Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 3759607, 11 pages
https://doi.org/10.1155/2019/3759607

Research Article

Detecting Fraudulent Bank Account Based on Convolutional Neural Network with Heterogeneous Data

Fang Lv, Wei Wang, Yuliang Wei, Yunxiao Sun, Junheng Huang, and Bailing Wang

Harbin Institute of Technology at Weihai, 2 West Wenhua Road, 264209 Weihai, China

Correspondence should be addressed to Bailing Wang; wbl@hit.edu.cn

Received 7 August 2018; Revised 13 February 2019; Accepted 4 March 2019; Published 25 March 2019

Academic Editor: Roberto Caldelli

Detecting fraudulent accounts by using their transaction networks is helpful for proactively preventing illegal transactions in financial scenarios. In this paper, three convolutional neural network models, i.e., NTD-CNN, TTD-CNN, and HDF-CNN, are created to identify whether a bank account is fraudulent. The three models, same in model structure, are different in types of the input features. Firstly, we embed the bank accounts' historical trading records into a general directed and weighted transaction network. And then, a DirectedWalk algorithm is proposed for learning an account's network vector. DirectedWalk learns social representations of a network's vertices, by modeling a stream of directed and time-related trading paths. The local topological feature, generating by accounts' network vector, is taken as input of NTD-CNN, and TTD-CNN takes time series transaction feature as input. Finally, the two kinds of heterogeneous data, being integrated into a novel feature matrix, are fed into HDF-CNN for classifying bank accounts. The experimental results, conducted on a real bank transaction dataset, show the advantage of HDF-CNN over the existing methods.

1. Introduction

According to Cornell University Law School (CULS) [1], bank fraud is defined as "whoever knowingly executes, or attempts to execute, a scheme or artifice (1) to defraud a financial institution; or (2) to obtain any of the moneys, funds, credits, assets, securities, or other property owned by, or under the custody or control of, a financial institution, by means of false or fraudulent pretenses, representations, or promises". Hence, bank fraudulent activities contain but not limit to money laundering, illegal pyramid selling, and illegal fund-raising. The activities are mainly involved in financial flows across bank accounts. Identifying the fraudulent accounts from massive bank accounts is of great significance in cracking down economic crime activities. In this paper, we call this as Fraud Account Detection (FAD) as below. Investigating large volume of financial transactions to identify fraudsters cannot be done manually for its heavy costs both in time and labor. Therefore, automatic FAD has attracted researchers' interest increasingly.

A FAD issue can be regarded as a problem of binary classification of accounts. In order to upgrade the classification performance, deep learning technique is employed to address the FAD issues on the basis of analyzing the transaction trading behaviors. $D \subseteq C \times B$ is used to denote the dataset that includes the whole bank account information, where $C = \{c_1, c_2, \ldots, c_n\}$ is the information set of bank accounts and $c_i = (x_1, x_2, \ldots, x_m)$ means the m-dimensional feature vector of the ith account. Moreover, the set $B = \{T, F\}$ is utilized as the label set of FAD, in which T and F represent abnormal label and normal label, respectively. Therefore, the target of a FAD task is to assign a correct label to a bank account from B.

Back in the 1960s, the biology research of Hubel et al. [2] shows the transfer process of visual information from retina to brain, which is accomplished by the activation of multiple receptive field. In recent years, deep learning method has been employed in extensive areas, i.e., image processing [3], Natural Language Processing (NLP) [4], network classification [5], and other fields. The reason why CNN architecture has been widely used can be listed as follows: (1) its flexible structure is easy to transfer into other scenarios and (2) CNN extracts the feature automatically. The good scalability

of CNN structure makes it successful to address many classification problems. In a specific classification scenario, one can tune the structural feature settings of CNN, e.g., the layer numbers, the neuron numbers of each layer, the types of pooling functions, and activation functions, to achieve the best performance.

