Hindawi
Security and Communication Networks
Volume 2020, Article ID 8810817, 12 pages
https://doi.org/10.1155/2020/8810817

Research Article

GFD: A Weighted Heterogeneous Graph Embedding Based Approach for Fraud Detection in Mobile Advertising

Jinlong Hu,[1,2] **Tenghui Li**,[1,2] **Yi Zhuang**,[1,2] **Song Huang**,[1,2] **and Shoubin Dong**[1,2]

[1]*School of Computer Science and Engineering, South China University of Technology, Guangzhou, China*
[2]*Communication and Computer Network Laboratory of Guangdong, South China University of Technology, Guangzhou, China*

Correspondence should be addressed to Jinlong Hu; jlhu@scut.edu.cn

Received 9 April 2020; Revised 28 July 2020; Accepted 25 August 2020; Published 4 September 2020

Academic Editor: Hammad Afzal

Online mobile advertising plays a vital role in the mobile app ecosystem. The mobile advertising frauds caused by fraudulent clicks or other actions on advertisements are considered one of the most critical issues in mobile advertising systems. To combat the evolving mobile advertising frauds, machine learning methods have been successfully applied to identify advertising frauds in tabular data, distinguishing suspicious advertising fraud operation from normal one. However, such approaches may suffer from labor-intensive feature engineering and robustness of the detection algorithms, since the online advertising big data and complex fraudulent advertising actions generated by malicious codes, botnets, and click-firms are constantly changing. In this paper, we propose a novel weighted heterogeneous graph embedding and deep learning-based fraud detection approach, namely, GFD, to identify fraudulent apps for mobile advertising. In the proposed GFD approach, (i) we construct a weighted heterogeneous graph to represent behavior patterns between users, mobile apps, and mobile ads and design a weighted metapath to vector algorithm to learn node representations (graph-based features) from the graph; (ii) we use a time window based statistical analysis method to extract intrinsic features (attribute-based features) from the tabular sample data; (iii) we propose a hybrid neural network to fuse graph-based features and attribute-based features for classifying the fraudulent apps from normal apps. The GFD approach was applied on a large real-world mobile advertising dataset, and experiment results demonstrate that the approach significantly outperforms well-known learning methods.

1. Introduction

Online mobile advertising plays a vital role in the mobile app ecosystem. One of the popular models in mobile app advertising is known as cost per action (CAP), where payment is based on user action, such as downloading and installing an app on the user's mobile device. This CAP model may incentivize malicious mobile content publishers (typically app owners) to generate fraudulent actions on advertisements to get more financial returns [1–3]. Some traditional methods and techniques have been used for detecting and stopping click fraud, such as threshold-based method [4], CAPTCHA [5], splay tree [6], TrustZone [7], power spectral density analysis [8], and social network analysis [9].

To automatically detect mobile advertising fraud behaviors, machine learning methods have been successfully applied to find fraud patterns in data, distinguishing suspicious advertising fraud operation from normal one [10–14]. As for learning model with attribute features, researchers usually use several attributes from each sample to train a learning model to identify the fraud behaviors. Unfortunately, such approaches may suffer from labor-intensive feature engineering and robustness of the detection algorithms, since the online advertising big data and complex fraudulent advertising actions generated by malicious codes, botnets, and click-firms are constantly changing. What is more, fraudsters could easily adjust their fraud patterns based on existing fraud detection attributes and rules to avoid being detected. Recently, some researchers try to use the relationship between information entities to construct a graph model and then use the graph mining or learning methods to identify the changing fraud behaviors

[15–17]. All these methods obtain useful insights into the learning mechanism to classify fraud behaviors from normal activities. Intuitively, if we could combine the complementary information from attributes of sample data and relationship between entities (e.g., users, apps, and ads), we will be able to improve the accuracy and robustness of fraud detection.

However, to unleash the power of attribute-based information and graph-based information, we have to address a series of challenges. First, to take advantage of the characteristic of graph, we should construct a suitable graph, which could potentially represent the interaction behaviors between information entities such as users, apps, and ads. Second, an efficient graph learning method should be developed to learn the useful structural and semantic representation information from constructed graph [18, 19], particularly learning from heterogeneous graph [20]. Third, fusing different kinds of information from sample attributes and node representation is difficult for their inherent heterogeneity and high-order characteristics.

To address the above challenges, in this paper, we propose a weighted heterogeneous graph embedding and deep learning-based fraud detection approach, namely, GFD, to identify fraudulent apps for mobile advertising. In the proposed GFD approach, (i) considering behavior patterns between users, mobile apps, and mobile ads, we construct a weighted heterogeneous graph to represent mobile app advertising behavior and propose a new weighted metapath to vector algorithm, namely, WMP2vec, to learn low-dimensional latent representation (graph-based features) for apps' nodes in the weighted heterogeneous graph; (ii) we use a time window based statistical analysis method to extract intrinsic features (attribute-based features) from the tabular sample data; (iii) we present a hybrid convolutional neural network model to fuse graph-based features and attribute-based features for classifying the fraudulent apps from normal apps.

We evaluate GFD approach and WMP2vec algorithm on a real-world dataset from one of the mobile advertising platforms in China. Results show that WMP2vec reaches higher performance than three well-known graph embedding algorithms in the constructed weighted heterogeneous graph, and GFD approach achieves highest classification performance compared with Support Vector Machine (SVM), Random Forest (RF), and Fully Connected Neural Networks (FCNN).

The rest of the paper is organized as follows. We introduce GFD approach to detect fraudulent apps with deep neural networks and heterogeneous graph embedding algorithm WMP2vec in Section 2. We present the experimental results and discussion in Section 3. In Section 4, we introduce the related work. We conclude this paper in Section 5.

2. Proposed Approach

The flow chart of the proposed GFD approach is shown in Figure 1. First, we propose a weighted heterogeneous graph embedding method to learn the node representation, including constructing the weighted heterogeneous graph and the WMP2vec algorithm. Second, we use statistical analysis method to extract attribute-based features from the tabular sample data. Third, we introduce the deep neural networks to fuse the attribute-based features and graph-based features for identifying fraudulent apps from normal ones.

2.1. Data Description. We collect advertising log data of mobile apps from a mobile advertising platform. Our mobile advertisement dataset contains the following attributes: user ID, a code to identify a unique mobile user; app ID, a code to identify a unique mobile app; ad ID, a code to identify a unique mobile advertisement; geographical attributes, a series of user geographical attributes used to detect anomalies, including encrypted IP and city; action type, user behavior related to the ads, such as viewing, clicking, app downloading start, app downloading completion, and app installation completion; action time, the time-stamp when the action happened; and device attribute, user device related attributes, such as device ID, device system models, and screen size.

A seven-day mobile advertising log dataset in June 2015 was studied in this paper, and some examples of our raw data are shown in Table 1.

