BotCF: Improving the social bot detection performance by focusing on the community features

Feng Liu , Zhenyu Li , Chunfang Yang , Daofu Gong , Fenlin Liu , Rui Ma , Adrian G. Bors , *Senior Member, IEEE*

***Abstract*—Various malicious activities performed by social bots have brought a crisis of trust to online social networks. Existing social bot detection methods often overlook the significance of community structure features and effective fusion strategies for multimodal features. To counter these limitations, we propose BotCF, a novel social bot detection method that incorporates community features and utilizes cross-attention fusion for multimodal features. In BotCF, we extract community features using a community division algorithm based on deep autoencoder-like non-negative matrix factorization. These features capture the social interactions and relationships within the network, providing valuable insights for bot detection. Furthermore, we employ cross-attention fusion to integrate the features of the account's semantic content, properties, and community structure. This fusion strategy allows the model to learn the interdependencies between different modalities, leading to a more comprehensive representation of each account. Extensive experiments conducted on three publicly available benchmark datasets (Twibot20, Twibot22, and Cresci-2015) demonstrate the effectiveness of BotCF. Compared to state-of-the-art social bot detection models, BotCF achieves significant improvements in accuracy, with an average increase of 1.86%, 1.67%, and 0.47% on the respective datasets. The detection accuracy is boosted to 86.53%, 81.33%, and 98.21%, respectively.**

***Index Terms*—Social Bot Detection, Sybil Detection, Community Structure, Graph Convolutional Network.**

I. Introduction

SOCIAL bots are controlled by automated programs or APIs and perform various malicious activities on social networks. These accounts are called bot accounts [1–3], Sybils [4], or fake accounts [5]. The explosive use of online social networks in marketing, news, public relations, mass information activities, entertainment, and globally and nationally significant events has boosted the development of social bots. During the 2016 U.S. election, bots disrupted the election by posting a large number of tweets supporting specific political lines as well as tweets smearing competitors [6]. It was reported that 50% of the posts about Trump as a candidate were written by social bots [7]. One statistic found that bot accounts accounted for 15% of Twitter's active accounts in 2017 [8]. Social bots massively posted and amplified low-credibility messages about COVID-19 on social networks [9]. In 2020, a study showed that among 200 million tweets discussing coronavirus, 82% of the top 50 influential forwarders were bots, and 62% of the top 100 forwards were bots[1]. The purposeful behavior of malicious bots on social networks seriously endangers the trust relationship of users in online social networks. For example, in Mumbai, social bots spread rumors on social media that the vaccines were a plot by the government to sterilize Muslim children, which led to only 50% of those who were expected to be vaccinated actually got the vaccine [10].

Social platforms and researchers proposed a series of social bot detection methods to minimize the impact of malicious social bots, with early success. These detection methods can be grouped into two categories - account feature-based methods and graph structure-based methods. Existing feature-based detection methods for social bots use many hand-crafted features from different categories of information, such as profiles, content, networks, properties and train machine learning models to separate the bots from benign users based on the information extracted. However, many of the existing features are effectless when facing the manually aided created profile properties and scheduled activities of social bots generated by complex stochastic algorithms. For instance, the rapid development of deep forgery techniques allows social bots to have identical profile information as normal accounts and automatically establish social relationships with other accounts, interspersing small amounts of malicious information with many neutral ones, which is very different from the traditionally considered bot behavior [11]. The study on Twitter bots [12] indicates that current social bots can more delicately disguise themselves as normal accounts and work in concert to achieve certain specific purposes, such as spreading rumors, posting advertisements. Existing feature-based methods suffer from three major limitations: (1) Human limitations and biases may influence the design of the features. (2) Features are susceptible to the manipulation of the social bots. (3) The generalization ability of the model is limited by the wide variations in the properties of the feature extracted in various online social networks.

Regarding these challenges, several studies used the interactions of accounts in social networks to construct the social graphs which are then divided into cohesive subgraphs

Feng Liu is with the School of Artificial Intelligence, Jilin University, Changchun 130012, China and the School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450002, e-mail: zzuliufeng@163.com. Zhenyu Li, Chunfang Yang, Daofu Gong and Fenlin Liu are with Henan Key Laboratory of Cyberspace Situation Awareness, Zhengzhou Science and Technology Institute, China, e-mail: {li1989zhenyu,chunfangyang}@126.com, gongdf@aliyun.com, liufenlin@vip.sina.com. Rui Ma is with the School of Artificial Intelligence, Jilin University, Changchun 130012, China, e-mail: ruim@jlu.edu.cn. Adrian G. Bors is with the Department of Computer Science, University of York, UK, e-mail: adrian.bors@york.ac.uk Zhenyu Li and Chunfang Yang are the corresponding authors.

[1]Nearly half of the Twitter accounts discussing "Reopening America" may be bots: https://www.cmu.edu/news/stories/archives/2020/may/twitter-bot-campaign.html

using graph mining techniques [13–18]. This type of approach usually considers only leveraging the links of social bots in online social networks, but misses the automated cues embedded in text, time, and profile information. Therefore, these methods are unable to detect social bots that have successfully established enough attack edges (links) with normal users [19]. Faced with this challenge, Feng et al. [12] proposed a social bot detection method that combines property features and semantic features. However, they did not consider the impact of community features and the fusion methods on multimodal feature integration.

Following the idea of integrating multimodal features for social bot detection, we design a social bot detection method focusing on community features, namely BotCF. In the proposed model, the community features are obtained by the community division algorithm based on deep autoencoder-like non-negative matrix factorization. Besides the existing semantic and property features, we also consider community features, obtaining a social bot detection model that simultaneously considers the account's semantic, property and community structure information. Specifically, we first extract and vectorize the semantic features of the accounts' tweets, the property and community structure features of the accounts. Then, we utilize cross-attention fusion to integrate the features of these three modalities and embed them in a graph convolutional network (GCN). The representation vectors of the accounts that learned during the training of the GCN are ultimately used for the classification of the accounts.

Objectives: The objective of the study is to overcome two main challenges faced by the current social bot detection methods as follows:

- The detection model based on a single type of features can be easily fooled by bots. For example, building a sufficient number of attack edges can render a detection model that relies solely on graph structure features ineffective [19].
- The detection models often neglect the fact that social bots usually work in groups, resulting in sub-optimal detection performance.

To counter the issues of objectives, we propose a new social bot detection model that can leverage more types of features and achieve better performance on various benchmark datasets.

Contributions: The main contributions of the study are as follows:

- A GCN-based social bot detection model focusing on the community structure features is proposed.
- The cross-attention mechanism is employed to integrate multimodal features, and we validate its effectiveness through ablation experiments.
- Extensive experimental results show that the proposed model achieves better performance compared to the existing state-of-the-art models.

The rest of the paper is organized as in the following. In Section II we discuss the main approaches in bot detection. In Section III we provide the problem definition and in Section IV we describe the proposed methodology. In Section V we present the experimental results while these are discussed in Section VI. Section VII draws the conclusions of this study.

II. Related Work

The earliest work on social bot detection dates back to 2010 [20], honeypot traps were designed to detect social bots. Over time, the development of social bot detection technology has shown two main trends: single-account feature-based social bot detection and groups-based one. This section introduces the characteristics of these two categories of methods.

A. Single-account feature-based social bot detection

Early social bot detection methods were mainly based on the account properties feature extraction and processing, using traditional classifiers for classification. The work from [21] filters social bots by analyzing Twitter account profiles. Specifically, it designed sixteen-dimensional features, representing among others: screen name length, active days, the number of posted tweets, by analyzing account properties, tweet content, historical activity, and friend lists. Afterwards, it feeds these features into a random forest classifier to distinguish bots from authentic human users, which is one of the foundational works on social bot detection based on the individual account features. Many follow-up studies continue to mine more features from accounts to improve the detection accuracy of the model [22–25]. Some researchers, considering that social accounts should not be classified only as bots and non-bots due to the hijacking of human accounts in social networks, studied the differences between humans, bots and cyborgs in terms of tweets (number of tweets, time of posts) and account properties (external URL ratio, account reputation, etc.) [26]. This work laid down the idea of designing different classifiers for different types of bots. Cresci et al. [27] designed digital DNA, a string of characters that encodes the sequence of the accounts' action, to train different classifiers to detect different bots.

However, over time, bot operators gradually learned about classical bot detection features and managed to hide their bot characteristics to evade detection. Continuous changes in bot design, aiming to avoid detection, have been described in [11, 12, 22]. In response to this trend, researchers continue to exploit the individual account features. Yang et al. [22] mined 10 new features from the data, such as account clustering coefficients, two-way following ratio, and tweet similarity, to train classifiers against the evolution of bots. Beskow et al. [23] extracted differentiated account profile features (degree centrality, K-betweenness centrality, mean eigen centrality, etc.) and tweet features (mean/max mentions, number of languages, etc.) from the collected data and used random forest as the classifier. Subsequent research designed new features to combat the continuous evolution of bots and achieved good performance [28, 29]. But it should be noted that the chosen features depend on the specific properties of social platforms, which limits the generalization ability of these models.

To overcome the challenge of generalization ability and design generic social bot detection models, some researchers designed various classifiers for bots using different datasets and combined these classifiers into ensembles [30]. Botometer-v3 [31], a social bot detection system that incorporates 1700-dimensional features to improve generalization, was analyzed in a series of research works on social bot detection [31–33]; Some scholars used natural language processing (NLP) methods to extract semantic differences from account tweets to detect social bots. For example, the work by [34] proposed a long short-term memory network (LSTM) based model to extract content features and temporal features of tweets to distinguish between bots and people. Pre-training models in natural language processing are also applied in social bot detection [1, 12].

The confrontation between bot detectors and operators is a never-ending race. The properties of a single account are easy to forge or manipulate. To address this challenge, research has shifted toward group-based social bot detection methods.