Having abstracted the bank accounts into vertices and their transaction relationships into directed edges, the trading behaviors of accounts can be formed into a directed and weighted network. The transaction relationship information and time series information of bank accounts are embedded into the generated network. As mentioned above, CNN models obtain excellent performance in time series classification and social network. The superiority in convolution kernel and structural design inspires us to employ CNN framework in FAD issue. Therefore, with the labelled data provided by economic investigation experts, three convolutional neural network (CNN) models are proposed to address the FAD issue. The models are listed as follows. (1) A CNN model uses network topological data (NTD) being called NTD-CNN model. (2) A CNN model utilizes time series data (TTD) being referred to as TTD-CNN model. (3) A CNN model employs the two kinds of heterogenous data features (HDF), which are extracted from the former two kinds of data, being short for HDF-CNN model. The experiments on a real dataset, containing illegal pyramid selling accounts, demonstrate the effectiveness of our three CNN models. Except for the TTD-CNN, the other two CNN models achieve better performance than traditional abnormal detection method regarding precision, sensitivity, and F1-score. In summary, the classification performance of HDF-CNN is much better than that of the other three methods. To the best of our knowledge, this is the first time that CNN is applied to this application domain.

The main contribution of this paper can be listed as follows.

(i) This paper establishes a general account transaction network mathematical model, embedding the transaction relationships and timestamp information, to represent accounts' historical trading behavior. The network is used as the foundation of the learning network vector of accounts.

(ii) This paper proposes a DirectedWalk algorithm to learn the accounts' network vector. DirectedWalk quantifies the network local topological structures of transaction network into high dimensional vectors.

(iii) This paper devises three CNN models in detecting fraudulent bank account, including HDF-CNN that classifies account by using the conjunction of its two kinds of heterogenous data.

(iv) The experimental results show that HDF-CNN achieves the best classification performance.

The rest of the paper is organized as follows. The related work is presented in Section 2. Section 3 gives the classification features and proposes three CNN models in turn. Performance evaluation on the proposed models is analyzed in Section 4. Section 5 concludes the paper, and finally Section 6 describes our future work.

2. Related Works

There rarely have been specific researches on quantitative study of bank account transaction activities, but most studies mainly focus on anomaly activity detection by using historical transaction records.

Zhu [6] develops a new empirical mode decomposition method by using information from the same group to detect financial suspicious transaction time series data. The model shows its superiorities in analyzing nonlinear and nonstationary stochastic time series data. However, the influence of the transaction relationships is ignored in describing the suspicious. Wang et al. [7] introduces decision tree to create determination rules of trading risk by using the customer profiles, which come from a commercial bank in China. The built classification model benefiting from the determination rules achieves high recall rate.

Yang [8] establishes a central model of criminal networks for discovering its Core-figure, with the help of the Fisher Discriminant Analysis method. In order to control and reduce the loan risk, Cao et al. [9] construct a loan classification model, by using an improved optimization technique and a multiclass support vector machine (SVM) [10] method. Ling et al. [11] build a customer credit scoring model, by using a novel multikernel function and a Chaos particle swarm optimization method. A multilayer neural network and the SVM method are utilized in [12] for predicting if a loan applicant can be classified as solvent or bankrupt. To detect the credit-card fraudulent activities, in [13], the transaction aggregation strategy is expanded into a comprehensive strategy, which combines transaction timestamps and spending behavioral patterns. In order to identify illegal pyramid selling network with social behavioral data, Li et al. [14] model the ego-network of different type of users, e.g., regular users and illegal pyramid scheme users. The authors take the structural property into consideration in the process of analyzing illegal pyramid network statistical features. Reference [15] proposes a novel framework for creating accounts' trading behavior profiles and detects the members of the pyramid schemes by using anomaly detection methods. The features used are extracted from the sequential transaction records of an account. Four kinds of anomaly detection methods are tested in [15], showing that the IForest [16] is the best method in this scenario. Taken the financial transaction relationship between clients into consider, Ma [17] advances a novel algorithm for discovering anomaly groups, by mining closed-loop client transaction relationships, in the financial network.

Summary. Many techniques and algorithms have been proposed for dealing with fraud detection. However, most of the existing feature extraction methods, designing for specific fields, have disadvantages in transferring to other scenarios. Therefore, this paper attempts to break the bottleneck by using CNN architecture.