2.2. Weighted Heterogeneous Graph Embedding. In this section, we firstly propose the problem definition and construct the weighted heterogeneous graph, and then we present WMP2vec algorithm to learn latent representation of nodes in weighted heterogeneous graph.

2.2.1. Problem Definition

(1) Given. An undirected weight heterogeneous graph $G = \langle V, E, W \rangle$ is given, where V is a set of app nodes, ad nodes, and user nodes; E is a set of undirected weight edges between any two types of nodes: app nodes and user nodes, user nodes and ad nodes, and ad nodes and app nodes; W is the set of weight of edges.

(2) Task. The task is to learn the d-dimensional latent representations $X_e \in \Re^{|V| \times d}$ (where $d \ll |V|$) for nodes, which could capture the structural and semantic relations among nodes in the graph G, and the representations could be used for classifying fraudulent apps.

2.2.2. Weighted Heterogeneous Graph Construction. Let U be the set of user nodes, let A be the set of app nodes, and let P be the set of advertisement nodes. If there exists an action from user $u \in U$ to advertisement $a \in A$ through app $p \in P$, we form edges from u to p, from u to a, and from p to a, respectively, such that $E_1 = U \times P$, $E_2 = U \times A$, and $E_3 = P \times A$ are the edges set of heterogeneous graph G. The set of weight is $W = \{w_{up}, w_{ua}, w_{pa}\}$, where the weights w_{up}, w_{ua}, and w_{pa} are defined proportional to the behavioral centrality of u to p, u to a, and p to a, respectively. The calculation

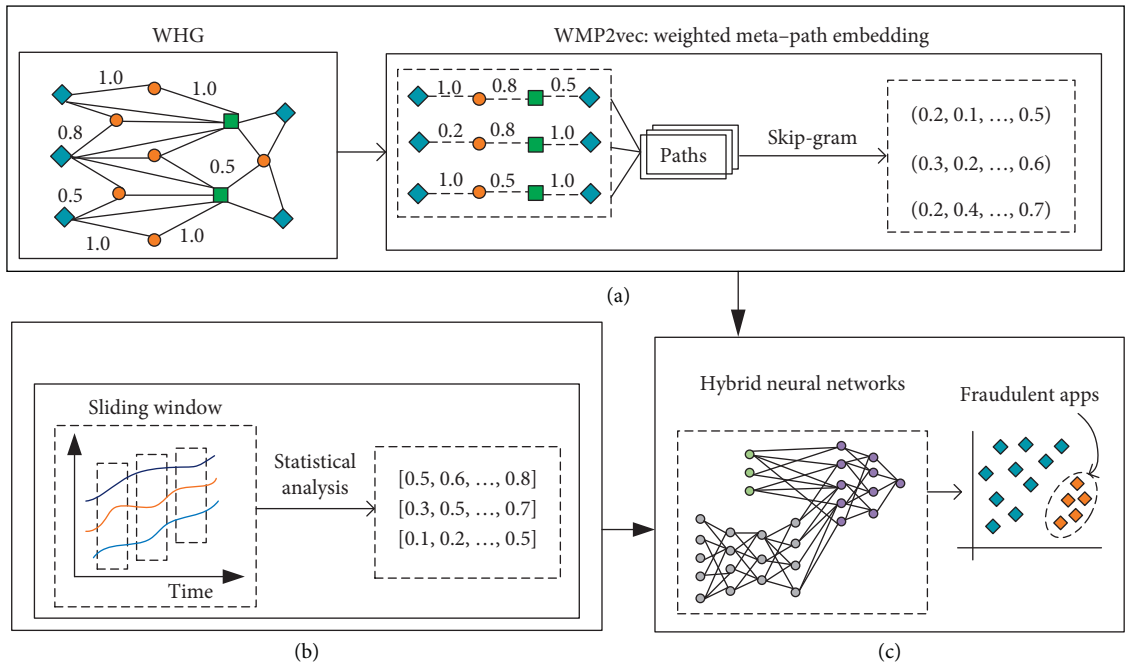

Figure 1: The flow chart of GFD approach. (a) Graph-based feature extraction. (b) Attribute-based feature extraction. (c) Deep Fusion.

Table 1: Example of the mobile advertising log data.

User ID	App ID	Ad ID	Action
B360**	*369	*103	Viewing
B360**	*369	*103	Clicking
Xjnh**	*370	*125	Viewing
Xjnh**	*370	*125	Downloading
Lmsv**	*412	*130	Downloaded
Lmsv**	*412	*130	Installing
Lmsv**	*412	*133	Installed

formula of w_{up} is shown in equation (1) and so on for w_{ua} and w_{pa}.

$$w_{\text{up}} = \frac{C_{\text{up}}}{\sum_{j \in \Omega(u)} C_{uj}}, \tag{1}$$

where C_{up} is the times of user u operating on advertisement p and $\Omega(u)$ is the set of operations of user u on all the advertisements.

2.2.3. Graph Embedding Algorithm. In this section, based on the sequence generation method from metapath based random walk in heterogeneous graph [20], we propose WMP2vec algorithm to generate random walk sequence in weighted heterogeneous graph and embed sequence to representation vector with Skip-Gram [21] for nodes.

(1) Weighted Metapath Based Random Walk. We predefined number of walks per node n, the number of walk sequences l, and a metapath M. The metapath is defined as a path in the heterogeneous graph G with its metatemplate $T_G = (Z, R)$, where $Z = \{Z_u, Z_p, Z_a\}$ and $R = \{R_u, R_p, R_a\}$. Each node v and each edge e are associated with mapping functions $\varphi(v)$: $V \longrightarrow Z$ and $\phi(e)$: $E \longrightarrow R$, respectively.

Supposing that current node is v^i, the relationship between v^i and next node v^{i+1} is R_i; that is, $\phi(v^i, v^{i+1}) = R_i$.

For walk sequences generation, we go through the metapath scheme l times, and each time generates one corresponding walk sequence. In the first time, we use two different selecting methods (first phase and second phase), because there are no limits to edge weight in the beginning. After first time, we use the method in the second phase to select next node.

