilon, \sigma)$
```

The entire detection process is described in Algorithm 3.

Algorithm 3 Detection Process for Social Bots

```
Input: preprocessed data T = $\{x_1, \cdots, x_n\}$, generator of the improved CGAN $G_g$,
Output: the network classifier C
Divide $T$ into training data set $T_t$ and testing data set $T_s$;
Divide $T_t$ into minority data set $F_i$ and majority data set $F_m$;
Classify $F_i$ by density peak clustering into y;
Generate samples with $G_g$ in $F_n$;
Mix $T_t$ and $F_n$ in $C_a$;
for epochs$_{\text{classifier}}$ do
  Train C with $F_n$;
end for
return C;
```

IV. EXPERIMENTS AND ANALYSES

A. EXPERIMENTAL PREPARATION

The experiment uses 1971 normal human accounts and 462 social bot accounts as original samples, of which 891 normal users are from the data set used in [15], and the other 1080 normal users and all the social bot users are from the data set used in [29]. Social bots accounts for 18% of the total original samples. All of the data that are original tweet content crawled from Twitter are converted into a raw data set that can be used directly through feature extraction. The first part of the data is the normal user ID crawled in 2014, and official Twitter API is used to crawl all the relevant content.

TABLE 1. Data set composition.

Category	Training set	Testing set	Total
Normal user	1577	394	1971
Social bot	370	92	462
Total	1947	486	2433

TABLE 2. Confusion matrix for a two-class problem.

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

The second part of the data is the Twitter account crawled by the Twitter API in 2015 and related content.

In order to evaluate the performance of oversampling methods, we used stratified sampling to randomly divide 80% of the original data set into the training set and the remaining 20% into the testing set, so as to ensure that the training and testing sets have approximately the same percentage of samples of each target class as the original data set, as shown in Table 1. All oversampling methods are used to synthesize the training samples and balance the training set. The testing set is used to test the performance of the classifier trained by the oversampled training set. Each experiment was run independently for 10 times to reduce the impact of randomly partitioning the training and testing sets.

In a two-class problem, the confusion matrix (Table 2) records the results of correctly and incorrectly recognized examples of each class.

Traditional evaluation metrics for the two classification problems are adopted, namely, accuracy rate, accuracy, and recall, as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{21}$$

$$Precision = \frac{TP}{TP + FP} \tag{22}$$

$$Recall = \frac{TP}{TP + FN} \tag{23}$$

The F1-score is the harmonic mean of precision and recall. In imbalanced data sets, the F1-score is much more effective than accuracy in determining the performance of the model. The F1-score is defined as follows:

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{24}$$

The true positive rate (TPR) is the ratio of the number of positive predictions that are actually classified as positive to the number of all positive samples, that is, the recall rate. The TPR is defined as follows:

$$TPR = \frac{TP}{TP + FN} \tag{25}$$

The false positive rate (FPR) is the ratio of the number of negative predictions that are classified as negative to the number of all negative samples. The FPR is defined as follows:

$$FPR = \frac{TN}{TN + FP} \tag{26}$$

The G-mean (geometric mean) is the square root of the product of the recall and precision and is used to evaluate model performance for imbalanced data. The best value for the G-mean is 1, and the worst value is 0. The G-mean is defined as follows:

$$G - mean = \sqrt{TPR \cdot FPR} \tag{27}$$

When the distribution of the positive and negative samples in the test set changes, the receiver operating characteristic (ROC) curve can remain unchanged. Therefore, the ROC curve is also an important classification index. The area under the curve (AUC) is defined as the area under the ROC curve. We use the AUC as the evaluation index because the ROC curve often does not clearly indicate which classifier performs better, and with the AUC as a value, the larger the value, the better the classifier performance.

B. COMPARED ALGORITHMS

To analyse the effectiveness of the improved CGAN in detecting social bots, we use three types of oversampling methods for comparison: the synthetic minority oversampling technique (SMOTE) algorithms, the adaptive synthetic (ADASYN) algorithm, and the random oversampling method. These three algorithms are common in the field of data augmentation, and the improvement of the effect on the classifier is obvious.

1) Random Oversampling [77]: Random oversampling is a simple method of copying minority samples that makes the rules learned by the classifier too specific and causes overfitting problems.
2) SMOTE [43]: The SMOTE algorithm randomly selects positions on the line of two minority class samples as a new minority class sample. This method improves the accuracy of classifiers for minority classes by increasing the number of minority classes.
3) ADASYN [55]: The ADASYN algorithm can adaptively generate bias by reducing the data imbalance for synthetic data samples of the minority classes. At the same time, the ADASYN algorithm can be extended to handle imbalances in different scenarios.

The implementation of these three algorithms is provided by the scikit-learn library [78]. We executed these algorithms by calling the appropriate modules in the library. All of the above algorithms need to use the training data set as data input, and the algorithms have parameters that achieve the best performance. The use of these three oversampling algorithms is similar to the improved CGAN in the experiment. The three oversampling algorithms use the training set as input data to generate more samples. We use the augmented data set to train the neural network classifier and use the test set to verify the performance of the classifier.