3. CNN for FAD

This section firstly defines the problem definitions. And then, we propose a DirectedWalk algorithm for learning network vector of the accounts. Subsequently, three CNN models, feeding with different kinds of features, are devised for classifying bank accounts.

3.1. Problem Definitions. This part gives the definitions of fraudulent account, bank accounts transaction network, and neighbor vertex set in turn.

Definition 1 (fraudulent account). A bank account v is called a fraudulent account if it is mainly used to process money that is relevant to illegal transaction activities.

The illegal transaction activities include illegal pyramid selling, illegal money laundering, and illegal financing.

Given a nonempty finite set $V = \{v_1, v_2, \ldots, v_n\}$, where $v_i \in V$ denotes a bank account, for $\forall v_i, v_j \in V$, each transaction record, containing transaction accounts, timestamp and amount, can be represented as a four-tuple (v_i, v_j, t_h, r_h). The four-tuple means transferring amount r_h from v_i to v_j on time t_h. The whole transactions between v_i and v_j within a time period can be denoted as $(v_i, v_j, \mathbf{ts_{ij}})$, where $\mathbf{ts_{ij}} = ((t_{ij}^{(1)}, r_{ij}^{(1)}), (t_{ij}^{(2)}, r_{ij}^{(2)}, \ldots, (t_{ij}^{(c)}, r_{ij}^{(c)}))$. Here, $\mathbf{ts_{ij}}$ means the time series transactions from v_i to v_j, and c is the total number of transactions happened during the $t_{ij}^{(1)}$ to $t_{ij}^{(c)}$ time period.

Taken accounts as vertices, transaction relationships between accounts as directed edges, and transaction information as edge weight, the time series transaction data can be created into a directed and dynamic network. We define the network G as follows.

Definition 2 (bank account transaction network). Given $G = (V, E, W, T)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set, $E = \{e_{ij}\}$, $1 \le i \le n, 1 \le j \le n$ denotes the set of edges. The $e_{ij} =< v_i, v_j >$ represents the directed weighted edge from v_i to v_j, if at least one transaction from v_i to v_j occurred. Then, we define $W = \{\mathbf{w}_{ij}\}$ as the set of all of the weight information, and $T = \{\mathbf{t}_{ij}\}$ means the set of all of the timestamp information, where $\mathbf{w}_{ij} = (r_{ij}^{(1)}, r_{ij}^{(2)}, \ldots, r_{ij}^{(Nt)})$, and $\mathbf{t}_{ij} = (t_{ij}^{(1)}, t_{ij}^{(2)}, \ldots, t_{ij}^{(Nt)})$, $\mathbf{w}_{ij} \in (R^+)^{Nt}$ is the weight vector of e_{ij}, R^+ is the set of positive real numbers, Nt is the total number of transactions, $r_{ij}^{(k)}$, $k = 1, 2, \ldots, Nt$, means the kth transaction amount from v_i to v_j, and $t_{ij}^{(k)} \in \mathbf{t}_{ij}$ is the timestamp of $r_{ij}^{(k)}$.

Definition 3 (neighbor set). Given G, and neighbor radius h, for $\forall v \in V$, $N_{v+}^{(h)} = \{w \mid< w, v >\in E\}$ represents its h step neighbor vertices existing in incoming transactions, and $N_{v-}^{(h)} = \{u \mid< v, u >\in E\}$ means the reverse vertex set. Therefore, the neighbor vertex set of v is denoted as $N_v^{(h)} = N_{v+}^{(h)} \cup N_{v-}^{(h)}$.

3.2. CNN with the Topological Data of Transaction Network. Economic crime investigators believe that it is helpful to consider the vertex itself and its neighbors comprehensively in determining whether a vertex is fraudulent. That means, for any v in G, which category it belongs to is closely related to its local topological structure G_v.

We propose a DirectedWalk algorithm, an improvement version of DeepWalk [18] in directed and dynamic network, to learn the network vectors of accounts. Based on this, a method to generate the local topological feature matrix of an account is presented by constructing its social relationships. And, finally, we devise a CNN framework with network topological data, which is called NTD-CNN for short.