For the first phase, when the length of walk sequence is less than 2, the next node in the sequence is randomly selected from the neighbors set $N_{t+1}(v^i)$ of current nodes, which meet the requirements of metapath M [20]. The transition probability from v^i to v^{i+1} is defined as follows:

$$p_1\left(v^{i+1} \mid v^i, M\right) = \begin{cases} 0, & \phi\left(v^i, v^{i+1}\right) \neq R_i, \\ \dfrac{1}{\left|N_{t+1}\left(v^i\right)\right|}, & \phi\left(v^i, v^{i+1}\right) = R_i. \end{cases} \tag{2}$$

For the second phase, when the length of walk sequence is between 2 and $l^*|R|$, the transition probability is restricted by a weight bias β. Supposing that the latest weight of edge of relationship R_i is w_i, the weight should be in the range of $[w_i - \beta, w_i + \beta]$. The transition probability from v^i to v^{i+1} is defined as follows:

$$p_2\left(v^{i+1} \mid v^i, M_w\right) = \begin{cases} 0, & \phi\left(v^i, v^{i+1}\right) \neq R_i, \\ 0, & \phi\left(v^i, v^{i+1}\right) = R_i; w_{v^{i+1}, v^i} \notin [w_i - \beta, w_i + \beta], \\ \dfrac{1}{\left|C\left(v^i\right)\right|}, & \phi\left(v^i, v^{i+1}\right) = R_i; w_{v^{i+1}, v^i} \in [w_i - \beta, w_i + \beta], \end{cases} \tag{3}$$

where $C(v^i)$ is the set of neighbors meeting the requirement.

(2) Embedding Sequence to Vector with Skip-Gram. Based on the weighted metapath random walk sequences, we use Skip-Gram model [21] and negative sampling [22] to learn low-dimensional representation of nodes.

A description of our proposed WMP2vec algorithm method is shown in Algorithm 1.

2.3. Attribute-Based Feature Extracting. From the raw log data (tabular data) of mobile advertising, we defined a time window (t hours) and divide original data into $24/t$ data block for one day (24 hours). Then, a plain statistical analysis is performed on each field in each data block. The ratio of the unique value of the field to the total number of records in the specified time window is computed. The attribute-based feature corresponding to one mobile app could be represented as a feature matrix with $24/t$ rows.

2.4. Hybrid Neural Network for Classification. To take advantage of the graph-based features and attribute-based features, we propose a hybrid convolutional neural networks (HNN) model to fuse and learn both information in GFD approach. The overview of the hybrid neural networks is shown in Figure 2.

In HNN model, the first layer (input layer) contains attribute-based feature matrix $X_s \in \mathfrak{R}^{N \times t \times m}$ and graph-based feature $X_e \in \mathbb{R}^{N \times d}$, where N is the number of samples, t is the number of time windows by one day (24 hours), m is the dimension of attribute-based feature in a time window, and d is the dimension of node embedding.

A convolutional part includes two convolutional layers, and the output of the first convolutional layer is

$$C_1 = \text{active}(z) = \text{active}\left(X_s{}^* W_{C_1} + b_{c_1}\right), \tag{4}$$

where $W_{C_1} \in \mathbb{R}^{w_1 \times w_1}$ and $b_{c_1} \in \mathbb{R}$ are the convolution kernel and bias, respectively, w_1 is the size of the kernel, * indicates the convolution operation, and the active function is $\text{relu}(x) = \max(0, x)$.

The second convolutional layer is constructed as follows:

$$C_2 = \text{acitve}\left(C_1 * W_{C_2} + b_{c_2}\right), \tag{5}$$

where $W_{C_2} \in \mathbb{R}^{w_2 \times w_2}$ and $b_{c_2} \in \mathbb{R}$ are the convolution kernel and bias, respectively. w_2 is the size of the kernel.

C_2 is flattened to $X_c \in \mathbb{R}^{N \times d_0}$, where d_0 is the number of elements in C_2.

We concatenate X_c and X_r into a single metric $X \in \mathbb{R}^{N \times (d_0 + d)}$ to be the input of the first fully connected layer l_1. l_1 is constructed as follows:

$$l_1 = \text{active}\left(XW_1 + b_1\right), \tag{6}$$

where $W_1 \in \mathbb{R}^{(d_0+d) \times d_1}$ and $b_1 \in \mathbb{R}^{d_1}$ are weight and bias, respectively, and d_1 is the number of neurons in the first fully connected layer.

The second fully connected layer l_2 is constructed as follows:

$$l_2 = \text{active}\left(l_1 W_2 + b_2\right), \tag{7}$$

where $W_2 \in \mathbb{R}^{d_1 \times d_2}$ and $b_2 \in \mathbb{R}^{d_2}$ are weight and bias, respectively, and d_2 is the number of neurons in the second fully connected layer.

In the output of HNN, $\hat{y} \in (0, 1)$ is the probability of an application to be a fraudulent application.

$$\hat{y} = \sigma\left(l_2 W_o + b_o\right), \tag{8}$$

where $W_o \in \mathbb{R}^{d_2 \times 1}$ and $b_o \in \mathbb{R}^{d_2}$ are weight and bias, respectively, and $\sigma(\cdot) = \text{sigmoid}(\cdot)$ is the sigmoid function.

The cross-entropy function with l2-regularization is used to calculate the loss of the hybrid convolutional neural network model.

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y)\log(1 - \hat{y}) + \frac{\lambda}{2}||\theta||^2. \tag{9}$$

3. Experiments

3.1. Data Description and Preprocessing. A real-world dataset was collected from a mobile advertising platform in China. The dataset consists of seven days with around 2 M users, 3.5 K apps, and 1 K advertisements per day. We partition our log data into seven subsets with one-day period and conduct experiments on each subset to evaluate our model. The proportion of fraudulent apps is about 2–4 percent in the total 3,500 apps each day. More details of the dataset are described in Section 2.1.

3.2. Evaluation Metric. In this paper, we define the fraudulent apps by positive samples and the other apps by negative samples. The Average Precision (AP) and the Area Under ROC Curve (AUC) are used to evaluate proposed algorithm and approach.

The AP criterion summarizes the Precision-Recall performances at different threshold levels and corresponds to area under the Precision-Recall curve. The ROC curve is

Input: The weighted heterogeneous information graph $G = <V, E, W>$, a meta-path scheme M, walks per node n, longest walk length per walk l, embedding dimension d, neighborhood size k
Output: The latent node embedding $X \in R^{|V| \times d}$
Initialize X, random walk sequence $S = \varnothing$
for $v \in V$ **do**
for $i = 1 \longrightarrow n$ **do**
S_t = WeightedMetaPathRandomWalk (G, M, v, l)
$S = S + S_t$
end
end
X = HeterogeneousSkipGram (X, k, S)
return X
WeightedMetaPathRandomWalk (G, M, v, l)
initialize random walk array $S = [v]$, weight array $w_i = [1.0]$, $i = 1 \ldots |M|/2$
relationship array $R = [R_1, \ldots, R_{|M|/2}, R_{|M|/2}, \ldots, R_1]$
for $j = 1 \longrightarrow l$ **do**
for $k = 1 \longrightarrow |M|/2$ **do**
if $j = 1$ **then**
    draw u and w according to equation (2) with relationship $R[k-1]$
    $S[j+1] = u, W_k[j+1] = w$
else
    draw u and w according to equation (3) with relationship $R[k-1]$
    if u does not exist **then** return S
else $S[j+1] = u, W_k[j+1] = w$
end
    for $k = |M|/2 + 1 \longrightarrow |M| - 1$**do**
        draw u and w according to equation (3) with relationship $R[k-1]$
    if u does not exist **then** return S
else $S[j+1] = u$, $W_{|M|-k}[j+1] = w$
end
end
return S

Algorithm 1: The WMP2vec algorithm.