B. Group-based social bot detection

The group-based social bot detection method utilizes the structural differences between the social graphs generated by humans and bots. The relationships that used to build the social graph are usually categorized as friend relationships [35], following/follower [2, 12], retweet/retweeted [36]. The detection mechanism is to use the homogeneity of social networks, in other words, the neighbor nodes of the bot tend to be other bots, while the neighbor nodes of a human tend to be humans [2, 13, 37]. A label-enhanced network integrates labels with social network and uses the defined badness score based on the random walk of nodes to distinguish the bots from humans [38]. Wang et al. [13] proposed paired Markov random field models to estimate the posterior probability of each user by loopy belief propagation while predicting the user's label based on the posterior probability. Moreover, they proposed a framework to unify random walk and loopy belief propagation in [14, 37] to counter the limitations of the method [38] that it cannot utilize the labels of bots and humans, meanwhile, avoiding the problems of the method [13] that it is not scalable and does not guarantee convergence. The study from [39] trained a local classifier to calculate the local trust scores of nodes and edges, and then the local trust scores used for prediction are propagated through the global network structure by a weighted random walk and loopy belief propagation mechanism.

However, the social bot detection algorithms based on the social graph usually consider only the local neighbor features of accounts, while ignoring their community structure features. Bot2vec [2] proposed by Pham et al. considers both macro-community structural features and micro-neighborhood features of accounts, but it ignores accounts' semantic and property information.These group-based social bot detection methods largely improve the generalization of the model and avoid manual feature engineering. However, this type of method only utilizes the link information between accounts, and its detection performance is greatly reduced when enough attack links are established between accounts [19].

With the rise of graph convolutional network (GCN) [40], it has been widely used in various occasions, such as for link prediction, node classification, community division. Researchers introduce GCN to detect social bots, because GCN can utilize the link information between accounts as well as lots of other information. Sun et al. [41] designed a GCN with trust mechanism. First, the method starts a short random walk from a known real node, and its walk probability is the trust score of the node. Then, it uses these trust scores as edge weights, and uses graph convolution operations to aggregate features from local graph neighborhoods onto a weighted graph for classification. Alhosseini et al. [42] proposed a GCN-based spam bot detection model which utilizes both account property features and neighborhood features. Following this direction, researchers designed a social bot detection model using the semantic features and property features of relational graph convolutional network (RGCN) [12]. First, it vectorizes the property features and semantic features of the accounts and then stitch the two types of vectors together. Then, the spliced semantic vectors are fed into a neural network model for training to detect social bots. This method achieves state-of-the-art results on homogeneous graph social bot detection.

III. Problem Definition

In this section, we give the key definitions used in this paper.

Definition 1 (Graph): Let $G = (V, E, T)$ be a directed graph, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of nodes. The social network is viewed as a graph, where each account is considered as a node. n denotes the number of nodes. E denotes the set of edges between nodes and the edges denote the interaction between nodes. T represents the edge type, which in this paper indicates whether the edge is an outgoing edge (Following) or an incoming edge (Follower).

Definition 2 (Social bot detection task): For a given social account U, we consider extracting its semantic features U_S, property features U_P and community features U_C, and learn a suitable social bot detection function f, $f(U_S, U_P, U_C) \to \hat{y}$. Train the model using account multi-dimensional features so that the predicted value $\hat{\boldsymbol{y}}$ is as close to the true value $\boldsymbol{y}$ as possible to maximize the detection accuracy.

IV. Proposed Approach

This section presents the framework of the proposed model which consists of four components: input, preprocessing, node representation, and output, as shown in Fig. 1. The various features, especially the community features, describing the social media accounts are first collected and considered as the input of the BotCF model. Then, the preprocessing module vectorizes these features and fusion these vectorized features to form the initial representation vector of the node, namely, social media account. The node representation module takes the initial feature vector and adjacency matrix of the node as the input to the GCN model. After the training of GCN, the final representation vector of the node is obtained and used for classification in the output module. The whole process is an end-to-end process. Each module is described specifically in the following subsections.

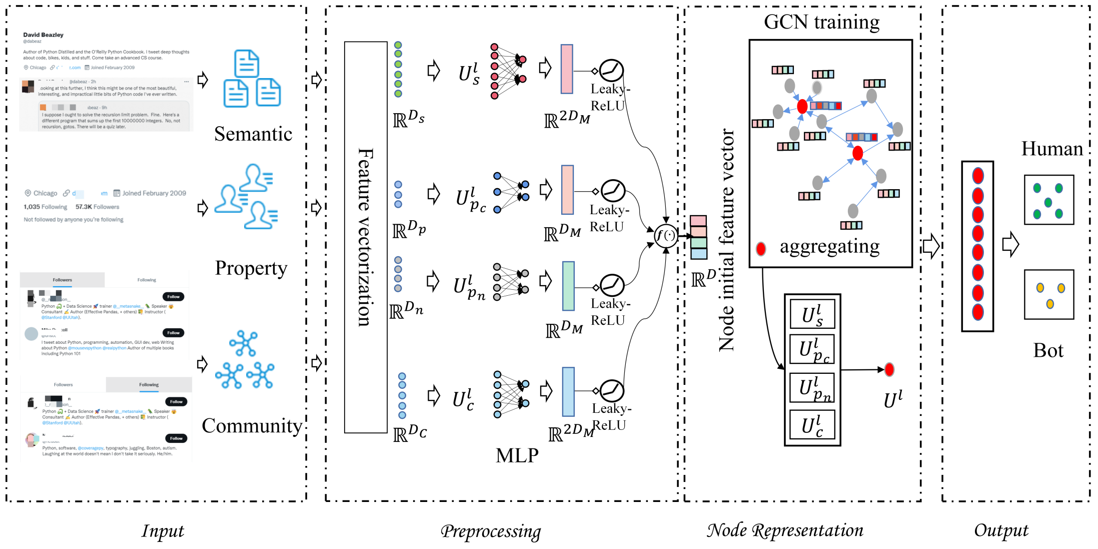


Fig. 1: The overall framework of the proposed social bot detection model.

A. Input Module

Under the normal situation, humans create their own accounts in social media, where they post information according to their wishes. Human accounts can be seen as a mapping of a person's identity in cyberspace, with multi-dimensional information being provided, reflecting his/her persona. Social bots are generated in batches by human manipulated programs attempting to mimic the property information and activities of human accounts in social networks. In order to find the flaws of the social bots, it is reasonable to utilize the social accounts' information for building the bot detector.

This paper proposes using account semantic features, property features, and community structure features to learn account representations. The semantic features are extracted from the account's descriptions and tweets. The account's profile, such as account ID, screen name, profile image, represent the property features. The social graph, where each edge represents a following or follower relationship between accounts, is used to extract community structure features.

B. Preprocessing Module

The preprocessing module uses multi-dimensional information to represent a set of accounts. First, the accounts' semantic features, numerical and categorical property features, and community structure features are encoded as vectors. Then, these vectors are scaled using a Multi-layer Perceptron (MLP) to ensure consistent feature dimensionality. The feature vectors are fused through a cross-attention mechanism to form the initial representation vector of each account, which serves as input to the subsequent module. In the following, we explain how the three types of features are extracted and preprocessed.

1) Semantic Features: Tweets can largely reflect the characteristics of the accounts and are widely used in existing bot detection methods. We utilize the pre-trained RoBERTa model from the Hugging Face library, specifically version 3.0.2[2], to encode account semantic information. This version provides a fully trained model, including pre-trained weights and accessible API interfaces for easy integration.

The semantic feature vector U^l_S for a given account l consists of two components: the account description semantic vector $U^l_{S_d}$ and the tweet semantic vector $U^l_{S_t}$. The account description is a short text set by Twitter users to briefly introduce themselves.

First, it uses RoBERTa to encode each word in the account description, obtaining a sequence of representation vectors for the l-th account description, $\{\boldsymbol{E}^l_{d_i}\}^{L_l}_{i=1}$,

$$\{\boldsymbol{E}^l_{d_i}\}^{L_l}_{i=1} = RoBERTa(\{d_i\}^{L_l}_{i=1}), \tag{1}$$

where $d_i \in \mathbb{R}^{D_R \times 1}$ and $\{d_i\}^{L_l}_{i=1}$ is the l-th account description that consists of L_l words and i represents the index of the word in the description. D_R is the embedding dimension which is predefined in $RoBERTa$.

Then, the representation vector $\boldsymbol{U}^l_{S_d}$ of the account description is as follows,

$$\boldsymbol{U}^l_{S_d} = \sigma(W_d \cdot \frac{1}{L_l}\sum_{i=1}^{L_l} \boldsymbol{E}^l_{d_i} + b_d), \tag{2}$$

where $\boldsymbol{E}^l_{d_i}$ is derived according to Eq. (1), W_d and b_d are learnable parameters. $\boldsymbol{U}^l_{S_d} \in \mathbb{R}^{D_M \times 1}$, D_M is the dimension of the output vector of the MLP. σ is the activation function. Here, Leaky-ReLU [43] is used as the activation function. Compared with Sigmoid/tanh, ReLU has faster convergence and calculation speed, and Leaky-ReLU can further solve the parameter updating problem caused by negative input of ReLU.

Meanwhile, it uses similar method to encode each word in the tweet, obtaining the representation vectors of the tweet, $\{\boldsymbol{E}_{t^i_j}\}^{H_j}_{i=1}$,

$$\{\boldsymbol{E}_{w^i_j}\}^{H_j}_{i=1} = RoBERTa(\{w^i_j\}^{H_j}_{i=1}), \tag{3}$$

[2]https://huggingface.co/transformers/v3.0.2/main_classes/pipelines.html

where $w_j^i \in \mathbb{R}^{D_R \times 1}$ and $\{w_j^i\}_{i=1}^{H_j}$ is the i-th word of the j-th tweet, and the tweet length is H.