C. EXPERIMENTAL PROCESS AND PARAMETER SETTING

1) PARAMETER SELECTION

The hyperparameters in a neural network affect the performance of the entire network, making it necessary to continuously adjust the hyperparameters of the neural network classifier until the classifier performance is optimal.

Having too few layers of the neural network can 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 have been tested in generators. Both the discriminator and classifier are used to find the best number of layers. One of the hyperparameters is the type of activation function in each layer. Common activation functions include Sigmoid, ReLU, tanh, and LeakyReLU. The LeakyReLU function is defined as follows:

$$y_i = \begin{cases} x_i & if\ x_i \geq 0 \\ \dfrac{x_i}{a_i} & if\ x_i < 0, \end{cases} \tag{28}$$

where a_i is a fixed parameter in the interval $(1, +\infty)$. In addition, the tanh function is defined as follows:

$$tanh\,(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{29}$$

The output of the generator is the input sample for the discriminator. When the input enters the neuron, it is multiplied by the weight. We randomly initialize the weights and update them during model training. In addition to the weights, another linear component applied to the input is the bias. It is added to the result of the weight multiplied by the input. For the initialization of the weights and biases, weight values are drawn from the normal distribution *N(0, 1)*, and biases are drawn from the uniform distribution *U(–1, 1)*. A neural network generally consists of an input layer, hidden layer, and output layer. Since the data dimension and the amount of data used in the experiment are small, we choose three network layers for the default implementation of the neural network.

Different optimization algorithms have different principles and are applicable to different scenarios. After many investigations, we select the RMSprop algorithms as the optimizers for testing the optimization algorithms. The learning rate, one of the hyperparameters, is the speed at which the neural network reaches the optimal status. If the learning rate is too small, then the loss of the network declines very slowly. If the learning rate is too large, then the range of the updated parameter is too large, resulting in the network convergence to a local optimum. We select values experimentally and observe that the performance is less extreme with the change in hyperparameters.

2) PARAMETER SETTING

The optimal performance of the improved CGAN should have resulted in an accuracy of the discriminator D of close to 50%. However, there are several hyperparameters

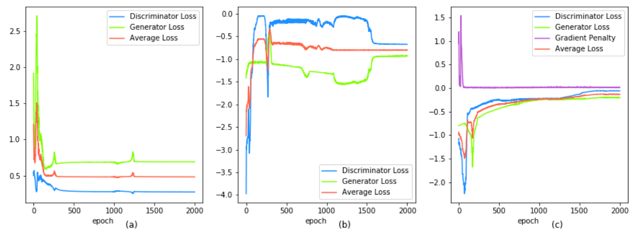


FIGURE 3. The generator loss, discriminator loss, and average loss changes during the training epochs. (a) The CGAN, (b) the WCGAN, and (c) the improved CGAN.

in the improved CGAN network that affect the performance of the improved CGAN, including network parameters for generator G and discriminator D and the parameters for random noise. Thus, we find the optimal combination of parameters through the grid search method. We performed a 10-fold cross-validation on the social bot samples of the training set to select the optimal network parameters of the improved CGAN. 10-fold cross-validation split the data into 10 equal subsets with 9 subsets used in the training the improved CGAN and the remaining subset employed in testing. According to the experience with neural network parameter setting and CGAN training, the structure of discriminator D and generator G is set to three layers. The dimensions of the input data for the discriminator are determined by our features, so the neuron numbers for the first layer of discriminator D are fixed. For the same reason, the neuron numbers for the third layer of generator G are fixed. The learning rate of the generator and discriminator have to be selected. The random noise z conforms to the random distribution *N (0, 10)*. The generator *G* in the CGAN has three layers, which contains 3 LeakyReLU units, 7 LeakyReLU units, and 11 tanh units. The discriminator also has three layers composed of 11 LeakyReLU units, 6 LeakyReLU units, and 1 linear unit. The learning rate of the generator is 5×10^{-5}, and that of the discriminator is selected as 5×10^{-5}. In addition, the optimizer of the generator and the discriminator is the RMSprop algorithm. To show the superiority of the improved CGAN, we compare the training performance with the original CGAN and WGAN. Because of the addition of the gradient penalty, the improved CGAN stabilizes more quickly, avoiding the vanishing gradient and mode collapse problems and producing more stable samples. The change in the loss of the improved CGAN is presented in Fig. 3.

During the process of training the CGAN, problems such as pattern collapse and overfitting may occur. Similarly, the WCGAN fluctuates greatly during the training process. The loss of the improved CGAN remains largely stable in the later period. The change trend of the improved CGAN is significantly different from that of the WCGAN. At the same time, the improved CGAN also solves the possible problems of the original GAN. The generated social bot samples are more consistent with the spatial distribution of the existing bots.