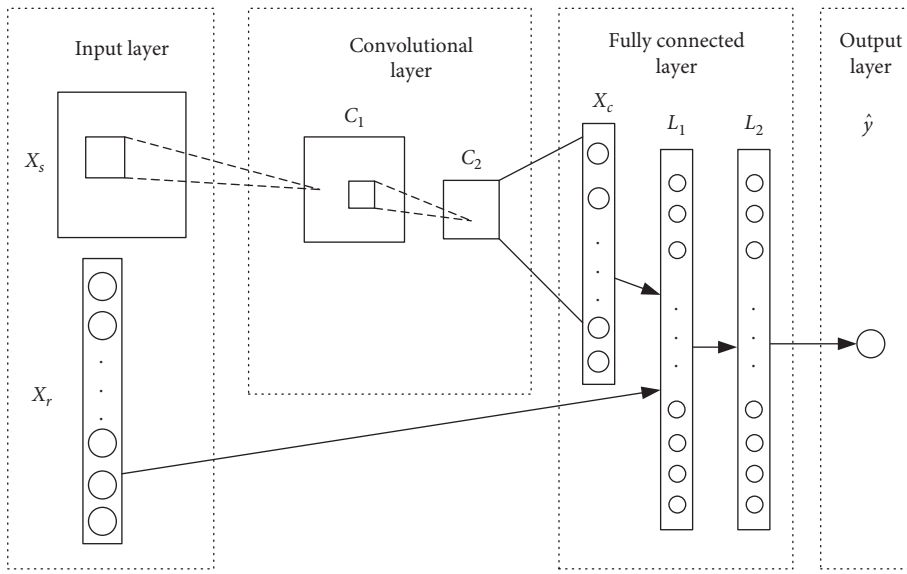

Figure 2: Hybrid convolutional neural networks for fraud detection.

created by plotting the true positive rate against the false positive rate at various threshold settings. The AUC is the total area under the ROC curve.

3.3. Evaluation of WMP2vec Algorithm. In this section, we use WMP2vec algorithm to learn the embedding vector of the nodes (apps) from the constructed weighted heterogeneous graph and then take their embedding vectors as the input of Random Forest (RF) model to classify fraudulent apps.

Based on Section 2.2.2, we construct a weighted heterogeneous graph and define a metapath: app-user-ad-user-app (PUAUP); that is, $M = Z_2 \longrightarrow^{R_1} Z_1 \longrightarrow^{R_3} Z_3 \longrightarrow^{R_3} Z_1 \longrightarrow^{R_1} Z_2$, which represents the heterogeneous semantic of fraud publishers (apps) that mimic legitimate users to act on the ads from the apps.

3.3.1. Comparison Models and Parameters. We compare the AP and AUC of the WMP2vec model with three well-known graph embedding models: DeepWalk [23], Node2vec [24], and Metapath2vec [20]. The compared algorithms and their parameters are as follows:

(1) DeepWalk: DeepWalk [23] is the first graph embedding model based on Word2vec. We use Skip-Gram model [21] and hierarchical softmax [25] with gradient descent to learn the node representation. Negative sampling technique [22] is used to accelerate the Skip-Gram model. The count of random walk is 30, and the walk length is 40.

(2) Node2vec: Node2vec [24] extends DeepWalk algorithm through introducing backward probability p and forward probability q. The same random walk parameters (count = 30 and length = 40) are used with DeepWalk, and the negative sampling technique is also used. In addition, we use $p = 0.5$ and $q = 0.2$ for backward probability and forward probability, respectively.

(3) Metapath2vec: Metapath2vec [20] uses the metapath based random walk to construct node sequences and then leverages Skip-Gram to perform node embedding. The metapath in this study is PUAUP. The count of random walk is 30, and the walk length is 10.

(4) WMP2vec: We use the same parameters (count = 30, length = 10, and metapath = PUAUP) with Metapath2vec, and the weighted bias β is 0.1 additionally.

In all the compared models, we train Skip-Gram model with window size of 5, and the negative samples is 5 in negative-sampling. The graph-based feature of each node is a 32-dimensional vector. The parameters of the RF model are as follows: the number of weak learners is 150, max. deep is 5, and min. sample leaf is 5.

3.3.2. Experimental Results. Tables 2 and 3 show the experimental results by comparing the AP and AUC over 10-fold cross-validation for seven days. The WMP2vec model reached highest AP value in six days and highest AUC value in three days over all seven days. The Metapath2vec model reached highest AP value in one day and highest AUC value in two days over seven days. Thus, WMP2vec outperforms all other models, such that WMP2vec > Metapath 2vec > Node2vec > DeepWalk.

Table 2: Classification results for all embedding models in AP (mean ± std).

Date	DeepWalk	Node2vec	Metapath2vec	WMP2vec
1st June	0.168 ± 0.082	0.343 ± 0.057	0.344 ± 0.065	0.384 ± 0.055
2nd June	0.318 ± 0.137	0.384 ± 0.079	0.342 ± 0.067	0.421 ± 0.084
3rd June	0.281 ± 0.075	0.439 ± 0.122	0.401 ± 0.110	0.459 ± 0.114
4th June	0.313 ± 0.073	0.335 ± 0.085	0.360 ± 0.096	0.409 ± 0.099
5th June	0.308 ± 0.121	0.379 ± 0.102	0.387 ± 0.074	0.411 ± 0.091
6th June	0.199 ± 0.113	0.230 ± 0.089	0.369 ± 0.097	0.404 ± 0.102
7th June	0.244 ± 0.054	0.297 ± 0.075	0.371 ± 0.094	0.353 ± 0.075

Table 3: Classification results for all embedding models in AUC (mean ± std).

Date	DeepWalk	Node2vec	Metapath2vec	WMP2vec
1st June	0.788 ± 0.027	0.801 ± 0.038	0.837 ± 0.038	0.877 ± 0.030
2nd June	0.828 ± 0.032	0.848 ± 0.025	0.864 ± 0.028	0.893 ± 0.027
3rd June	0.935 ± 0.023	0.950 ± 0.024	0.912 ± 0.028	0.921 ± 0.024
4th June	0.824 ± 0.032	0.824 ± 0.040	0.849 ± 0.045	0.879 ± 0.045
5th June	0.799 ± 0.060	0.835 ± 0.055	0.853 ± 0.052	0.849 ± 0.038
6th June	0.757 ± 0.076	0.891 ± 0.042	0.844 ± 0.046	0.856 ± 0.041
7th June	0.804 ± 0.031	0.820 ± 0.027	0.844 ± 0.038	0.825 ± 0.049

3.3.3. Impacts of Parameters. In this subsection, we evaluate the impacts of parameters over the classification task: (i) count of random walk, walk length, and window size of Skip-Gram in WMP2vec and Metapath2vec model; (ii) weighted bias β of WMP2vec. We compare the AP and AUC values in the dataset from one day.