The representation vector $\boldsymbol{E}_{t_j}$ for each tweet of an account is as follows,

$$\boldsymbol{E}_{t_j} = \frac{1}{H_j} \sum_{i=1}^{H_j} \boldsymbol{E}_{w_j^i}, \tag{4}$$

where $\boldsymbol{E}_{t_j} \in \mathbb{R}^{D_R \times 1}$ and $\boldsymbol{E}_{w_j^i}$ is derived from Eq. (3).

The tweet semantic vector $\boldsymbol{U}_{S_t}^l$ of the l-th account is the average of the representation vectors of all its tweets, as follows,

$$\boldsymbol{U}_{S_t}^l = \sigma(W_t \cdot \frac{1}{M_l} \sum_{j=1}^{M_l} \boldsymbol{E}_{t_j} + b_t), \tag{5}$$

where $\boldsymbol{U}_{S_t}^l \in \mathbb{R}^{D_M \times 1}$ and M_l is the number of tweets from the l-th account. W_t and b_t are learnable parameters.

Combing the two parts obtained above, we can get the semantic feature vector of the l-th account, namely, $\boldsymbol{U}_S^l = [\mathrm{U}_{S_d}^l, \boldsymbol{U}_{S_t}^l], \boldsymbol{U}_S^l \in \mathbb{R}^{2D_M \times 1}$.

2) Property Features: The properties of accounts are divided into statistical features (e.g., number of followers, likes, retweets) and category features (e.g., whether the account is authenticated, whether it uses default profile information, whether it displays location information).

All the property features used for account representation are shown in Table I, including two novel features proposed in this paper, retweets and semantic vector distance. Retweets feature is calculated as the number of retweet tweets ('RT @') in the latest 200 tweets per account (in the Twibot-20 dataset and the Twibot-22 dataset, each account has 200 tweets). Bots usually work collaboratively in social networks, so they would retweet more tweets than humans to amplify their influence.

The semantic vector distance ("distance") corresponding to each account reflects the variety of the content of the account's tweets. Since the social bots are controlled by specific software, the diversity of the content of their tweets is rather limited, resulting in the fact that their semantic vector distances would be probably smaller than those of humans. The average semantic vector distance can reflect the semantic discreteness of each account's tweets. For example, tweets from human accounts are more diverse, so they have a greater average semantic vector distance. Regarding the l-th account, the distance between the semantic vector of k-th tweet and the averaged semantic vector of all tweets is first calculated using the Manhattan distance as follows,

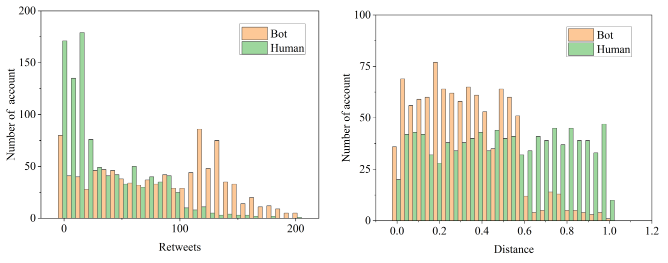
$$sv\_dis^k(U_{S_t{}^k}^l, U_{S_t}^l) = \sum_{h=1}^{n} |x_h^k - \bar{x}_h|, \tag{6}$$

where $U_{S_t{}^k}^l = (x_1^k, x_2^k, \cdots, x_n^k)$ is the semantic vector of the k-th tweet for the l-th account, $U_{S_t}^l = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ is the averaged semantic vector of the l-th account. Then, the semantic vector distance corresponding to the l-th account, $SV\_DIS(U^l)$, is the average of the semantic vector distances of all its tweets, namely,

$$SV\_DIS(U^l) = \frac{1}{M} \sum_{k=1}^{M} (sv\_dis^k(U_{S_t{}^k}^l, U_{S_t}^l)), \tag{7}$$

where M represents the total number of tweets for an account.

In order to justify the two proposed property features, we visualize the differences between the features from the human account and bot account. We randomly select 1000 human accounts and 1000 bot ones from the TwiBot-20 dataset. Fig.2a and Fig.2b show the distribution of the number of retweet and semantic vector distance of the selected social accounts, respectively. From Fig.2a we can see that the retweets of most of the human account is less than 100, but quite a lot bot account retweets more than that. In addition, it can be observed from Fig.2b that the semantic vector distance of the bot account is usually less than 0.6, while, that of the human account is much more uniformly distributed. So it is reasonable to use these two features as part of the property features for the account representation.



(a) Retweets (b) Semantic vector distance

Fig. 2: Validity evaluation of the proposed property features on the TwiBot-20 dataset.

With respect of vectorization of the property features, we use Z-Score normalization for the numerical features and One-hot encoding for the category features.

On one hand, in the case of numerical features, we first normalize the i-th numerical feature of an account's properties, p_n^i, as follows,

$$\hat{\boldsymbol{p}}_n^i = \frac{p_n^i - mean(p_n^i)}{std(p_n^i)}, \tag{8}$$

where $mean(\cdot)$ is the average operation and $std(\cdot)$ is the standard deviation operation.

The representation vector of the numerical property features of the l-th account, $\boldsymbol{U}_{P_n}^l$, can be derived from Eq.(9).

$$\boldsymbol{U}_{P_n}^l = \sigma(W_{P_n} \cdot [\hat{p}_n^1; \hat{p}_n^2; \cdots; \hat{p}_n^r] + b_{P_n}), \tag{9}$$

where $\boldsymbol{U}_{P_n}^l \in \mathbb{R}^{D_M \times 1}$, and W_{P_n} and b_{P_n} are learnable parameters. r is the total number of the numerical feature.

On the other hand, in the case of category features, the i-th category feature in the properties of an account, $\boldsymbol{p}_c^i$, is obtained as follows,

$$\boldsymbol{p}_c^i = \begin{cases} 1, & \boldsymbol{p}_c^i = True \\ 0, & \boldsymbol{p}_c^i = False \end{cases} \tag{10}$$

Then, the representation vector of the category features in the property features of the l-th account, $\boldsymbol{U}_{P_c}^l$, is derived by Eq.(11),

$$\boldsymbol{U}_{P_c}^l = \sigma(W_{P_c} \cdot [p_c^1; p_c^2; \cdots; p_c^t] + b_{P_c}), \tag{11}$$

where $\boldsymbol{U}_{P_c}^l \in \mathbb{R}^{D_M \times 1}$, and W_{P_c} and b_{P_c} are learnable parameters. t is the total number of the category feature.

Finally, the property feature vector U_P^l for the l-th account is combined as $\boldsymbol{U}_P^l = [\boldsymbol{U}_{P_n}^l, \boldsymbol{U}_{P_c}^l], \boldsymbol{U}_P^l \in \mathbb{R}^{2D_M \times 1}$.

3) Community Structure Feature: It was studied that bot accounts in online social networks work in cooperation to achieve certain purposes [12], indicating the existence of community aggregation of bot accounts. Inspired by this, besides semantic and property features, we consider using the community structure features as well for the account representation. The community structure features of accounts are obtained using the community division algorithm based on deep autoencoder-like non-negative matrix factorization (DANMF) [44]. The DANMF community detection algorithm can achieve good results in non-overlapping communities and is also applicable to overlapping communities.

The DANMF social community division algorithm uses encoders and decoders to optimize the probability of nodes belonging to each community and it provides a probability vector of each node belonging to each community. Specifically, it decomposes the adjacency matrix into $p+1$ non-negative factor matrices, see Eq. (12):

$$\boldsymbol{A} \approx \boldsymbol{M}_1 \boldsymbol{M}_2 \cdots \boldsymbol{M}_p \boldsymbol{H}_p, \tag{12}$$

where, $\boldsymbol{A}$ represents the adjacency matrix of the graph G, which defines the relationships between nodes in the graph. In the experimental process, both the adjacency matrix of the graph generated by the training set, $\boldsymbol{A}_{train}$, and that of the testing set, $\boldsymbol{A}_{test}$, are decomposed in the same way. A lot of studies have shown that a 2-layer GCN (or shallow GCN) yields the best performance. However, the receptive field of a 2-layer GCN is limited to 2-hop neighbors, allowing it to capture only local structural features and preventing it from extracting features from a more macro-level perspective. Community detection algorithms, on the other hand, can extract features from a more global community perspective. In other words, using $\boldsymbol{H}$ enables the GCN to capture macro-structural features. Community detection algorithms, on the other hand, can extract features from a more global community perspective. Therefore, using $\boldsymbol{H}$ enables the GCN to capture macro-structural features. In the experimental process, we incorporate the community structural features $\boldsymbol{H}$ extracted by the detection algorithms into U_{init} as one of the three feature types, compensating for the shallow GCN's limitation. $\boldsymbol{M}_i$ is the i-th layer's mapping matrix and each column of $\boldsymbol{M}$ denotes the description of a community. $\boldsymbol{H}_p$ is the p-th layer's community member matrix and each column of $\boldsymbol{H}$ represents the association relationship of a node to different communities. The DANMF [44] algorithm suggests setting $p = 2$ for optimal performance. Therefore, in our experiments, we have set $p = 2$.

The community representation vector of the l-th account, $\boldsymbol{U}_C^l$, is obtained after the MLP transformation as follows,

$$\boldsymbol{U}_C^l = \sigma(W_c \cdot \boldsymbol{H}_{\cdot l} + b_c), \tag{13}$$

where $\boldsymbol{U}_c^l \in \mathbb{R}^{2D_M \times 1}$ and $\boldsymbol{H}_{\cdot l}$ represents the association relationship of the l-th node to different communities. W_c and b_c are learnable parameters.