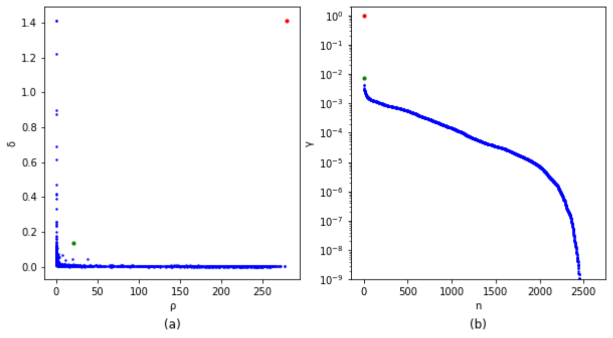


FIGURE 4. **Decision graph and γ sorting graph of the GKDPCA algorithm on the original data set.**

D. EXPERIMENTAL RESULTS AND ANALYSIS

The improved CGAN requires condition variables as auxiliary information, so we choose the Gaussian kernel density peak algorithm as the generator for the condition variables. The coefficients are calculated for all the bots by the Gaussian kernel density peak algorithm, and the number of large elements is selected as the number of categories. The decision graph of the Gaussian kernel density peak cluster is presented in Fig. 4, which shows the decision graph and the γ sorting graph on the original data set and augmented data set, respectively. Fig. 4(a) shows the relationship between the local density, ρ, and the distance, δ, of each training sample. Fig. 4(b) shows that $\gamma_i = \log \rho_i \delta_i$.; the graph is sorted in descending order.

The category of the bot is selected by using the density peak algorithm as an input. We choose to classify bot users into two categories through the decision diagram of γ.

We use T-distributed stochastic neighbor embedding (t-SNE) to map the original data set and the augmented data set from the high-dimensional space to 2D space. The t-SNE is a machine-learning algorithm for nonlinear dimensionality reduction. It is suitable for reducing high-dimensional data to 2D or 3D forms for visualization. Fig. 4 shows, the relationship between the data generated by the improved CGAN and the original data. The generated bot data and the existing bots satisfy certain correlations and differ from the normal human samples. The difference between normal humans and social bots on a 2D plane is shown in Fig. 5(a). The yellow dots represent normal human samples, the purple dots represent social bot samples, and a small number of social bots overlap with normal human samples. Some interference samples affect the accuracy of classification. As Fig. 5(b) shows, the social bot samples generated by the improved CGAN are somewhat different from the normal human samples and original social bots. The spatial distribution of social bots only includes the two purple areas in the Fig. 5(a). In contrast, the samples generated by the improved CGAN in the Fig. 5(b) also conform to the spatial distribution of existing social bots, which can improve the detection of the social bots.

To analyze the effectiveness of the improved CGAN method in detecting social bots, we performed three experiments with the original algorithm and three traditional oversampling algorithms. We use the same dataset and different oversampling methods to generate a balanced data set. The same hyperparameters and balanced data sets from different sources are used in the training process of the classifier, and the performance of the classifier is detected through the same test set.

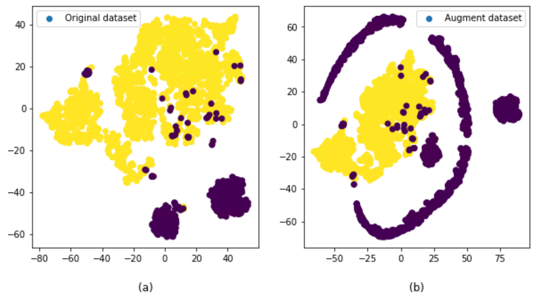
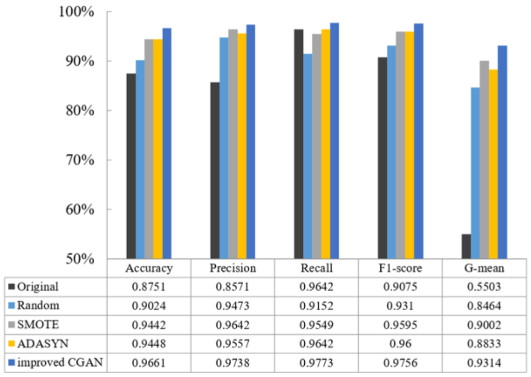


FIGURE 5. **Different data sets are projected into a low-dimensional space by t-SNE. (a) Original data set. (b) Augmented data set.**



	Accuracy	Precision	Recall	F1-score	G-mean
■ Original	0.8751	0.8571	0.9642	0.9075	0.5503
■ Random	0.9024	0.9473	0.9152	0.931	0.8464
■ SMOTE	0.9442	0.9642	0.9549	0.9595	0.9002
■ ADASYN	0.9448	0.9557	0.9642	0.96	0.8833
■ improved CGAN	0.9661	0.9738	0.9773	0.9756	0.9314

FIGURE 6. **Variation trend of the four evaluation indexes with the change in the oversampling mode.**

The experiments are conducted with the aim of answering the following questions:

1) Does the improved CGAN oversampling method improve the accuracy of bot detection?