(1) Count of Random Walk. Figure 3 shows the experimental results by comparing the AP and AUC with different count of random walk, with fixed walk length of 5. When the count of random walk is larger than 30, WMP2vec and Metapath2vec models have better performance than count = 10, respectively. In addition, the values of AP and AUC have

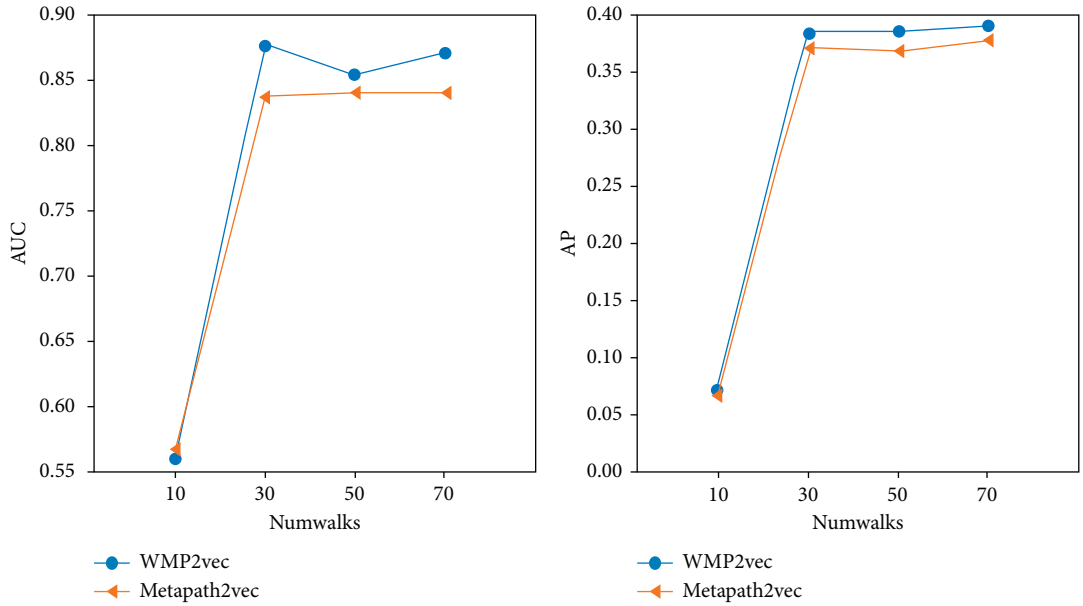

Figure 3: Comparing impacts of the different count of random walk with AUC and AP.

slight changes when the count of random walk is 30, 50, or 70.

(2) Walk Length. Figure 4 shows the experimental results by comparing the AP and AUC with different walk length (length = 5, 10, 20, 50, and 80), with fixed count of random walk of 10. WMP2vec and Metapath2vec models reach better performance when the walk length ≥10. In addition, when the length changes from 10, 20, and 50 to 80, the AP values change very little and the AUC values have some fluctuations.

(3) Window Size of Skip-Gram. Figure 5 shows the experimental results by comparing the AP and AUC with different window size (size = 3, 4, 5, 6, and 7) over the classification task. The best performance of models is reached when the window size is 5.

(4) Weighted Bias of WMP2vec. Figure 6 shows the experimental results by comparing the AP and AUC with different weighted bias β of WMP2vec ($\beta = 0.1$, 0.3, 0.5, 0.7, and 1.0) over the classification task. As the weighted bias β increases, the performance of WMP2vec gets closer to the performance when $\beta = 1.0$. The values of AP and AUC change very little when $\beta \geq 0.5$.

3.4. Evaluation of Hybrid Neural Network. In this section, we evaluate the classification performance of HNN model for fusing graph-based features and attribute-based features in GFD approach. As the flow of GFD approach in Figure 1, we extract the attribute-based features and the graph-based features and then use HNN model to fuse two kinds of features to identify fraudulent apps.

3.4.1. Features Extraction. Based on Section 2.3, we divide the log data for each app into 24 parts per day; that is, the time window is one hour. We calculate the ratio of records whose attributes take a certain value to all records in each time window, and we calculate them for each of 22 attributes in total, such as anonymized user id, advertisement id, country id, and device operating system. In addition, we calculate the ratio for browsing behavior and other actions on ads of users, respectively. Finally, we get 24 features for a time window (one hour), and the dimension of attribute-based features of each app is 24×24 for one day.

Based on Section 2.2.2 and Section 3.3, for the graph-based feature extraction, we construct the weighted heterogeneous graph of user-app-ad and then extract the graph-based feature through training by using WMP2vec. The dimension of graph-based features for each app is 32.

3.4.2. Comparison Models and Experiment Setup. We compare the proposed HNN with Support Vector Machine (SVM), Random Forests (RF), and Fully Connected Neural Networks (FCNN).

(i) SVM : SVM is an effective widely used two-class classification model. The RBF kernel is used and penalty parameter C is 0.9.

RF : RF is a well-known ensemble learning method that operates by constructing a multitude of decision trees at training time. The number of decision trees is 200 with depth of 5. Minimum samples split and minimum samples leaf are set to 5, respectively.

FCNN : FCNN is a fully connected neural network. The number of hidden layers is 4, with 100 neurons in each layer. The learning rate is 0.001 and the keep probability of dropout is 0.9.