Ultimately, the initial feature representation vector of account l can be expressed as $U_{init}^l = fusion(\boldsymbol{U}_S^l, \boldsymbol{U}_P^l, \boldsymbol{U}_C^l)$, $\boldsymbol{U}_{init}^l \in \mathbb{R}^{6D_M \times 1}$. $fusion(\cdot)$ is the fusion function. We compared four fusion methods (concatenation, summation, self-attention and cross- attention), and their specific settings are as Section V-D.

Concatenation and summation are the most commonly used feature fusion methods in the early days, which are characterized by their simplicity, but cannot distinguish the degree of contribution of different features to the downstream task and cannot capture the interactions between different modal features. With the rise of Transformer, more and more attention-based fusion methods have been focused on. Self-attention distinguishes the contribution of different features to the downstream task by weighting the features, and cross-attention can model the interactions between different modal features. Specific ablation experiments have also demonstrated the superiority of the cross-attention fusion approach for multimodal social bot detection tasks (see Section V-D).

C. Node Representation Module

This subsection describes the process of obtaining the final representation vector of nodes. The initial representation vector of each node is the output of the preprocessing module. Then, the directed graph is constructed using the following-follower relationship, and the node's neighbor information and initial features are aggregated using the GCN message passing mechanism. The final representation vectors are the output of the GCN model. The detailed learning process for node representation vectors is as follows.

First, the initial hidden vector of the i-th node in the graph, $l_i^{(0)}$, can be expressed as Eq. (14),

$$l_i^{(0)} = \sigma(W_1 \cdot U_{init}^i + b_1), \tag{14}$$

where $l_i^{(0)} \in \mathbb{R}^{D \times 1}$ and D is the dimension of the output embedding vector of the GCN model. U_{init}^i is the i-th node's initial feature representation vector. W_1 and b_1 are learnable parameters.

Then, after applying RGCN to the constructed directed graph, the aggregation of k-th layer of RGCN can be formulated as Eq. (15),

$$l_i^{(k+1)} = \sigma\left(\sum_{r \in [N_f, N_t]} \sum_{j \in N_r(i)} \frac{1}{N_r(i)} {W_r}^k {l_j}^{(k)} + {W_0}^k {l_i}^{(k)}\right), \tag{15}$$

where r represents the relationship type (following or follower). $N_r(i)$ denotes the set of r-relational type neighbors of node i. N_f is following and N_t is follower. W_0^k and W_r^k are the self-1oop projection matrix and the relational projection matrix, respectively, both of which are learnable parameters. k is the index of the layer of GCN.

Meanwhile, the node representation vector of the i-th node produced by the k-th layer of the GCN, U_k^i, is obtained after the MLP processing as follows:

$$U_k^i = \sigma(W_k l_i^{(k)} + b_k), \tag{16}$$

TABLE I: Property features used in our model.

Feature Name	Description
followers	The number of followers.
followings	The number of accounts that the account follows.
favorites	The number of favorites or likes a account receives.
statuses	The number of statuses a account posts.
active days	The number of days from the account's registration to present.
screen name length	The length of the account's current screen name
protected	Whether the account is currently protected.
is translator	Whether there is a translator or not.
is translation enabled	Whether translation is available or not.
geo enabled	Whether the account is geo enabled or not.
profile background tile	Whether the account uses a background tile.
profile background image	Whether the account uses a background image.
has extended profile	Whether the account has extended profile or not.
default profile	Whether the account uses a default profile.
default profile image	Whether the account uses a default profile image or not.
verified	Whether the account is verified or not.
retweets	The number of retweets in the latest 200 tweets for this account.
semantic vector distance ("distance")	The distance between the semantic vector of each tweet of the account and the tweet vector of the account

where W_k and b_k are learnable parameters. σ is the Leaky-ReLU activation function.

The final node representation vector of the i-th node, U^i, is the output of the GCN network after multiple iterations of aggregation (Eq. (15)) and MLP processing (Eq. (16)).

D. Output Module

The node representation vector U^i is obtained based on the processing described in the previous subsection, and our model classifies node i as a social bot or human by the softmax layer (see Eq. (17)).

$$\hat{y}_i = softmax(W_o U^i + b_o), \tag{17}$$

where $\hat{y}$ is the prediction of the label, W_o and b_o are the learnable parameters of the softmax layer.

The loss function used in the training of the model is the cross-entropy loss function. (see Eq. (18)).

$$Loss = -\sum_{i\in U}[y_i log(\hat{y}_i) + (1-y_i)log(1-\hat{y}_i)] + \lambda \sum_{\omega\in\Omega}\omega^2, \tag{18}$$

where U stands for the set of labeled users, $\hat{y}$ is the prediction of the label and y is the ground truth. Ω represents the set of all learnable parameters in the model.

Thus, the final optimization objective function can be written as:

$$\min_{\mathbf{W},\mathbf{b}} \mathcal{L} = -\sum_{i\in U}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] + \lambda \sum_{\omega\in\Omega}\omega^2 \tag{19}$$

Algorithm 1: BotCF

```
Input: Social network: $G = (V, E, T)$,
       Semantic information: $U_S = \{U_{S_d}, U_{S_t}\}$,
       Property information: $U_P = \{U_{P_n}, U_{P_c}\}$
Output: Node representation vector: $\mathbf{U}$
1  initialization $\mathbf{U_{init}}$, $\mathbf{U_S}$, $\mathbf{U_P}$,$\mathbf{U_C}$ ;
2  //Pre-processed
3  for $U^l \in V$ do
4      $\mathbf{U^l}_S \longleftarrow RoBERTa(U^l{}_S)$;
5      $\mathbf{U^l}_{P_n} \longleftarrow Z\text{-}Score\ Normalization(U^l{}_{P_n})$;
6      $\mathbf{U^l}_{P_c} \longleftarrow One\text{-}hot\ Encoding(U^l{}_{P_c})$;
7      $\mathbf{U^l}_C \longleftarrow DANMF(G)$;
8      $\mathbf{U^l}_{init} = fusion((\mathbf{U^l}_S, \mathbf{U^l}_{P_n}, \mathbf{U^l}_{P_c}, \mathbf{U^l}_C))$;
9  //Node Representation
10 while $RGCN$ has not converged do
11     for $U^l \in V$ do
12         $Loss(\mathbf{U^l}) \longleftarrow RGCN(\mathbf{U^l}_{init}, G)$
13     Update: $\mathbf{U} \longleftarrow BackPropagate(Loss(\mathbf{U^l}))$;
14 return U;
```

where, $\mathbf{W}$ and $\mathbf{b}$ represent the set of learnable parameters, which are updated iteratively through multiple rounds of training.

The proposed model is named as BotCF and its pseudo-code is given in Algorithm 1.

V. Experiments

In this section, we perform extensive experiments on three benchmark datasets to validate the performance of the proposed model. All experiments are conducted on a server with Intel (R) Xeon (R) Gold 6234 CPU (4×8 cores, 128 GB, 3.3 GHz) and RTX 3090 (2×24 GB) GPU running Ubuntu 20.04 (64-bit).

A. Datasets

The experiments are based on three different publicly available datasets, namely, the Cresci-2015 dataset [45], the TwiBot-20 dataset [46] and the TwiBot-22 dataset [47]. The Cresci-2015 dataset is a social bot detection dataset introduced in 2015 by Cresci et al. [45]. It contains five sub-datasets, named E13, TFP, INT, FSF, and TWT. A total of 574 accounts were collected in the TFP (The Fake Project) dataset, of which 469 were verified as human accounts. The E13 (Election 2013) dataset was validated by social scientists to identify 1,481 human accounts. FSF, TWT, and INT are bot datasets purchased from three different websites (Intertwitter, Fastfollowerz, and Twittertechnology, respectively), and were crawled again after purchase, getting 1169, 845, and 1337 bot accounts, respectively. The TwiBot-20 dataset is a social bot dataset made public by Feng et al. [46] in 2020, which includes 229,573 Twitter users, 33,488,192 tweets, 8,723,736 user property items and 455,958 following relationships. The TwiBot-22 dataset is a larger social bot dataset made public by Feng et al. [47] in 2022, which includes 1,000,000 Twitter users (human: 860,057, bot: 139,943), 86,764,167 tweets and 170,185,937 following relationships. An overview of the datasets is presented in Table II.

TABLE II: Overview of the benchmark dataset.

Datasets	Account Count	Bot Count	Human Count
Cresci-2015 [38]	5301	3351	1950
TwiBot-20 [39]	229573	5273	6589
TwiBot-22 [40]	1,000,000	139,943	860,057

B. Baseline methods

In this section, we give a brief introduction of the baseline bot detection models compared with our model.

Miller et al. [33]: Miller et al. extract 107 features from the tweets and property information of accounts and utilizes an improved stream clustering algorithm to identify Twitter bots.

Deepwalk [48]: Deepwalk is considered to be the first method combining graph embedding with word embedding, which uses unbiased random walk to learn the neighbor information of nodes. It is widely used for graph node classification and other tasks.

Node2vec [49]: Node2vec is an improved version based on Deepwalk. It uses a depth-first search strategy (DFS) and a breadth-first search strategy (BFS) to protect the structural equivalence and homogeneity of the network nodes. Pham et al. [2] used Node2vec combined with traditional machine learning models for social bot detection.

GCN [40]: GCN is a graph representation model proposed by Kipf et al. that can efficiently utilize graph neighbor structure data. It has been widely used for tasks such as node classification, link prediction.

GAT [50]: GAT model introduces an attention mechanism that considers different weights when aggregating neighboring nodes. Similar to GCN, it is also widely used for downstream tasks such as link prediction, node classification and graph clustering.