2) Is the improved CGAN oversampling method more advantageous than other oversampling methods?

3) How many samples are generated for the most obvious improvement in bot detection?

4) When the degree of data imbalance in the original data set is different, what is the degree of difference in the improvement of the detection effect by different oversampling methods?

Experiment 1: To verify the effectiveness of the improved CGAN method in generating minority samples, the original algorithm and three popular oversampling algorithms are compared. The random oversampling, ADASYN, and SMOTE algorithms are considered in the experiment. Minority class samples are generated with the oversampling algorithms until the quantity of minority and majority samples is the same, and then the neural network is adopted to classify

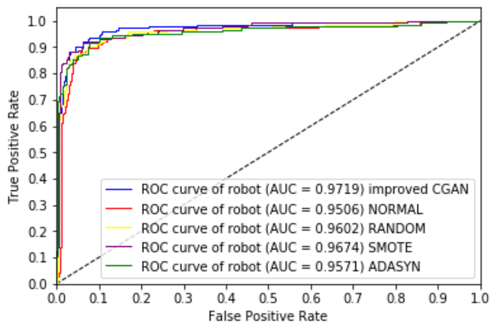


FIGURE 7. ROC curves of the five classifiers using different oversampling methods.

the social bots. The detection results are shown in Fig. 6. The ROC curves are shown in Fig. 7.

The oversampling method based on the improved CGAN model works best among all of the oversampling methods in Fig. 6. The F1-score of the improved CGAN reaches 97.56%, which is 7 percentage points higher than that of the classifier trained by the original data set. The accuracy of SMOTE and ADASYN are relatively similar, and the accuracy of the classifier generated by the improved CGAN is nine percentage points higher than that of the classifier generated by the original data set. The accuracy of the classifier generated by the improved CGAN is the highest among all of the methods. The recall of the improved CGAN methods is also the highest among all methods. The higher recall indicates that the classifier generated by the improved CGAN can detect more social bots than the others and that the generator of the improved CGAN can generate more samples fitting the law of the minority sample. The random oversampling randomly selects existing samples as new samples, which causes overfitting. The SMOTE algorithm randomly selects a new minority sample from the connection line between the nearest neighbor samples and a specific sample. Additionally, ADASYN automatically determines how many minority samples are synthesized rather than synthesizing the same number of samples for each minority sample as the SMOTE algorithm does. When the original data set is imbalanced, the G-mean value is relatively low. When the original data set is oversampled to a balanced data set, the G-mean value noticeably improves. The G-mean value of the improved CGAN is noticeably larger than that of the other three oversampling algorithms. In Fig. 7, the trends of the five ROC curves are approximately the same. However, the AUC value of the improved CGAN is the highest. The G-mean and the AUC value of the improved CGAN indicate that the data generated by the oversampling method fits the existing data well. The results also indicate that the improved CGAN is superior to other oversampling methods because the improved CGAN can learn the spatial distribution characteristics of social bots in the process of iterative training.

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

$$\varphi = a : b \tag{30}$$

where b is the number of social bots in the training data and a is the number of samples generated by the oversampling method.

The main purpose of this experiment is to examine the sensitivity of the four oversampling methods in relation to the expansion multiple. In this experiment, the number of minority samples accounts for 18% of the total number of samples. Minority samples are generated with four different φ ratios: 1:1, 2:1, 3:1, and 4:1. Thus, four different training sets are generated, and then the classification of the data is performed. The results are shown in Table 3. Observing the accuracy trend graph of the classifier, as φ changes, the accuracy of the classifier generated by SMOTE and ADASYN fluctuate significantly. The accuracy of all of the algorithms is higher than that of the original data set. The accuracy of the improved CGAN methods is stable, and its fluctuations are gentle, which indicates that the improved CGAN method is less affected by changes in the sampling proportion.

The precision rate of the classifiers generated by the SMOTE algorithm is worse than that of other methods where $\varphi = 4 : 1$, which indicates that the social bot samples generated by this method do not fully conform to the inherent distribution of existing bot samples. The precision of the classifier generated by the improved CGAN method is stable; it remains at 96%. Among the other oversampling methods, the recall rate of all the algorithms fluctuates greatly when φ changes. The recall of the classifier generated by the improved CGAN is always superior to the classifier produced by the original data set. For the classifiers generated by the other oversampling methods, the F1-score and G-mean of all oversampling methods change with φ, which indicates that φ is important for the effect of oversampling, and that the improved CGAN works best when φ is 2:1.

Experiment 3: Considering the lack of proportion between minority and majority samples in the original data, we make the number of minority samples equal to the number of majority samples to influence the performance of the oversampling methods. Therefore, this experiment is designed to check the sensitivity of the four methods relative to the imbalance degree r, which is defined as:

$$r = \frac{x_r}{x}, \tag{31}$$

where x is the number of accounts in the original data set and x_r is the number of social bots in the original data set.