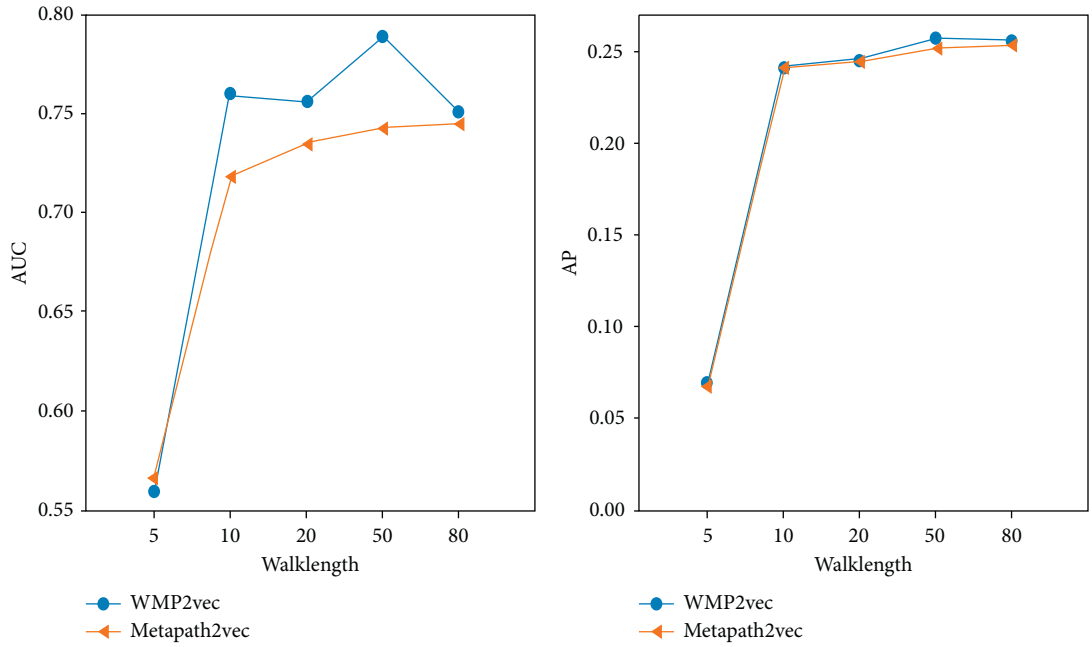

Figure 4: Comparing impacts of the different walk length with AUC and AP.

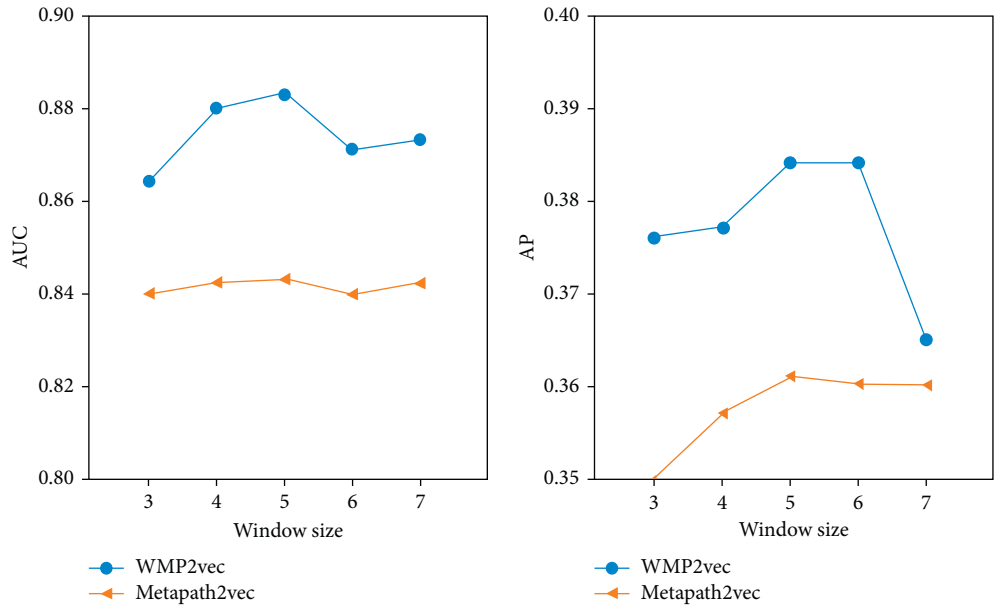

Figure 5: Comparing impacts of the different window size with AUC and AP.

HNN : HNN is the fusing model proposed in this study. The number of convolutional layers is 2, and the kernel size is 3×3. The number of fully connected layers is 2 with 100 neurons, using activation function "ReLU," and the keep probability of dropout is 0.9. The learning rate is 0.0001, the weight decay factor of learning rate is 0.98, and the batch size is 100.

In order to make sure that all models could learn the same knowledge from the dataset, when training the comparison models, we flatten the attribute-based features into a 576-dimensional vector. Furthermore, the vector is concatenated with graph-based features, and the dimension of total input vector is $576 + 32 = 608$.

We randomly divide the negative samples and positive samples of the dataset into three subsets 8 : 1 : 1, respectively, and combine the corresponding positive and negative example subsets into training (80%), validation (10%), and test (10%) sets. In order to handle the imbalanced category problem between fraudulent and nonfraudulent apps, we adopt upsampling technique during training.

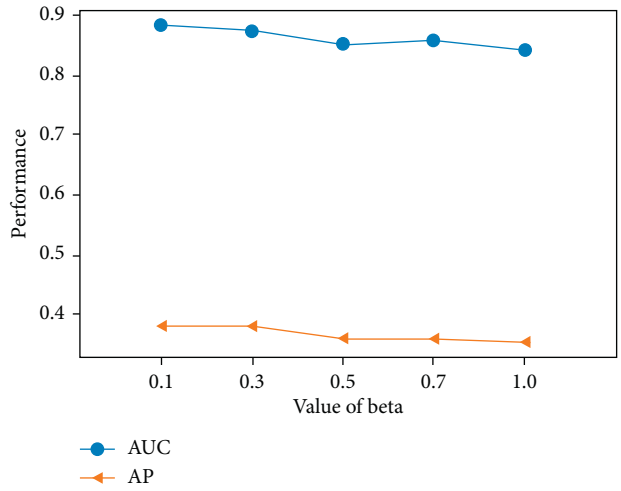

Figure 6: Comparing impacts of the weighted bias β with AUC and AP.

3.4.3. Experimental Results. The experimental results are shown in Tables 4 and 5. The HNN model proposed in this study reaches the highest AP value in six days and the highest AUC value in four days over all seven days. The FCNN, RF, and SVM models have similar performance to AUC measure, and Table 4 shows that HNN > FCNN > RF > SVM with AP measure. Thus, HNN outperforms all other models in terms of AP and AUC measures.

3.4.4. Comparative Experiments without Graph-Based Features. To show the contribution of graph-based feature extraction in proposed GFD approach, we remove the graph-based features in our dataset. When the proposed HNN model has only attribute-based features as input and no graph-based features as input, the HNN model leaves only the fully connected part to work, since the convolution part of HNN model has no input. This also means that the working HNN model would change to a fully connected neural network, that is, FCNN model, in this setting. So we use the SVM, RF, and FCNN models in this comparative experiment. The results are shown in Tables 6 and 7. Comparing the performances of models with/without graph-based features in Tables 4 and 5 and Tables 6 and 7, we could find that the FCNN model with graph-based features reaches better performance than the model without the graph-based features in both AP and AUC measures, while the performance improvement of SVM and RF models is not obvious with graph-based features.

3.4.5. Impacts of Parameters. (1). Time Windows t. Time window in attribute-based feature extraction of GFD approach decides the dimension of attribute-based features. We designed experiments to show the impact of time window, and the result is shown in Table 8. The size of time window is set to be 1, 3, and 6 hours. The continuous increase in size of time window makes HNN perform worse AP values. The other models seem to be not sensitive to the size of time window.