Dropedge [51]: Dropedge is a method proposed by Rong et al., which mitigates the oversmoothing and overfitting problems that exist in deep neural networks by randomly deleting edges from the original graph during model training, and is often used for node classification tasks. We chose the combination of JK-NET [52] + Dropedge for our specific experiments.

Bot2vec [2]: Bot2vec is an improved social bot detection algorithm based on Node2vec proposed by Pham et al. It combines a community detection algorithm and a graph representation learning algorithm for social bot detection.

SATAR [1]: SATAR is a self-supervised Twitter account representation model combining account semantic information, property information and neighbor information proposed by Feng et al. It achieved very good results in the task of detecting novel bots.

BotRGCN [12]: BotRGCN is an RGCN-based social bot detection model proposed by Feng et al. It embeds account semantic information and property information into RGCN for social bot detection.

C. Implementation details

We conducted the experiments based on the source code provided by the authors. For model-specific parameters, we used the default configuration of the code, and we tried our best to ensure that the common parameters have the same configuration. The parameter configuration of all models in the experiments is shown in Table III. The source code for these models can be found in the original paper as well as in TwiBot-22[3].

D. Experimental results

Comparative Experiments: To evaluate the model performance, we use 70% of the data in the dataset as the training set, 20% of the data as the validation set, and the remaining 10% as the testing set. Accuracy, F1-Score and Precision are used as evaluation metrics and experiments are conducted on three benchmark datasets. The experimental results are shown in Table IV, each result is the average of three repeated experiments. where the best results are in bold.

From the results shown in Table IV, it is evident that on the latest datasets, TwiBot-20 and TwiBot-22, the performance of GCN-based social bot detection models (GCN, GAT, SATAR, BotRGCN, and our model BotCF) improves after incorporating features from other dimensions. The model performance shows improvements compared with the model only using the property features alone (Miller et. al [33]) or graph structure alone (Deepwalk [48], Node2vec [49] and

[3]https://github.com/LuoUndergradXJTU/TwiBot-22

TABLE III: Overview of models' parameter configuration.

Parameter	Miller	Deepwalk	Node2vec	Bot2vec	GCN	GAT	Dropedge	SATAR	BotRGCN	Ours
Network Layers	-	-	-	-	2	2	4	1	2	2
Dropout value	-	-	-	-	0.3	0.3	0.3	0.6	0.3	0.3
Embedding size	-	128	128	128	128	128	128	128	128	128
Learning rate	-	-	-	-	0.001	0.001	0.001	0.01	0.001	0.001
Weight decay	-	-	-	-	0.005	0.005	0.005	0	0.005	0.005
Optimizer	-	-	-	-	AdamW	AdamW	AdamW	SGD	AdamW	AdamW
Epochs	-	-	-	-	100	100	100	100	100	100
Number_Cluster	30	-	-	-	-	-	-	-	-	-
Window Size	-	7	7	7	-	-	-	-	-	-
Negative Sampling	-	5	5	5	-	-	-	-	-	-
Walk Length	-	30	30	30	-	-	-	-	-	-
Number of walks	-	20	20	20	-	-	-	-	-	-
Return Parameter	-	-	1	1	-	-	-	-	-	-
In–out Parameter	-	-	1	1	-	-	-	-	-	-

TABLE IV: Performance comparison of multiple social bot detection models on three benchmark datasets.

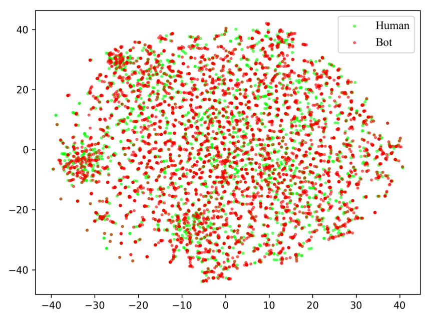
	Cresci-2015			Twitbot-20			TwiBot-22		
Model	Acc	F1	Pre	Acc	F1	Pre	Acc	F1	Pre
Miller	0.7549	0.8372	0.7213	0.6447	0.7472	0.6073	0.3037	0.4529	0.2946
Deepwalk	0.9358	0.9473	0.8871	0.5631	0.6113	0.5327	0.5187	0.3794	0.4917
Node2vec	0.9705	0.9734	0.9321	0.6066	0.6605	0.5923	0.5711	0.3927	0.5597
Bot2vec	0.9774	0.9801	0.9462	0.6328	0.7147	0.6318	0.5914	0.4108	0.5726
GCN	0.9537	0.9617	0.9459	0.7973	0.8047	0.7659	0.7239	0.4480	0.7119
GAT	0.9617	0.9714	0.9488	0.8348	0.8521	0.8131	0.7836	0.5586	0.7223
Dropedge	0.9643	0.9737	0.9508	0.8371	0.8574	0.8133	0.7842	0.5587	0.7281
SATAR	0.9631	0.9748	0.9497	0.8402	0.8607	0.8150	0.7871	0.5710	0.7407
BotRGCN	0.9652	0.9770	0.9551	0.8467	0.8707	0.8379	0.7966	0.5750	0.7481
Ours	**0.9821** (0.0047↑)	**0.9883** (0.0082↑)	**0.9679** (0.0128↑)	**0.8653** (0.0186↑)	**0.8835** (0.0128↑)	**0.8567** (0.0188↑)	**0.8133** (0.0167↑)	**0.5827** (0.0077↑)	**0.7618** (0.0137↑)

Bot2vec [2]), indicating that property features and structural features can complement each other. Compared with the GCN-based social bot detection models (GCN [40], GAT [50], SATAR [1], and BotRGCN [12]), Node2vec and Bot2vec perform better on the Cresci-2015 dataset, which may be because our experiments on the Cresci-2015 dataset only use the structural features of the account. It shows that Node2vec and Bot2vec are better at extracting the structural features of nodes, which may be because they can not only extract the neighbor features of nodes, but also extract the structural features of nodes [2, 53].Our model BotCF also performs well on the Cresci-2015 dataset, reflecting the fact that embedding community structure information can compensate for the loss of some structural features when GCN extracts node structure information. On the Cresci-2015 dataset, all models achieved good results, while on the TwiBot-20 or TwiBot-22 datasets, the performance of each model decreased more significantly, which may be due to the fact that after five years (both TwiBot-20 and TwiBot-22 are 2020 datasets) of development, the bots have evolved to be more indistinguishable from real users (some studies have shown that bots are constantly evolving [11, 12]). In general, embedding account property features, semantic features, and community structure features can improve the performance of the model. The experimental results on three benchmark datasets validate the accuracy, generalization and scalability of our model detection.

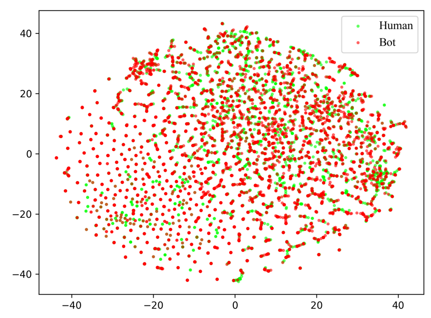
We also consider the t-SNE 2D visualization technique to visualize the embedding vectors and the corresponding homogeneity score obtained by each model on the TwiBot-20 dataset, and the results are illustrated in Fig. 4 and Fig. 3. The t-SNE visualization results can reflect the quality of model training to a certain extent [1, 2]. A higher homogeneity score means the samples are better clustered. It can be observed from Fig. 4 and Fig. 3 that our model achieves the highest homogeneity score and that the embedding vector obtained from our model training is more beneficial for the social bot detection task.

Model Selection Study: We study the impact when using different GNN training models, embedding different dimension features, and embedding community structure features on the performance of the proposed model. Taking TwiBot-20 [46] as the experiment dataset, GCN [40], GAT [50], GraphSAGE [54], and RGCN [55] are considered as the GNN training models in the node representation model, respectively. Fig. 5 shows the detection performance of the models when considering the GNN training module with various models when the number of layers is 2. The results show that RGCN works better on TwiBot-20, which is probably because RGCN is more powerful in handling heterogeneous relationships. GAT and GraphSAGE have similar performance, but both work better than using the original GCN. This may be due to the fact that both GraphSAGE and GAT are improved versions of GCN. They improve the aggregation of nodes and introduce the attention mechanism, respectively.

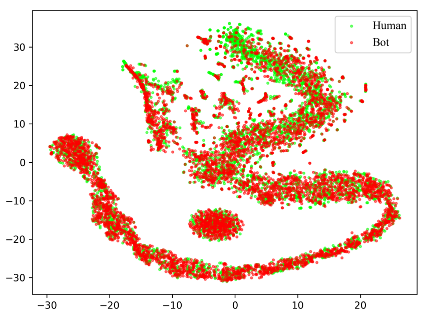
Fig. 6 shows the changing trend of detection performance of our model, BotRGCN, GCN, GAT and DropEdge when the



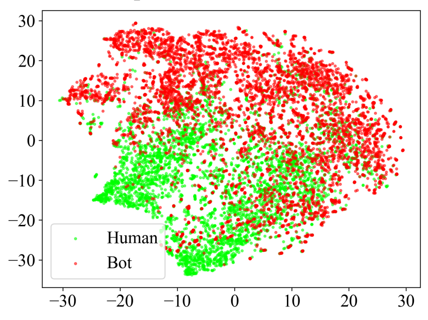
(a) Deepwalk (1.129×10^{-3})



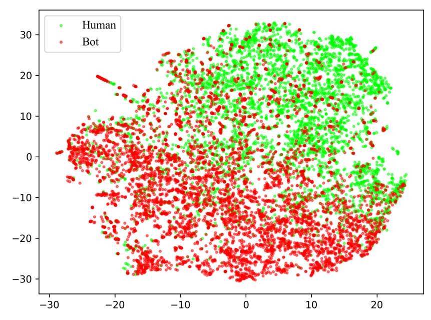
(b) Node2vec (2.559×10^{-3})