We let r take the values of 10%, 15%, 20%, 25%, and 30%. In each case, specific numbers of minority samples are sampled randomly and placed into data set P together with all of the normal samples. Thus, we obtain four different

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

φ	Category (%)	Original	Random	SMOTE	ADASYN	Improved CGAN
1:1	Accuracy	90.28	91.51	92.33	93.34	94.60
	Precision	92.46	94.73	94.82	95.57	96.51
	Recall	93.26	92.86	93.99	94.73	95.65
	F1-score	92.86	93.79	94.40	95.15	96.08
	G-mean	79.37	85.08	85.67	87.74	90.30
2:1	Accuracy	90.28	91.45	94.71	94.77	96.55
	Precision	92.46	94.40	96.60	96.42	97.29
	Recall	93.26	93.10	95.74	95.40	97.73
	F1-score	92.86	93.75	96.17	95.82	97.51
	G-mean	79.37	84.36	90.54	89.99	92.94
3:1	Accuracy	90.28	89.96	91.51	92.57	94.77
	Precision	92.46	91.75	94.73	94.98	96.42
	Recall	93.26	93.34	92.86	94.15	96.00
	F1-score	92.86	92.54	93.79	94.57	96.21
	G-mean	79.37	77.41	85.08	86.12	90.22
4:1	Accuracy	90.28	89.54	90.59	92.44	92.00
	Precision	92.46	93.10	91.52	94.90	94.57
	Recall	93.26	91.52	94.73	94.07	93.75
	F1-score	92.86	92.30	93.10	94.48	94.15
	G-mean	79.37	80.65	76.98	85.90	84.98

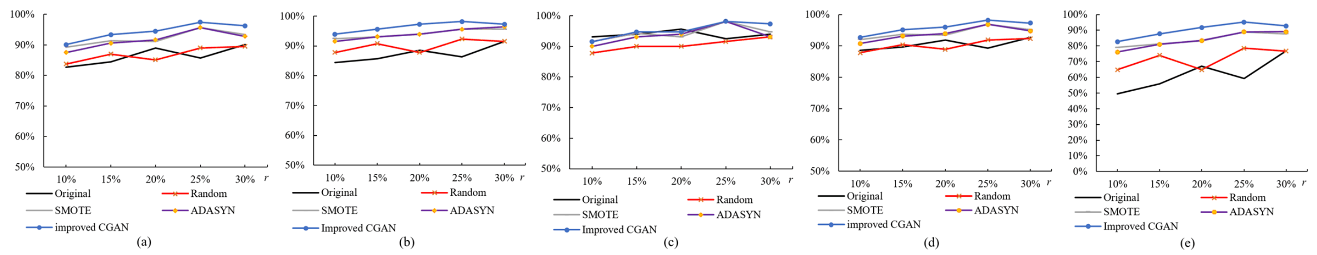


FIGURE 8. Trend diagram of the evaluation indexes pertaining to the classifier generated by different oversampling methods with changes in r: (a) accuracy, (b) precision, (c) recall, (d) F1-score, and (e) G-mean.

data sets with different r values. Then, P is oversampled using the improved CGAN and the compared algorithms previously mentioned, and then classifications are performed. The experimental results are shown in Fig. 8. As r increases, the accuracy of SMOTE, ADASYN, and the improved CGAN methods gradually increase, and the accuracy of the random algorithm fluctuates in all evaluation indexes. The improved CGAN is found performing better than all the other oversampling methods. Moreover, the detection effect of the classifiers produced by all of the oversampling methods decreases slightly with an increase in the imbalance degree when r equals 30%. This indicates that when the data set tends toward balance, it is somewhat redundant to augment the data set using oversampling methods. Only when the number of the normal human and social bots in the data set differs greatly the effect of the oversampling method becomes obvious.

V. CONCLUSION

In this paper, we propose an improved CGAN model to increase the detection accuracy of malicious social bots. Since the number of normal human beings and social bots on Twitter is imbalanced, we improve the traditional CGAN by incorporating the Wasserstein distance with a gradient penalty and a clustering algorithm as a new data-augmentation approach. Specifically, the Wasserstein distance with a gradient penalty is introduced to the CGAN loss function to solve the problem of model collapse and gradient disappearance in the traditional CGAN. In addition, we propose the GKDPCA as a conditional model to label the social bot data in the CGAN model, which improves the Euclidean distance calculation method of the density peak clustering algorithm. Furthermore, we use Gaussian kernel function to project the original features into a high-dimensional kernel space. The findings indicate that the data-augmentation approach-improved CGAN based on the GKDPCA and Wasserstein distance with a gradient penalty can rebalance skewed social bot data sets, which avoids the generation of data-augmentation noise and effectively overcomes imbalances between and within class distributions.

The proposed improved CGAN algorithm is also compared with three more common oversampling algorithms. Experimental results show that the improved CGAN outperformed the three common oversampling algorithms, with an F1 score of 97.56%. This finding indicates that it is an effective oversampling method in the field of social bot generation.