(2). Number of Convolutional Layers in HNN Model. We compare the effect of the number of convolutional layers of 1, 2, and 3 in HNN model and show the results in Table 9. The AUC and AP values achieve a high level when the number of convolutional layers is 2.

(3). Number of Fully Connected Layers in HNN Model. We set the number of fully connected layers to be from 1 to 4, and the experiment result is shown in Table 10. When the number of fully connected layers is 2, the HNN model reaches the highest performance.

(4). Activation Functions in HNN Model. We compare three well-known activation functions, ReLU, tanh, and Sigmoid, in HNN model, and the experiment results are shown in Table 11. The AUC values of the models with different activation functions are similar, and ReLU is slightly better than others. In terms of AP, ReLU is obviously better than the other two activation functions.

4. Related Work

Our work is related to existing studies on attribute-based fraud detection and graph-based fraud detection with machine learning. The challenges of fraud detection problem in mobile advertising system are summarized as accuracy requirement, throughput requirement, and the ability to combat the latest fraud methods [1].

Attribute-based fraud detection approaches have been used in fraud detection domain. Crussell et al. [26] built decision trees based on the features extracted from their dataset for classification. Liu et al. [27] proposed a binary SVM classifier to determine whether two UIs are likely to lead to equivalent states. This classification is used to simulate user interaction in the context of ad clicking. In order to classify malicious publishers, Mouawi et al. [11] evaluated KNN, SVM, and ANN based on features extracted from dataset, and the experimental results show that all three classifiers give very promising result. Haider et al. [2] proposed an ensemble-based method to classify each

Table 4: The comparison of models with AP.

Model	1st June	2nd June	3rd June	4th June	5th June	6th June	7th June
SVM	0.443	0.552	0.566	0.321	0.394	0.261	0.353
RF	0.547	0.329	0.645	0.505	0.428	0.636	0.489
FCNN	0.584	0.678	0.638	0.541	0.574	0.538	0.541
HNN	0.632	0.689	0.752	0.567	0.586	0.592	0.592

Table 5: The comparison of models with AUC.

Model	1st June	2nd June	3rd June	4th June	5th June	6th June	7th June
SVM	0.941	0.960	0.970	0.942	0.961	0.916	0.937
RF	0.919	0.942	0.949	0.937	0.944	0.938	0.945
FCNN	0.876	0.956	0.936	0.965	0.957	0.940	0.958
HNN	0.919	0.952	0.981	0.962	0.965	0.954	0.963

Table 6: AP of SVM, RF, and FCNN without graph-based features.

Model	1st June	2nd June	3rd June	4th June	5th June	6th June	7th June
SVM	0.389	0.547	0.449	0.393	0.304	0.357	0.366
RF	0.526	0.490	0.639	0.512	0.507	0.628	0.536
FCNN	0.433	0.605	0.636	0.528	0.547	0.526	0.517

Table 7: AUC of SVM, RF, and FCNN without graph-based features.

Model	1st June	2nd June	3rd June	4th June	5th June	6th June	7th June
SVM	0.889	0.944	0.947	0.950	0.951	0.930	0.938
RF	0.908	0.940	0.945	0.932	0.950	0.932	0.948
FCNN	0.907	0.950	0.931	0.940	0.933	0.913	0.951

Table 8: AUC and AP of HNN, FCNN, RF, and SVM with different time widows.

Model	AUC			AP		
	1 hour	3 hours	6 hours	1 hour	3 hours	6 hours
HNN	0.95	0.91	0.91	0.63	0.56	0.44
FCNN	0.88	0.89	0.87	0.58	0.56	0.54
RF	0.92	0.92	0.92	0.55	0.48	0.55
SVM	0.94	0.95	0.95	0.44	0.45	0.45

Table 9: AUC and AP of HNN with different number of convolution layers.

Model	AUC			AP		
	1	2	3	1	2	T3
HNN	0.922	0.942	0.950	0.605	0.630	0.405

Table 10: AUC and AP of HNN with different number of fully connected layers.

Model	AUC				AP			
	1	2	3	4	1	2	3	4
HNN	0.926	0.935	0.903	0.863	0.575	0.660	0.649	0.632

individual ad display as fraudulent or nonfraudulent. Gabriel et al. [28] evaluated the performance of logistic regression, gradient trees, and deep learning method in credit card fraud detection and proved that deep learning method outperforms the other compared methods.

Graph-based fraud detection approaches have been studied recently. Hu et al. [15] proposed a weighted graph propagation algorithm to identify the fraudulent apps in the user-app bipartite graphs. Vasumati et al. [29] applied decision trees to classify spam publishers based on constructed

Table 11: AUC and AP of HNN with different activation functions.

Model	AUC			AP		
	ReLU	tanh	Sigmoid	ReLU	tanh	Sigmoid
HNN	0.926	0.923	0.921	0.634	0.625	0.623

feature vector and computed spam score for each of the spam publishers by constructing a bipartite graph between users and publishers to find fraud publishers. What is more, the natural language processing (NLP) models known as Word2vec [23] have been applied to graph embedding, such as DeepWalk [10], Node2vec [21], and Metapath2vec [22]. Zheng et al. [30] proposed an unsupervised method to detect abnormal users and items through deep joint network embedding. Yu et al. [16] proposed a deep embedding approach for anomaly detection in dynamic networks by learning network representations which can be updated dynamically as the network evolves.

Mobile advertising fraud detection is still challenging; however, ensemble learning methods were usually the winner algorithms in fraud detection competition [10], and deep learning and graph learning are recently the most promising methods in this area.

There are two key differences between our proposed approach and existing works. First, we used app id, ad id, and user id from the real-world dataset to construct a weighted heterogeneous graph with these three types of nodes and proposed the graph embedding algorithm for mobile advertising fraud detection. The popular existing datasets, such as TalkingData dataset [31], usually have one or two types of entities (e.g., app id), so there are not enough entities to construct a heterogeneous graph as we did in this paper. Second, we proposed a fusing model to combine attribute-based and graph-based information for mobile advertising fraud detection by graph embedding and deep learning methods.

5. Conclusion

In this paper, we focus on the fraud detection problem in mobile advertising to detect fraudulent publishers. We propose a novel weighted heterogeneous graph and deep learning-based fraud detection approach, namely, GFD, to identify fraudulent apps for mobile advertising. Based on the relationship of users, publishers, and advertisement in mobile ad system, we construct a weighted heterogeneous graph and proposed a weighted metapath based graph embedding approach, named WMP2vec, to learn structural features of publishers in the graph. Furthermore, we construct a hybrid convolutional neural network to learn high-order features from attribute-based features and graph-based features. The experimental results in a real-world dataset show that our method is effective in classifying fraudulent apps for mobile advertising system.