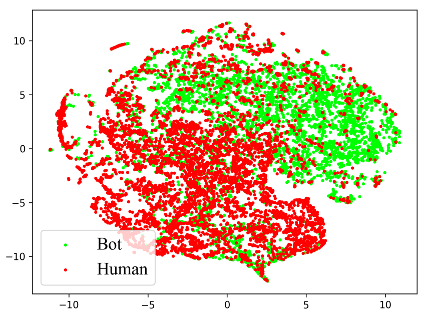
(c) Bot2vec (2.371×10^{-2})



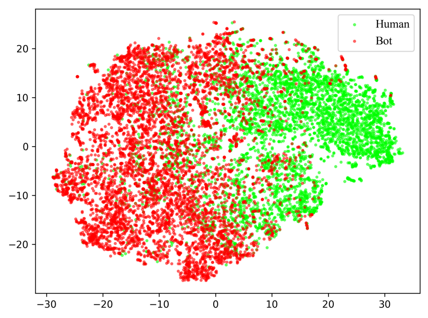
(d) GCN (1.609×10^{-1})



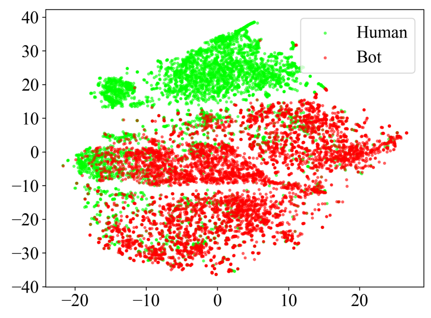
(e) GAT (1.851×10^{-1})



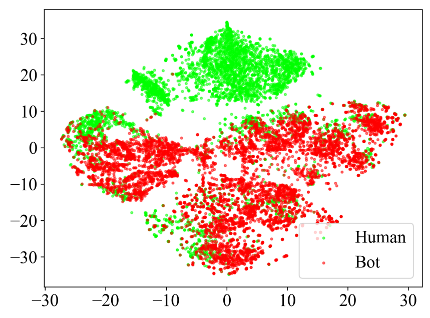
(f) Dropedge (1.937×10^{-1})



(g) SATAR (1.973×10^{-1})



(h) BotRGCN (2.356×10^{-1})



(i) Ours (3.812×10^{-1})

Fig. 3: Visualization of human-bot user representations of the TwiBot-20 dataset by various models via t-SNE 2D projections and the corresponding homogeneity score.

number of network layers increases. In Fig. 6a, Fig. 6b and Fig. 6c, the X-axis represents the number of network layers, while Y-axis represents the values of Accuracy, F1-Score, and Precision, respectively. As the number of layers increases, the overall performance of all models tends to increase initially and afterwards decrease. When the number of layers is equal to five, the performance of the BotRGCN and our model decreases significantly. To explain this phenomenon, we refer to [56] that the propagation operation in GCNs is essentially a Laplace smoothing, and thus the propagation operation makes the node embedding in deep GCNs indistinguishable, leading to a degradation of model performance. From Fig. 6, it can be observed that with the increase in the number of GCN layers, the overall performance decline of Dropedge is gradual. Our method shows a slightly faster decline in performance with the increase in layers compared to it. However, overall, after incorporating community structure features into our model, there is indeed a mitigation in the performance decline trend when compared to baselines like BotRGCN. Looking from another angle, performance degradation is mitigated by the proposed model compared to BotRGCN, which indicates that embedding community structure features may alleviate the over-smoothing problem to some extent.

Ablation experiment on features: In the following we test how the proposed features can work with novel bot. We investigate the effects of various combinations of the features, including neighbor features (N), semantic features (S), property features (P) and community structure features (C), on the detection performance of social bot. For example, the RGCN model that only uses the neighbor feature is denoted as "N". "N+P " represents the scenario using both neighbor feature and property feature for the training of the classifier. The detection performance is consistently evaluated by Accuracy, F1-Score and Precision on the TwiBot-20 dataset. We consider two layers in the network structure.

Most importantly, the best detection performance is

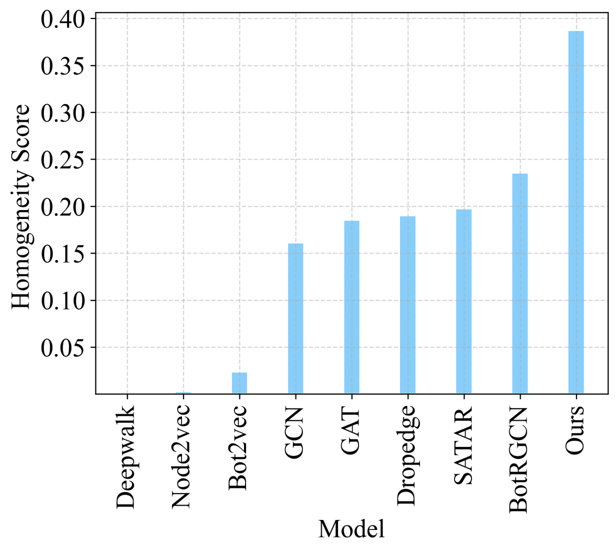


Fig. 4: Overview of homogeneity score obtained by various social bot detection models.

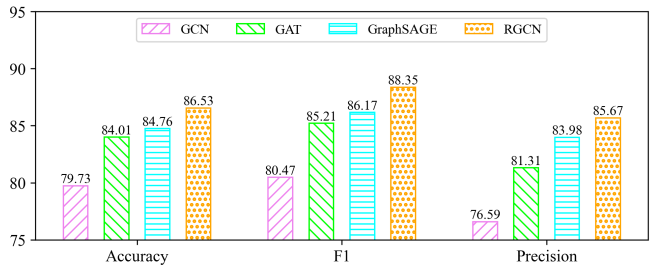


Fig. 5: Performance (%) comparison of social bot detection using various GNN training models on the Twibot-20 dataset.

achieved by "N+S+P+C", which validates that all three types of features are necessary for social bot detection. More specifically, when comparing using "N" with "N+S", "N+P", and "N+C" in Fig. 7, it can be seen that the semantic, property and community structure features can complement the neighbor feature in the social bot detection task. "N+P" achieves better performance than "N+S" and "N+C", indicating that the property features are more beneficial than the semantic features and community structure ones. This is because some of the property features have very strong distinguishability, for instance, for identifying whether the account is verified or not. Furthermore, by comparing ("N", "N+C"), ("N+P", "N+P+C") and ("N+S","N+S+C"), respectively, the detection performance is improved after adding the community structure feature. This observation indicates that the community structure feature is necessary and adding community structure features can compensate for the loss of some structural features extracted by a shallow GCN model.

In addition, we also conducted community structure feature ablation experiments for each GNN model, and the specific results are shown in Table V. Each result is the average of three repeated experiments. We added property features and semantic features to the original model, after which we added community structure features to observe the changes in Accuracy across models. From Table V, we can see that the accuracy of each base model is improved after adding community structure features. Among them, our base model is based on BotRGCN with the addition of the cross-attention module. Considering the good performance of property features in the previous feature ablation experiments, in the cross-attention fusion part of our method, in the case without community feature, we set Q and K as the semantic feature matrix, and V as the property feature matrix. However, in the case when community structure features are included, we set Q as the community structure feature matrix, K as the community structure feature matrix, and V as the property feature matrix.

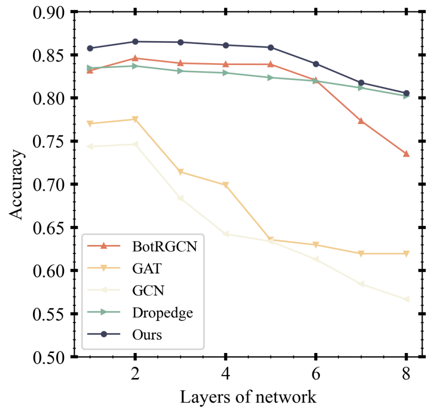
Ablation experiment on fusion: To explore the impact of feature fusion methods on the detection accuracy, we compared four different approaches: concatenation (Cat), summation (Sum), self-attention-based (SA) fusion and cross-attention-based (CA) fusion. Specifically, we scale the features of three modalities - property, semantic, and community structure - to a consistent dimension using an MLP. Then, we perform summation, concatenation, self-attention-based fusion, and cross-attention-based fusion, respectively. In the self-attention-based fusion method, we set Q, K, and V as matrices formed by concatenating the features of the three modalities. In the cross-attention-based fusion method, the setting of Q, K, and V matrices remains the same as mentioned in the previous paragraph when community structure features are included. The experimental results are shown in Table VI, each result is the average of three repeated experiments. It can be observed that the cross-attention-based fusion method achieves the best performance.

Therefore, through the ablation experiments on community structure features and fusion methods, we have validated the effectiveness of community structure features and the cross-attention fusion method in the proposed approach.

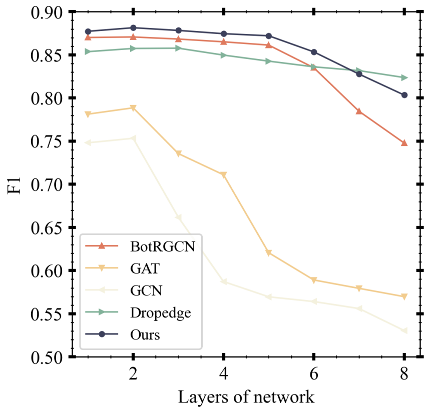
Complexity analysis: Our model demonstrates a slight increase in complexity compared to traditional GCN-based methods. This is primarily due to the integration of community structure features and the application of cross-attention mechanisms. Specifically, we follow a similar approach to BotRGCN for incorporating community structure features. In our model, the increase in feature dimension is modest, going from 128 ($\mathbb{R}^{N\times 128}$) to 160 ($\mathbb{R}^{N\times 160}$) dimensions. With cross-attention, the dimensions for semantic, property, and community structure features are uniformly set at 64 ($\mathbb{R}^{N\times 64}$). Post-fusion, the output dimension remains at $\mathbb{R}^{N\times 64}$, leading to an added time complexity of $N \times 64^2$, where N represents the number of accounts. Overall, this results in a time complexity increase that is linear, or $O(N)$.