Future work may focus on malicious social bot detection. Additional behavioral patterns and feature sequences of malicious social bots will be further considered. We also want to extend this work to other social networks, such as Facebook and Instagram, with the aim of creating a model for bot detection on social networks, network processing, network security, and network coding.

REFERENCES

[1] J. Clement. (Nov. 2019). *Facebook: Number of Monthly Active Users Worldwide 2008–2019*. [Online]. Available: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/

[2] J. Clement. (Aug. 2019). *Twitter: Number of Monthly Active Users 2010–2019*. [Online]. Available: https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

[3] S. Aslam. (Sep. 2019). *Twitter by the Numbers: Stats, Demographics & Fun Facts*. [Online]. Available: https://www.omnicoreagency.com/twitter-statistics/

[4] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Commun. ACM*, vol. 59, no. 7, pp. 96–104, Jun. 2016.

[5] C. Cassa, R. Chunara, K. Mandl, and J. S. Brownstein, "Twitter as a sentinel in emergency situations: Lessons from the boston marathon explosions," *PLoS Currents*, 2013.

[6] M. Conover, J. Ratkiewicz, M. Francisco, B. Gonçalves, F. Menczer, and A. Flammini, "Political polarization on Twitter," in *Proc. 5th Int. AAAI Conf. Weblogs Social Media*, Jul. 2011, pp. 89–96.

[7] Accessed: Jul. 6, 2018. [Online]. Available: https://www.washingtonpost.com/technology/2018/07/06/twitter-is-sweeping-out-fake-accounts-like-never-before-putting-user-growth-risk/?utm_term=.5d2f229e7794

[8] N. Abokhodair, D. Yoo, and D. W. McDonald, "Dissecting a social botnet: Growth, content and influence in Twitter," in *Proc. 18th ACM Conf. Comput. Supported Cooperat. Work Social Comput. (CSCW)*, 2015, pp. 839–851.

[9] 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, Jun. 2016.

[10] G. Lingam, R. R. Rout, and D. V. L. N. Somayajulu, "Detection of social botnet using a trust model based on spam content in Twitter network," in *Proc. IEEE 13th Int. Conf. Ind. Inf. Syst. (ICIIS)*, Dec. 2018, pp. 280–285.

[11] J. Lemley, S. Bazrafkan, and P. Corcoran, "Smart augmentation learning an optimal data augmentation strategy," *IEEE Access*, vol. 5, pp. 5858–5869, Mar. 2017.

[12] C. Charalambous and A. Bharath, "A data augmentation methodology for training machine/deep learning gait recognition algorithms," in *Proc. Brit. Mach. Vis. Conf.*, Sep. 2016, pp. 110.1–110.12.

[13] E. Alothali, N. Zaki, E. A. Mohamed, and H. Alashwal, "Detecting social bots on Twitter: A literature review," in *Proc. Int. Conf. Innov. Inf. Technol. (IIT)*, Nov. 2018, pp. 175–180.

[14] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on Twitter: Human, bot, or cyborg?" in *Proc. 26th Annu. Comput. Secur. Appl. Conf.*, Dec. 2010, pp. 21–30.

[15] O. Varol, E. Ferrara, A. Clayton Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Proc. 11th Int. AAAI Conf. Web Social Media*, pp. 280–289, May 2017.

[16] Z. Yang, C. Wilson, X. Wang, T. Gao, Y. Ben Zhao, and Y. Dai, "Uncovering social network sybils in the wild," in *Proc. ACM SIGCOMM Conf. Internet Meas. Conf.*, Nov. 2011, pp. 259–268.

[17] Z. Gilani, R. Farahbakhsh, G. Tyson, L. Wang, and J. Crowcroft, "Of bots and humans (on Twitter)," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2017, pp. 349–354.

[18] R. Zafarani and H. Liu, "10 bits of surprise: Detecting malicious users with minimum information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 423–431.

[19] E. M. Clark, J. R. Williams, C. A. Jones, R. A. Galbraith, C. M. Danforth, and P. S. Dodds, "Sifting robotic from organic text: A natural language approach for detecting automation on Twitter," *J. Comput. Sci.*, vol. 16, pp. 1–7, Sep. 2016.

[20] E. Van Der Walt and J. Eloff, "Using machine learning to detect fake identities: Bots vs humans," *IEEE Access*, vol. 6, pp. 6540–6549, 2018.

[21] O. Loyola-González, R. Monroy, J. Rodríguez, A. López-Cuevas, and J. I. Mata-Sánchez, "Contrast pattern-based classification for bot detection on Twitter," *IEEE Access*, vol. 7, pp. 45800–45817, 2019.

[22] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection," in *Proc. 22nd USENIX Secur. Symp.*, Aug. 2013, pp. 241–256.

[23] P. Shi, Z. Zhang, and K.-K.-R. Choo, "Detecting malicious social bots based on clickstream sequences," *IEEE Access*, vol. 7, pp. 28855–28862, 2019.

[24] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *Proc. 10th USENIX Symp. Networked Syst. Des. Implement.*, pp. 197–210, Apr. 2012.