3.2.1. DeepWalk. To capture the network topological information, [18] proposes a DeepWalk approach, which learns features that describe the graph structure. The optimization techniques, originally designed for language modeling, are used for learning social representations of a graph's vertices. DeepWalk takes a graph as input and produces its latent structure representation as an output. The learned structural features are used in many applications such as network classification and anomaly detection with outperform results.

DeepWalk algorithm, borrowing the concept in language modeling, considers a set of short truncated random walks as its own corpus, and the graph vertices as its own vocabulary. There are two main components, a random walk generator and an update procedure. Given graph G as input, the random walk generator samples uniformly a random vertex v as the root of the random walk W_v. For each vertex v, being the start of γ random walks, $|W_v|$ vertices are selected in order randomly. Aiming at updating the vector representation of v, Skip-Gram [19] is utilized to maximize the probability of its neighbors in the walk W_v. Inspired by DeepWalk, we propose a DirectedWalk algorithm to learn network vectors of accounts in the transaction network.

3.2.2. Learning the Network Vector. We generate a corpus and a vocabulary from the directed and dynamic network, which is the only required input for learning the network vectors. Give $G = (V, E, W, T)$, the vertex set V is considered as its own vocabulary, and the directed sequential transaction paths are seen as its own corpus. It is well known that, in DeepWalk, the relationship strength of two vertices is determined by the frequency that the two vertices occur in an adjacent position in the random walks. Here, as defined in Definition 2, the strength of the relationship between any two vertices v_i and v_j is determined by weight $\mathbf{w}_{ij}$ of e_{ij} and $\mathbf{w}_{ji}$ of e_{ji}. Moreover, in G, the order of vertices that passed by a transaction path are not random but time-related. Therefore, we propose a DirectedWalk algorithm for learning the network vector of the transaction vertices. Each vertex can be seen as the start vertex of a directed walk with a maximum length of l. For the last visited vertex v, the walk will pass over all of its directed neighbors, which have occurred in at least one transaction with v within a time interval τ. The walk grows up iteratively until not satisfying the constraints.

Line (2)-(18) in Algorithm 1 shows the core of our algorithm. The outer loop specifies the paths that start with each vertex and generates a time series ordering of the

Require:
network $G = (V, E, W, T)$
maximum walk length l, timestamp interval τ
window size w, embedding size d
Ensure:
matrix of vertex rpresentations $\Phi \in R^{|V| \times d}$
(1) the directed corpus NC is initialized to be empty;
(2) **for** $v_i \in V$ **do**
(3) $p_i = [v_i], p_i \subseteq P_i$
(4) **while** $\exists p_i.state = True$ **do**
(5) NodeSentence $p = P_i$.pop() (pop a p in its active state)
(6) $v = p.lastvertex, t = v.timestamp$
(7) retrieve v's directed neighbor vertices $((v_1, t_1), \ldots, (v_m, t_m))$ from G
(8) **for** $j \in m$ **do**
(9) **if** $(\exists t_k \in t_j, t < t_k < t + \tau)$ or $(len(t_j) = 1$ and $t_j^{(0)} > t)$ **then**
(10) p adds a new vertex (v_j, t_j)
(11) **if** v_j has no directed neighbor vertex or $len(p) = l$ **then**
(12) $p.state$ = False
(13) **end if**
(14) $P_i = P_i.push(p)$ (push p into P_i)
(15) **end if**
(16) **end for**
(17) **end while**
(18) **end for**
(19) $NC = NC \bigcup P_i$
(20) **for** $c \in NC$ **do**
(21) $SkipGram(\Phi, c, w)$
(22) **end for**

Algorithm 1: DirectedWalk(G, l, τ, w, d).

directed neighbor vertices. The state factor of p_i is initialized to $True$ and will be reset to $False$, in condition that its last vertex has no directed neighbor vertex or its length attains l. On condition that all of the paths reach their $False$ states, the generation procedure of P_i starting at v_i is completed. As depicted in line (8)-(16), for the last passed vertex v, the timestamp information is used to determine whether a vertex will be appended to v. In summary, fixing a start vertex, the longer the weight vectors of the passed directed edges are, the more walks will be produced by our DirectedWalk approach. Therefore, the frequent traders are more likely to exist in walks and appear in a window with high probability.