VI. Discussion

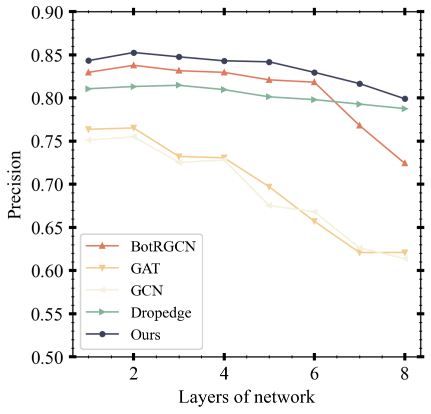
This section discusses the significance and effects of the research results provided in this paper. The investigation in [11] and extensive experimental results in Section V show that the evolution of social bots made social bot detection methods using only a single type of feature less effective in detecting novel bots. Existing social bot detection methods



(a) Accuracy



(b) F1-Scores



(c) Precision

Fig. 6: The values of Accuracy, F1-Score and Precision obtained by various models on the Twibot-20 dataset when the network layer increases.

TABLE V: Testing accuracy (%) comparisons on different backbones w and w/o community feature.

Backbones	Twibot-20		Twibot-22	
	Original	Add “C”	Original	Add “C”
GCN	77.53	**79.27**	72.39	**73.47**
GAT	83.27	**83.97**	78.36	**79.11**
Dropedge	83.71	**84.16**	78.42	**79.73**
BotRGCN	84.61	**85.97**	79.66	**80.67**
Ours	85.67	**86.53**	79.97	**81.33**

TABLE VI: Testing accuracy (%) comparisons on different fusion methods.

Dataset	Fusion Methods			
	Sum	Cat	SA	CA
Twibot-20	85.89	86.27	85.98	**86.53**
Twibot-22	79.98	81.07	80.34	**81.33**

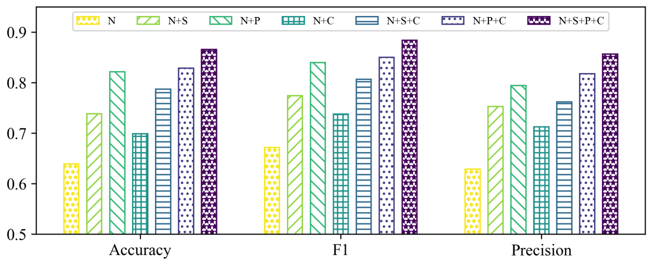


Fig. 7: Illustration of Accuracy, F1-Score and Precision when using various combination of the features for the training of RGCN model on the Twibot-20 dataset. The features used are accounts' neighbor information (N), semantic information (S), property information (P) and community structure information (C).

using multiple types of features have yielded promising results in detecting novel bot tasks, but they ignore the novel macro-community structure features of the accounts [1, 12]. The ablation experiments in Section V verify that property features, semantic features and community structure features can complement each other, and the simultaneously leverage of them is beneficial to the detection of novel social bots. At the same time, it also verifies the effectiveness of adding community structure features to the GCN-based model for social bot detection.

The most significant difference between our model and the existing models is the first model simultaneously leverage of accounts' semantic features, property features, community structure features and neighborhood features for social bot detection. Besides, we also improved the method of multi-modal feature fusion. To explicitly compare with the detection models, we present an overview of the account features used by each model in Table VII. The social bot detection method proposed by Miller et al. [33] utilizes only the property features of the account. Deepwalk [48] and Node2vec [49] utilize the local neighborhood structure features of the account, and Bot2vec [2] utilizes the local neighborhood structure features and macro community features of the account. GCN [40], GAT [50], Dropedge [51], SATAR [1] and BotRGCN [12] all exploit the semantic features, property features, and neighbor features of the account. However, our model additionally considers the community structure features of the account as well.

Although Bot2vec takes into account the community structure features, the way we extract and utilize community structure features is different from it. Firstly, the methods of community division used by Bot2vec and BotCF are different. Bot2vec uses Louvain for community division, while our method employs the more advanced community detection method based on Non-negative Matrix Factorization, DANMF [44], for community division. Compared to Louvain, DANMF exhibits better performance in community detection. Secondly, the detected communities play different roles in the two methods. In Bot2vec, the step of community division is to control the direction of the random walk, increasing the attention weight among nodes within the same community. In our method, the detected community serves as one of the multimodal features and supplements the structural features. The purpose is to prevent the graph neural network from excessively focusing on the features of neighbor nodes while ignoring the features of nodes with high similarity in the macro structure.

The improvement obtained by simultaneously using more types of features is quite obvious. By taking the results obtained on the TwiBot-20 dataset provided in Table IV as an example, our model improves the detection accuracy by 22.06% compared to the model ([33]) that only uses property features, and improves by 23.25% over the model, Bot2vec, that only uses structural features. Compared with the state-of-the-art model, BotRGCN, that uses multi-type features except community structure features, the detection accuracy of our model is improved by 1.86%.

In general, the benefits achieved by adding accounts' community structure features to the GCN-based social bot detection model may result from the following reason. It may circumvent the problem of shallow GCN that ignores the structural similarity of nodes, meanwhile, mitigate the effects of over-smoothing in deep GCN. It should be noted that the proposed model is a general social bot detection framework, which is applicable for further more meaningful features. It can also adjust the features dynamically according to the development of the social bots, which has great potential for industrial applications.

TABLE VII: Overview of account information used by the compared models.

Model	Semantic	Property	Neighbor	Community
Miller et. al		✓		
Deepwalk			✓	
Node2vec			✓	
Bot2vec			✓	✓
GCN	✓	✓	✓	
GAT	✓	✓	✓	
Dropedge	✓	✓	✓	
SATAR	✓	✓	✓	
BotRGCN	✓	✓	✓	
Ours	✓	✓	✓	✓

VII. Conclusion

In this paper, we propose a social bot detection method that further considers the community features, namely BotCF. In the proposed model, the community features are obtained by the community division algorithm based on a deep autoencoder-like non-negative matrix factorization. Besides the existing semantic and property features, two new property features, retweets and semantic vector distance, are proposed for social bot detection, forming a social bot detection model that simultaneously considers the account's semantic, property and community structure information. Further, we also consider the fusion of multimodal features using the cross-attention mechanism to fuse multimodal features. The experimental results on the three benchmark datasets show that the proposed model achieves better performance than the SOTA method. The investigative experiments indicate that the proposed model mitigates the effects of over-smoothing in deep GCNs, as well as circumvents the problem of shallow GCNs that ignore the structural similarity of nodes. In the future, we will further mine additional account information sources and explore the construction of heterogeneous graphs to detect social bots using accounts in social networks with multiple types of activity relationships.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No.2022YFB3102904), the National Natural Science Foundation of China (Grant Nos.62002387, 62472440, U1804263) and Science and Technology Research Project of Henan Province (No.222102210075), China.

References

[1] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, "SATAR: A self-supervised approach to Twitter account representation learning and its application in bot detection," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, 2021, pp. 3808–3817.

[2] P. Pham, L. T. Nguyen, B. Vo, and U. Yun, "Bot2vec: a general approach of intra-community oriented representation learning for bot detection in different types of social networks," *Information Systems*, vol. 103, p. 101771, 2022.

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

[4] X. Zhang, H. Xie, P. Yi, and J. C. Lui, "Enhancing sybil detection via social-activity networks: A random walk approach," *IEEE Transactions on Dependable and Secure Computing*, 2022, published online, DOI:10.1109/TDSC.2022.3151701.

[5] K. K. Bharti and S. Pandey, "Fake account detection in Twitter using logistic regression with particle swarm optimization," *Soft Computing*, vol. 25, no. 16, pp. 11 333–11 345, 2021.

[6] H.-C. H. Chang, E. Chen, M. Zhang, G. Muric, and E. Ferrara, "Social bots and social media manipulation in 2020: the year in review," in *Handbook of Computational Social Science, Volume 1*. Routledge, 2021, pp. 304–323.

[7] P. N. Howard, B. Kollanyi, and S. Woolley, "Bots and automation over Twitter during the U.S. Election," *Computational propaganda project: Working paper series*, vol. 21, no. 8, 2016.

[8] O. Varol, E. Ferrara, C. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Proceedings of the international AAAI conference on web and social media*, vol. 11, no. 1, 2017, pp. 280–289.

[9] K.-C. Yang, C. Torres-Lugo, and F. Menczer, "Prevalence of low-credibility information on Twitter during the

COVID-19 outbreak," in *Proceedings of the international AAAI conference on web and social media*, 2020, pp. 224–228.

[10] J. Donovan, "Stuck: How vaccine rumors start-and why they don't go away," *Nature*, vol. 583, no. 7818, pp. 680–681, 2020.

[11] S. Cresci, "A decade of social bot detection," *Communications of the ACM*, vol. 63, no. 10, pp. 72–83, 2020.

[12] S. Feng, H. Wan, N. Wang, and M. Luo, "BotRGCN: Twitter bot detection with relational graph convolutional networks," in *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2021, pp. 236–239.

[13] B. Wang, N. Z. Gong, and H. Fu, "GANG: Detecting fraudulent users in online social networks via guilt-by-association on directed graphs," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 465–474.