[25] C. Cai, L. Li, and D. Zengi, "Behavior enhanced deep bot detection in social media," in *Proc. IEEE Int. Conf. Intell.and Secur. Inform. (ISI)*, Jul. 2017, pp. 128–130.

[26] N. Chavoshi, H. Hamooni, and A. Mueen, "DeBot: Twitter bot detection via warped correlation," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 817–822.

[27] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Inf. Sci.*, vol. 467, pp. 312–322, Oct. 2018.

[28] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "CopyCatch: Stopping group attacks by spotting lockstep behavior in social networks," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, 2013, pp. 119–130.

[29] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Social Fingerprinting: Detection of Spambot Groups Through DNA-Inspired Behavioral Modeling," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 561–576, Jul. 2018.

[30] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 687–719, 2009.

[31] H. Kaur, H. S. Pannu, and A. K. Malhi, "A systematic review on imbalanced data challenges in machine learning: Applications and solutions," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–36, Aug. 2019.

[32] X. Xu, W. Chen, and Y. Sun, "Over-sampling algorithm for imbalanced data classification," *J. Syst. Eng. Electron.*, vol. 30, no. 6, pp. 1182–1191, 2019.

[33] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017.

[34] R. Longadge, S. Snehlata Dongre, and L. Malik, "Class imbalance problem in data mining: Review," *Int. J. Comput. Sci. Netw.*, vol. 2, pp. 83–87, Feb. 2013.

[35] F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, "Data imbalance in classification: Experimental evaluation," *Inf. Sci.*, vol. 513, pp. 429–441, Mar. 2020.

[36] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Netw.*, vol. 106, pp. 249–259, Oct. 2018.

[37] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Inf. Sci.*, vol. 250, pp. 113–141, Nov. 2013.

[38] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Inf. Theory*, vol. 14, no. 3, pp. 515–516, May 1968.

[39] I. Tomek, "Two modifications of CNN," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, no. 11, pp. 769–772, Nov. 1976.

[40] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proc. Int. Conf. Mach. Learn.*, Jul. 1997, pp. 179–186.

[41] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Artificial Intelligence in Medicine*. Berlin, Germany: Springer, Jun. 2001, pp. 63–66.

[42] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-2, no. 3, pp. 408–421, Jul. 1972.

[43] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[44] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE-majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 405–425, Feb. 2014.

[45] H. Han, W. Wang, and B. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intell. Comput.*, vol. 3644, 2005, pp. 878–887.

[46] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*, vol. 5476, 2009, pp. 475–482.

[47] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," *Inf. Sci.*, vol. 465, pp. 1–20, Oct. 2018.

[48] T. Zhu, Y. Lin, and Y. Liu, "Synthetic minority oversampling technique for multiclass imbalance problems," *Pattern Recognit.*, vol. 72, pp. 327–340, Dec. 2017.

[49] T. Zhu, Y. Lin, Y. Liu, W. Zhang, and J. Zhang, "Minority oversampling for imbalanced ordinal regression," *Knowl.-Based Syst.*, vol. 166, pp. 140–155, Feb. 2019.

[50] C. Zhang, Y. Chen, X. Liu, and X. Zhao, "Abstention-SMOTE: An over-sampling approach for imbalanced data classification," in *Proc. Int. Conf. Inf. Technol.*, pp. 17–21, Dec. 2017.

[51] K. Jiang, J. Lu, and K. Xia, "A novel algorithm for imbalance data classification based on genetic algorithm improved SMOTE," *Arabian J. Sci. Eng.*, vol. 41, no. 8, pp. 3255–3266, May 2016.

[52] M. R. Prusty, T. Jayanthi, and K. Velusamy, "Weighted-SMOTE: A modification to SMOTE for event classification in sodium cooled fast reactors," *Progr. Nucl. Energy*, vol. 100, pp. 355–364, Sep. 2017.

[53] J. Mathew, C. K. Pang, M. Luo, and W. H. Leong, "Classification of imbalanced data by oversampling in kernel space of support vector machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4065–4076, Sep. 2018.

[54] K. Cheng, C. Zhang, H. Yu, X. Yang, H. Zou, and S. Gao, "Grouped SMOTE with noise filtering mechanism for classifying imbalanced data," *IEEE Access*, vol. 7, pp. 170668–170681, Nov. 2019.

[55] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, Jun. 2008, pp. 887–1322.

[56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf.Process. Syst.*, pp. 2672–2680, 2014.

[57] U. Fiore, A. De Santis, F. Perla, P. Zanetti, and F. Palmieri, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Inf. Sci.*, vol. 479, pp. 448–455, Apr. 2019.

[58] G. Douzas and F. Bacao, "Effective data generation for imbalanced learning using conditional generative adversarial networks," *Expert Syst. Appl.*, vol. 91, pp. 464–471, Jan. 2018.