It is shown in line (21) that the Skip-Gram model [19] is exploited to maximize the cooccurrence probability Φ among the vertices in the time ordered walks. Skip-Gram, using the independence assumption, iterates over all possible collocations in directed corpus NC within the window w.

We finally obtain the optimal network vector representation of the vertices by using the same learning process of DeepWalk. DirectedWalk maps the directed and dynamic transaction network into a d-dimensional vector space. The vertices that contain similar local directed topological structures are mapped into adjacent vectors.

3.2.3. Constructing the Local Topological Feature Matrix. For $\forall v \in V$, given $N_v^{(h)}$ in G, the local topological structure of vertex v is the spanning subgraph G' of vertex set $N_v^{(h)} \cup \{v\}$. As mentioned above, the structure of v in G' is embedded into a d-dimensional network vector $\mathbf{v}^t$. We calculate the Euclidean distance between $\mathbf{v}^t$ and $\mathbf{v}_i^t \in N_v^{(h)}$ by using their network vectors. The size of set $N_v^{(h)}$ is denoted as l_v. Therefore, the $l_v + 1$ vertex vectors are spliced into a matrix by ranging their distances with v in ascending order. This yields the local topological feature matrix T_v as follows.

$$T_v = \left(\mathbf{v}_0^t, \mathbf{v}_1^t, \mathbf{v}_2^t, \ldots, \mathbf{v}_{l_v}^t\right)^T, \quad T_v \in R^{(l_v+1) \times d} \tag{1}$$

where $\mathbf{v}_i^t$ $(1 \le i \le l_v)$ is the network vector of $v_i \in N_v^{(h)}$, while $\mathbf{v}_0^t$ denotes the network vector of v itself.

3.2.4. Creating the NTD-CNN Framework. This part establishes a NTD-CNN classification framework, which is input with accounts' local topological feature matrices. NTD-CNN is composed of the following six elements, as shown in Figure 1.

(1) Input layer: for $\forall v \in V$, its input matrix is defined as matrix T_v (given in Formula (1)). In CNN model, the input matrix is seen as the pixel matrix of an image. Therefore, the size of the input image equals that of the input matrix. Based on the introduction above, matrix T_v embeds the local topological structure of vertex v with each row expressing a network vector. This paper keeps the row of T_v on a fixed value $L = \max_{v \in V}(d_{in} \bigcup d_{out})$, where $d_{in} \bigcup d_{out}$ corresponds to the maximum union of in-degree set and out-degree set for all