[14] B. Wang, J. Jia, L. Zhang, and N. Z. Gong, "Structure-based sybil detection in social networks via local rule-based propagation," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 3, pp. 523–537, 2018.

[15] N. El-Mawass, P. Honeine, and L. Vercouter, "Similcatch: Enhanced social spammers detection on Twitter using markov random fields," *Information processing & management*, vol. 57, no. 6, p. 102317, 2020.

[16] S. Noekhah, N. binti Salim, and N. H. Zakaria, "Opinion spam detection: Using multi-iterative graph-based model," *Information Processing & Management*, vol. 57, no. 1, p. 102140, 2020.

[17] A. Abou Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "Botchase: Graph-based bot detection using machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 15–29, 2020.

[18] Z. Qu, C. Lyu, and C.-H. Chi, "Mush: Multi-stimuli hawkes process based sybil attacker detector for user-review social networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4600–4614, 2022.

[19] M. Fazil, A. K. Sah, and M. Abulaish, "DeepSBD: a deep neural network model with attention mechanism for socialbot detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4211–4223, 2021.

[20] S. Yardi, D. Romero, G. Schoenebeck *et al.*, "Detecting spam in a twitter network," *First monday*, vol. 15, pp. 1–4, 2010.

[21] K. Lee, B. Eoff, and J. Caverlee, "Seven months with the devils: A long-term study of content polluters on Twitter," in *Proceedings of the international AAAI conference on web and social media*, vol. 5, no. 1, 2011, pp. 185–192.

[22] C. Yang, R. Harkreader, and G. Gu, "Empirical evaluation and new design for fighting evolving Twitter spammers," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1280–1293, 2013.

[23] D. M. Beskow and K. M. Carley, "Bot conversations are different: leveraging network metrics for bot detection in Twitter," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 2018, pp. 825–832.

[24] K.-C. Yang, O. Varol, P.-M. Hui, and F. Menczer, "Scalable and generalizable social bot detection through data selection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 1096–1103.

[25] J. Pastor-Galindo, M. Zago, P. Nespoli, S. L. Bernal, A. H. Celdrán, M. G. Pérez, J. A. Ruipérez-Valiente, G. M. Pérez, and F. G. Mármol, "Spotting political social bots in twitter: A use case of the 2019 spanish general election," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2156–2170, 2020.

[26] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Detecting automation of Twitter accounts: Are you a human, bot, or cyborg?" *IEEE Transactions on dependable and secure computing*, vol. 9, no. 6, pp. 811–824, 2012.

[27] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Social fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 561–576, 2017.

[28] J. Rodríguez-Ruiz, J. I. Mata-Sánchez, R. Monroy, O. Loyola-González, and A. López-Cuevas, "A one-class classification approach for bot detection on Twitter," *Computers & Security*, vol. 91, p. 101715, 2020.

[29] R. De Nicola, M. Petrocchi, and M. Pratelli, "On the efficacy of old features for the detection of new bots," *Information Processing & Management*, vol. 58, no. 6, p. 102685, 2021.

[30] M. Sayyadiharikandeh, O. Varol, K.-C. Yang, A. Flammini, and F. Menczer, "Detection of novel social bots by ensembles of specialized classifiers," in *Proceedings of the 29th ACM international conference on information & knowledge management (CIKM)*, 2020, pp. 2725–2732.

[31] K.-C. Yang, E. Ferrara, and F. Menczer, "Botometer 101: Social bot practicum for computational social scientists," *Journal of Computational Social Science*, pp. 1–16, 2022.

[32] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "BotOrNot: A system to evaluate social bots," in *Proceedings of the 25th international conference companion on world wide web*, 2016, pp. 273–274.

[33] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, "Twitter spammer detection using data stream clustering," *Information Sciences*, vol. 260, pp. 64–73, 2014.

[34] H. Ping and S. Qin, "A social bots detection model based on deep learning algorithm," in *2018 IEEE 18th international conference on communication technology (ICCT)*. IEEE, 2018, pp. 1435–1439.

[35] I. Karpov and E. Glazkova, "Detecting automatically managed accounts in online social networks: Graph embeddings approach," in *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 2020, pp. 11–21.

[36] J. Zhang, R. Zhang, J. Sun, Y. Zhang, and C. Zhang, "TrueTop: A sybil-resilient system for user influence measurement on Twitter," *IEEE/ACM Transactions on*

Networking, vol. 24, no. 5, pp. 2834–2846, 2015.

[37] B. Wang, L. Zhang, and N. Z. Gong, "SybilSCAR: Sybil detection in online social networks via local rule based propagation," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[38] J. Jia, B. Wang, and N. Z. Gong, "Random walk based fake account detection in online social networks," in *2017 47th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2017, pp. 273–284.

[39] P. Gao, B. Wang, N. Z. Gong, S. R. Kulkarni, K. Thomas, and P. Mittal, "SybilFuse: Combining local attributes with global structure to perform robust sybil detection," in *2018 IEEE conference on communications and network security (CNS)*. IEEE, 2018, pp. 1–9.

[40] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[41] Y. Sun, Z. Yang, and Y. Dai, "TrustGCN: enabling graph convolutional network for robust sybil detection in OSNs," in *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2020, pp. 1–7.

[42] S. Ali Alhosseini, R. Bin Tareaf, P. Najafi, and C. Meinel, "Detect me if you can: Spam bot detection using inductive representation learning," in *Companion Proceedings of The 2019 World Wide Web Conference*, 2019, pp. 148–153.

[43] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[44] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proceedings of the 27th ACM international conference on information and knowledge management (CIKM)*, 2018, pp. 1393–1402.

[45] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Fame for sale: Efficient detection of fake Twitter followers," *Decision Support Systems*, vol. 80, pp. 56–71, 2015.

[46] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, "TwiBot-20: A comprehensive Twitter bot detection benchmark," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, 2021, pp. 4485–4494.

[47] S. Feng, Z. Tan, H. Wan, N. Wang, Z. Chen, B. Zhang, Q. Zheng, W. Zhang, Z. Lei, S. Yang *et al.*, "TwiBot-22: Towards graph-based Twitter bot detection," *arXiv preprint arXiv:2206.04564*, 2022.

[48] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[49] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations(ICLR)*, 2018.

[51] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *International Conference on Learning Representations*, 2020.

[52] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International conference on machine learning*. PMLR, 2018, pp. 5453–5462.

[53] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 385–394.

[54] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[55] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[56] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI conference on artificial intelligence*, 2018, pp. 3538–3545.

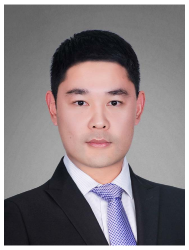
Feng Liu received the B.S. degree in information and computing sciences from Shenyang university of chemical technology, Shenyang, China, in 2018, the M.S. degree in Cyberspace Security from Zhengzhou University, Zhengzhou, China, in 2023. He is currently working toward the PhD degree with School of Artificial Intelligence, Jilin University, Changchun, China. His research interest includes include social bot detection and deep learning.

Zhenyu Li received the B.S. degree in information computing sciences from Hefei University of Technology, Hefei, China, in 2011, the M.S. degree in computer application technology from Zhengzhou Information Science and Technology Institute, Zhengzhou, China, in 2014, and the Ph.D. degree in computer science from University of York, York, UK, in 2018. He is currently an associate professor with Henan Provincial Key Laboratory of Cyberspace Situation Awareness. His research interests include social bot detection, multimedia security and machine learning.

Chunfang Yang received the MA and PhD degrees in computer science and technology from Zhengzhou Information Science and Technology Institute, Zhengzhou, China, in 2008 and 2012, respectively. He is currently an associate professor of Henan Key Laboratory of Cyberspace Situation Awareness. His research interest includes network and information security.

Daofu Gong received the B.S. degree, the M.S. degree, and the Ph.D. degree from Zhengzhou Information Science and Technology Institute, Zhengzhou, China, in 2006, 2009, and 2013, respectively. He is currently an associate professor with Henan Provincial Key Laboratory of Cyberspace Situation Awareness. His research interests include machine learning, network security situation awareness, and social network analysis.

Fenlin Liu received the B.S. degree from Zhengzhou Information Science and Technology Institute in 1986, the M.S. degree from the Harbin Institute of Technology in 1992, and the Ph.D. degree from Northeast University in 1998. He is currently a professor with Henan Provincial Key Laboratory of Cyberspace Situation Awareness. His research interests include digital image forensics, information hiding, and recommender systems.



Rui Ma is an associate professor in School of Artificial Intelligence, Jilin University, China. He obtained his PhD from School of Computing Science, Simon Fraser University, Canada and his MSc and BSc from School of Mathematics, Jilin University, China. His research is in computer graphics, computer vision and artificial intelligence, with special interests in intelligent analysis, creation and application of visual content.

Adrian G. Bors (Senior Member, IEEE) received the MSc degree in electronics engineering from the Polytechnic University of Bucharest, Bucharest, Romania, in 1992, and the PhD degree in informatics from the University of Thessaloniki, Thessaloniki, Greece, in 1999. In 1999 he joined the Department of Computer Science, Univ. of York, U.K., where he is currently an associate professor. He was a research scientist with the University of Tampere, Finland, and held visiting positions with the University of California at San Diego (UCSD), the University of Montpellier, France and with the MBZ University of Artificial Intelligence, Abu Dhabi, UAE. He was an associate editor for IEEE Transactions on Image Processing from 2010 to 2014 and IEEE Transactions on Neural Networks from 2001 to 2009. He was also a co-guest editor for special issues for the International Journal for Computer Vision in 2018 and the Journal of Pattern Recognition in 2015. He has authored and coauthored more than 180 research papers, including 50 in journals. His research interests include machine learning, computer vision, pattern recognition, and image processing.