[59] N. V. Chawla, A. Lazarevic, O. Lawrence Hall, and K. Bowyer, "SMOTE-Boost: Improving prediction of the minority class in boosting," in *Proc. 7th Eur. Conf. Princ. Pract. Knowl. Discovery Database*, Jan. 2003, pp. 107–119.

[60] M. V. Joshi, V. Kumar, and R. C. Agarwal, "Evaluating boosting algorithms to classify rare classes: Comparison and improvements," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2001, pp. 257–264.

[61] T. E. Tallo and A. Musdholifah, "The implementation of genetic algorithm in SMOTE (Synthetic Minority Oversampling Technique) for handling imbalanced dataset problem," in *Proc. 4th Int. Conf. Sci. Technol.*, Aug. 2018, pp. 1–4.

[62] L. Ma and S. Fan, "CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests," *BMC Bioinf.*, vol. 18, no. 1, Mar. 2017.

[63] A. Doroshenko, "Piecewise-linear approach to classification based on geometrical transformation model for imbalanced dataset," in *Proc. IEEE 2nd Int. Conf. Data Stream Mining Process. (DSMP)*, Aug. 2018, pp. 231–235.

[64] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3573–3587, Aug. 2018.

[65] S. Dhar and V. Cherkassky, "Development and evaluation of cost-sensitive universum-SVM," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 806–818, Apr. 2015.

[66] C. Qiu, L. Jiang, and C. Li, "Randomly selected decision tree for test-cost sensitive learning," *Appl. Soft Comput..*, vol. 53, pp. 27–33, Apr. 2017.

[67] A. Palacios, K. Trawiński, O. Cordón, and L. Sánchez, "Cost-sensitive learning of fuzzy rules for imbalanced classification problems using FURIA," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 22, no. 05, pp. 643–675, Oct. 2014.

[68] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognit.*, vol. 48, no. 5, pp. 1623–1637, May 2015.

[69] I. Benchaji, S. Douzi, and B. ElOuahidi, "Using genetic algorithm to improve classification of imbalanced data sets for credit card fraud detection," in *Proc. 2nd Cyber Secur. Netw. Conf.*, Oct. 2018, pp. 1–5.

[70] V. K. Awasare and S. Gupta, "Classification of imbalanced data sets using partition method and support vector machine," in *Proc. 2nd Int. Conf. Electr., Comput. Commun. Technol.*, pp. 1–7, Feb. 2017.

[71] M. Barstuğan and R. Ceylan, "A discriminative dictionary learning-AdaBoost-SVM classification method on imbalanced data sets," in *Proc. Int. Artif. Intell. Data Process. Symp.*, pp. 1–4, Sep. 2017.

[72] T. K. Landauer, W. Peter Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse Process.*, vol. 25, nos. 2–3, pp. 259–284, Nov. 2009.

[73] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*. [Online]. Available: http://arxiv.org/abs/1411.1784

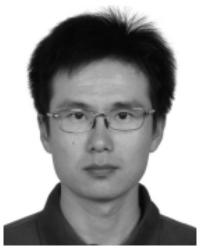
[74] L. Li, H. Zhang, H. Peng, and Y. Yang, "Nearest neighbors based density peaks approach to intrusion detection," *Chaos, Solitons Fractals*, vol. 110, pp. 33–40, May 2018.

[75] Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks," *Appl. Sci.*, vol. 9, no. 2, p. 238, Jan. 2019.

[76] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," in *Proc. Neural Inf. Process. Syst.*, Dec. 2017, pp. 5767–5777.

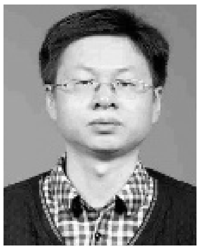
[77] A. Liu, J. Ghosh, and E. Cheryl Martin, "Generative oversampling for mining imbalanced data sets," in *Proc. Int. Conf. Data Mining*, Jun. 2007, pp. 66–72.

[78] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.



BIN WU received the Ph.D. degree in signal and information processing from the Beijing University of Posts and Telecommunications. He is currently a Lecturer with 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 received the bachelor's degree in cyberspace security from the Beijing University of Posts and Telecommunications (BUPT), where he is currently pursuing the master's degree with the School of Cyberspace Security. His research interests include network security, social engineering, and artificial intelligence security.



YANQING YANG received the B.S. and M.S. degrees from Xinjiang University, Urumqi, China, in 2006 and 2009, respectively. He is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications. His research interests include network attack detection and network security.

KANGFENG ZHENG received the Ph.D. degree in information and signal processing from the Beijing University of Posts and Telecommunications, in July 2006. He is currently an Associate Professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications. His research interests include network security and network data analysis.

XIUJUAN WANG received the Ph.D. degree in information and signal processing from the Beijing University of Posts and Telecommunications, in July 2006. She is currently an Instructor Lecturer with the College of Computer Sciences, Beijing University of Technology. Her research interests include information and signal processing, network security, and network coding.