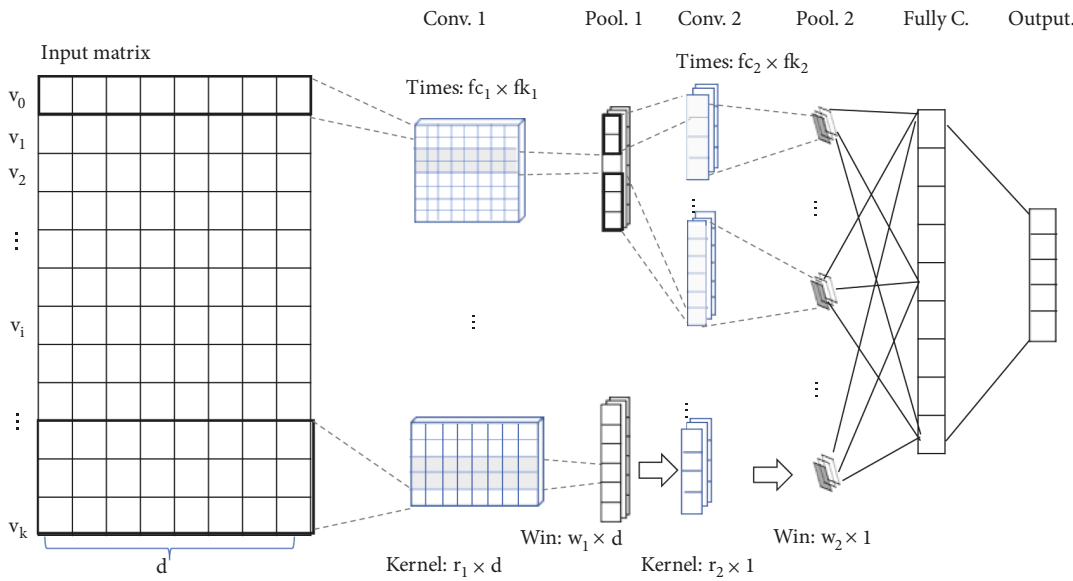


Figure 1: NTD-CNN classification model inputs with the local topological matrices of bank accounts. There are two convolution layers and each is followed by a pooling layer. *Conv*.1 denotes the first convolution layer, which contains fc_1 kinds of convolution kernels. Each kernel, in size of $r_1 \times d_h$, generates fk_1 feature maps. *Pool*.1 represents the first pooling layer, which reshapes the size of a feature map but does not change the number of feature maps. The window size, in *Pool*.1, is $w_1 \times d_h$. In *Conv*.2, each kind of input feature map is calculated by fc_2 kinds of convolution kernels, in which each contains fk_2 convolution kernel. *FullyC*. means the fully connected layer and *Output*. is the output layer.

vertices in network G. If the size of neighbor set of v satisfies $|N_v| < L$, we supplement the $L - |N_v|$ empty rows with 0. The format of input image is shown in Figure 2(a), whose specific values of L and d are given in experimental section.

(2) Convolution layer: the input matrix T_v is used as the first layer feature map. Similar to the definition of semantic unit features (unigram, bigram, trigram) in NLP, we employ three kinds of convolution filters to extract n-vertex, e.g., single-vertex, two-vertex, and three-vertex, features from T_v. As shown in Figure 1, for the lth convolution layer, the processing function on each feature map is shown in formula (2),

$$X_i^l = f\left(X^{l-1} * K_i^l + \mathbf{b}_i^l\right) \tag{2}$$

where $f(\bullet)$ represents an activation function, the operator "$*$" expresses the convolution operation, X^{l-1} denotes an input feature map, $K_i^l, (1 \le i \le 3)$ denotes the ith convolution kernel, and $\mathbf{b}_i^l$, means the bias vector. The stride length for K_i^l is set as 1. And the sizes of r_1 and r_2 are the same and range from $\{1, 2, 3\}$. Moreover, the numbers of fk_1 and fk_2 are denoted as fc_1 and fc_2, respectively. Therefore, the number of output feature maps of lth layer is $fc_l \times fk_l$ times that of the input. To apply the CNN to the nonlinear classification problems, Rectified Linear Unit (ReLU) function is used in the process of generating output feature maps.

(3) Pooling layer: the processing procedure, on the feature map of the lth pooling layer, is shown in formula (3),

$$X_i^l = X_i^{l-1} \cdot P_i^l + \mathbf{b}_i^l \tag{3}$$

where operator "$\cdot$" represents a pooling function, which is used for downsizing the ith input matrix X_i^{l-1}. The notations P_i^l and $\mathbf{b}_i^l$ denote the weight matrix and bias vector, respectively. The number of feature maps is not changed by the pooling procedures.

(4) Fully connected layer. In this layer, the input feature maps, each in size of 1×1 and is obtained from the last pooling layer, are connected with each neuron of this layer. The fully connected structure compresses the input vector $\mathbf{V}^c$ into a shorted one $\mathbf{X}^{re}$, by using the following map function.

$$\mathbf{V}^{re} = f\left(\mathbf{V}^c \cdot W^{re} + \mathbf{B}^{re}\right) \tag{4}$$