There are two limitations in the work presented here. First, the dataset is limited to one mobile advertising dataset. In order to be more generalizable, it would be important to see whether the proposed GFD approach excels in more fraud detection datasets. Second, the dataset is limited to seven days. In the complex and dynamic online advertising environment, more time is still needed to evaluate the proposed approach.

Despite being focused on mobile advertising fraud detection in this presentation, the proposed GFD approach could be generalized to benefit many other online applications (e.g., e-commerce) that involve relationship between several types of entities. Future work should focus on the robustness and accuracy of our proposed model for other large-scale online datasets.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported in part by the Natural Science Foundation of Guangdong Province of China (Grant no. 2018A030313309), the Innovation Fund of Introduced High-End Scientific Research Institutions of Zhongshan (Grant no. 2019AG031), and the Fundamental Research Funds for the Central Universities, SCUT (Grant no. 2019KZ20).

References

[1] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna, "The dark alleys of madison avenue: understanding malicious advertisements," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, pp. 373–380, Vancouver, BC, Canada, November 2014.

[2] C. M. R. Haider, A. Iqbal, A. H. Rahman, and M. S. Rahman, "An ensemble learning based approach for impression fraud detection in mobile advertising," *Journal of Network and Computer Applications*, vol. 112, pp. 126–141, 2018.

[3] V. Dave, S. Guha, and Y. Zhang, "Measuring and fingerprinting click-spam in ad networks," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 175–186, ACM, New York, NY, USA, August 2012.

[4] H. Haddadi, "Fighting online click-fraud using bluff ads," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 21–25, 2010.

[5] A. Rodrigo, R. JGB de Queiroz and E. R. Cavalcanti, A proposal to prevent click-fraud using clickable captchas," in *Proceedings of the 2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, pp. 62–67, IEEE, Gaithersburg, MD, USA, June 2012.

[6] D. Antoniou, M. Paschou, E. Sakkopoulos et al., "Exposing click-fraud using a burst detection algorithm," in *Proceedings of the 2011 IEEE Symposium on Computers and*

Communications (ISCC), pp. 1111–1116, IEEE, Kerkyra, Greece, June 2011.

[7] W. Li, H. Li, H. Chen, and Y. Xia, "Adattester: secure online mobile advertisement attestation using trustzone," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 75–88, Florence Italy, May 2015.

[8] J. Kwon, J. Kim, J. Lee, H. Lee, and P. Adrian, "PsyBoG: power spectral density analysis for detecting botnet groups," in *Proceedings of the 2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE)*, pp. 85–92, IEEE, Fajardo, PR, USA, October 2014.

[9] M. Faou, L. Antoine, D. Décary-Hétu et al., "Follow the traffic: stopping click fraud by disrupting the value chain," in *Proceedings of 2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pp. 464–476, IEEE, Auckland, New Zealand, December 2016.

[10] R. Oentaryo, Ee-P. Lim, M. Finegold et al., "Detecting click fraud in online advertising: a data mining approach," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 99–140, 2014.

[11] R. Mouawi, M. Awad, C. Ali, H. Imad, H. El, and A. Kayssi, "Towards a machine learning approach for detecting click fraud in mobile advertizing," in *Proceedings of 2018 International Conference on Innovations in Information Technology (IIT)*, pp. 88–92, IEEE, Al Ain, United Arab Emirates, November 2018.

[12] G. S. Thejas, K. G. Boroojeni, K. Chandna, I. Bhatia, S. S. Iyengar, and N. R. Sunitha, "Deep learning-based model to fight against ad click fraud," in *Proceedings of the 2019 ACM Southeast Conference*, pp. 176–181, ACM, Kennesaw, GA, USA, April 2019.

[13] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proceedings of the ADKDD'17*, ACM, Halifax, NS, Canada, pp. 1–7, August 2017.

[14] G. S. Thejas, J. Soni, K. G. Boroojeni et al., "A multi-time-scale time series analysis for click fraud forecasting using binary labeled imbalanced dataset,"Proceedings of 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), vol. 4, pp. 1–8, IEEE, Bengaluru, India, December 2019.

[15] J. Hu, J. Liang, and S. Dong, "iBGP: a bipartite graph propagation approach for mobile advertising fraud detection," *Mobile Information Systems*, vol. 2017, Article ID 6412521, 2017.

[16] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: a flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2672–2681, London, UK, July 2018.

[17] L. Bertrand, F. Braun, C. Olivier, and M. Saerens, *A Graph-based, Semi-supervised, Credit Card Fraud Detection System: International Workshop on Complex Networks and Their Applications*, Springer, Cham, Switzerland, 2016.

[18] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: a survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[19] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: a survey," *IEEE transactions on Big Data*, vol. 6, no. 1, pp. 3–8, 2018.

[20] Y. Dong, N. V. Chawla, and A. Swami, "Metapath2vec: scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 135–144, ACM, Halifax, NS, Canada, August 2017.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, https://arxiv.org/abs/1301.3781.

[22] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Proceedings of Advances in Neural Information Processing Systems*, NIPS, Lake Tahoe, Nevada, pp. 2265–2273, January 2013.

[23] P. Bryan, 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*, ACM, New York, NY, USA, pp. 701–710, August 2014.

[24] 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*, ACM, New York, NY, USA, pp. 855–864, August 2016.

[25] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," *Aistats*, vol. 5, pp. 246–252, 2005.

[26] J. Crussell, R. Stevens, and H. Chen, "Madfraud: investigating ad fraud in android applications," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 123–134, Bretton Woods, NH, USA, June 2014.

[27] B. Liu, S. Nath, R. Govindan, and J. Liu, "DECAF: detecting and characterizing ad fraud in mobile apps," in *Proceedings of 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pp. 57–70, Renton, WA, USA, April 2014.

[28] R. Gabriel, C. Stancil, M. Sun, S. Adams, and B. Peter, "Horse race analysis in credit card fraud—deep learning, logistic regression, and gradient boosted tree," in *Proceedings of 2017 Systems and Information Engineering Design Symposium (SIEDS)*, pp. 117–121, IEEE, Charlottesville, VA, USA, April 2017.

[29] D. Vasumati, M. Sree Vani, R. Bhramaramba, and O. Yaswanth Babu, "Data mining approach to filter click-spam in mobile ad networks," in *Proceedings of Int'l Conference on Computer Science, Data Mining & Mechanical Engg.(ICCDMME'2015)*, Bangkok, Thailand, April 2015.

[30] M. Zheng, C. Zhou, J. Wu, S. Pan, J. Shi, and Li Guo, "Fraudne: a joint embedding approach for fraud detection," in *Proceedings of 2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, Rio de Janeiro, Brazil, July 2018.

[31] Kaggle Inc, "Talking data adtracking fraud detection challenge can you detect fraudulent click traffic for mobile app ads?," 2018, https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data.