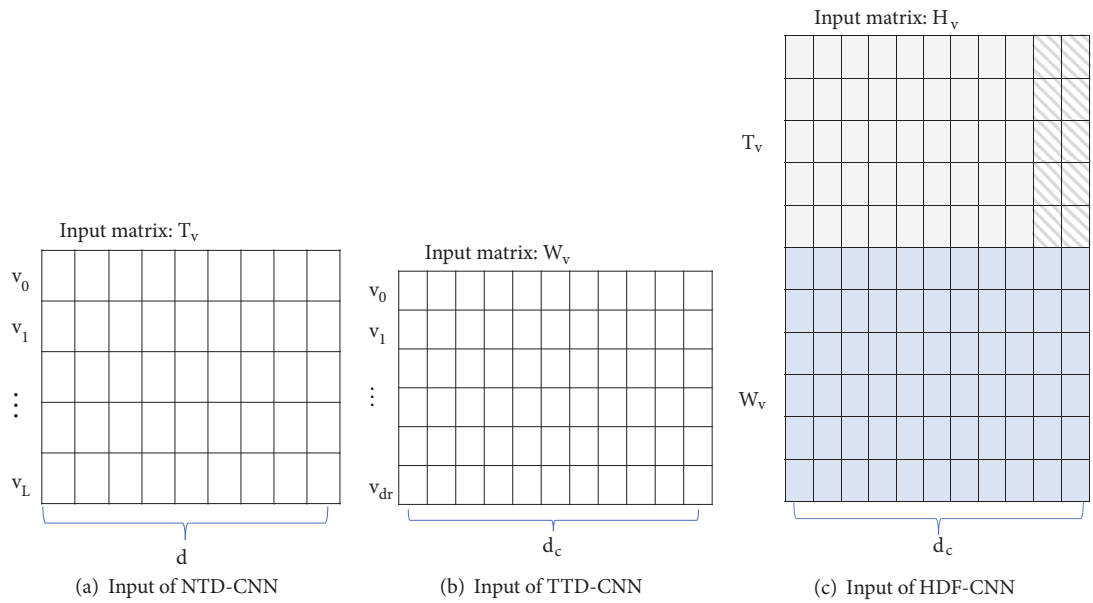
where W^{re} denotes the parameter matrix and $\mathbf{B}^{re}$ means the bias vector.

(5) Output layer: this layer is a fully connected layer, which contains two neurons and adopts softmax activation function. The softmax function is used to determine the category. For this reason, a vector V^{ou} is built through formula (5),

$$\begin{aligned} \mathbf{V}^{ou} &= \sigma\left(\mathbf{V}^{re} \cdot W^{ou} + \mathbf{V}^{ou}\right) \\ \sigma &= \frac{\exp\left(x_i\right)}{\exp\left(x_T\right) + \exp\left(x_F\right)} \end{aligned} \tag{5}$$

where $\mathbf{V}^{ou} \in R^2$ denotes the probabilities of a node belonging to each category, i.e., T and F. The $\sigma(\bullet)$ is defined in formula (6)

$$P\left(x_i \mid S\right) = \frac{\exp\left(x_i\right)}{\exp\left(x_T\right) + \exp\left(x_F\right)} \tag{6}$$



(a) Input of NTD-CNN (b) Input of TTD-CNN (c) Input of HDF-CNN

Figure 2: The input matrices of three CNN models.

where $i = T, F$, x_i denotes a category label in C. The category that node v belongs to is given by y', which is defined as follows:

$$y' = \arg\max_i \left(P\left(x_i \mid S\right)\right) \tag{7}$$

3.3. CNN with the Time Series Data of Transactions. Each transaction record of an account is composed of three kinds of information, two trading accounts, a timestamp, and transaction amount. All the transaction records that belong to one account can be classified into two parts, i.e., incoming transaction and outgoing transaction. By the opinion of economic investigators, accounts' historical trading information can be used as immediate evidence in detecting whether they are fraudulent. Thus, in this part, the time series transaction features are quantified into matrices and being input into a CNN classifying model, which is named TTD-CNN for short.

For $\forall v \in G$, its set of in-edge weight vectors is expressed by the matrix $W_{-v} = (\mathbf{w}_{1v}, \mathbf{w}_{2v}, \ldots, \mathbf{w}_{mv})^T$, including its whole incoming transactions, where m is the in-degree of vertex v. Each row of matrix W_{-v} is a vector, which is composed of the trading amounts ordering in time. The row vectors are spliced into matrix, according to the timestamps of their first trading records. That means, the earlier an account has transferred money to v, the upper it is arranged in W_{-v}. Given any vertex $u \in N^h_{v+}$, the weight vector of e_{uv}, which is defined in Definition 2, is shown in formula (8),

$$\mathbf{w_{uv}} = \left(r^{(1)}_{uv}, r^{(2)}_{uv}, \ldots, r^{(h_1)}_{uv}\right), \quad 1 \le u \le m \tag{8}$$

where h_1 is the total transaction times from u to v, and $r^{(j)}_{uv}$, $1 \le j \le h_1$, means the jth transaction amount from u to v. Similarly, the out-edge set vector $W_{v-} = (\mathbf{w}_{v1}, \mathbf{w}_{v2}, \ldots, \mathbf{w}_{vn})^T$ represents the whole outgoing transactions of v. Moreover, the row vectors of W_{v-} is sorted in the same way with that of W_{-v}. We denote the weight vector of e_{vi} as formula (9),

$$\mathbf{w_{vi}} = \left(r^{(1)}_{vi}, r^{(2)}_{vi}, \ldots, r^{(h_2)}_{vi}\right) \tag{9}$$

As shown in Figure 2(b), for any vertex v, the fixes-size time series feature matrix W_v is denoted as follows:

$$W_v = \begin{bmatrix} W_{-v} \\ W_{v-} \end{bmatrix}, \quad W_v \in R^{d_r \times d_c} \tag{10}$$

where $d_r = \max_{v \in V}(d_{in} + d_{out})$ is denoted as the maximum sum of in-degree d_{in} and out-degree d_{out}. d_c is the maximum number of Nt, being defined in Definition 2, of each vertex $v \in V$. And then, we pad any blank position in W_v with 0.

Finally, the time series transaction feature of vertex v is formed into matrix W_v. Utilizing W_v as the input matrix, a TTD-CNN model is devised for identifying whether an account is fraudulent. The specific values of d_r and d_c, which is obtained from statistical computing, are given in experimental section.

We ignore the architecture details of TTD-CNN for they are similar to that of NTD-CNN, which is depicted in Figure 1.

3.4. CNN with Two Kinds of Heterogeneous Data. The former two models (NTD-CNN and TTD-CNN) take advantage of two types of transaction information of an account, i.e., the local topological structure and time series transaction information, respectively. We reasonably think that full use of the two complementary node features will improve the accuracy of FAD. Therefore, we merge the former two kinds of heterogeneous data into an integrated feature matrix. The obtained matrix is used as the input of a CNN classifying model, which is named as HDF-CNN for short.

Given $\forall v \in V$, its local topological structure matrix T_v, and its time series feature matrix W_v, we generate a new feature matrix H_v as follows:

$$H_v = \begin{bmatrix} T_v \\ W_v \end{bmatrix}, \quad H_v \in R^{ro \times co} \tag{11}$$

where $T_v \in R^{(L+1)\times d}$ and $W_v \in R^{d_r \times d_c}$. Therefore, ro and co are constrained to fixed sizes as follows: $ro = L + d_r + 1$ and $co = \max\{d, d_c\}$, where, ro equals the sum of numbers of the two matrices' row vectors and co is set as the maximum of d and d_c.

Finally, the integrative feature matrix H_v is used as the input of HDF-CNN. As depicted in Figure 2(c), the input matrix means a picture of length co and width ro. The length of H_v is the larger one of that of T_v and W_v. Obviously, there are many empty positions in H_v, which is generated by formula (11). We pad the empty positions with 0. The shape of H_v, i.e., the input picture of HDF-CNN, is determined by that of T_v and W_v.

We ignore the architecture details of HDF-CNN for its structure is similar to that of NTD-CNN.

3.5. Training the CNN Models. In this paper, we optimize the three CNN models by employing the negative log-likelihood loss function. We aim to obtain the global optimum parameters, with which the most samples are predicted into their correct categories. It is the fact that, in this condition, the cost of the loss function reaches its minimum value. Therefore, we minimize the cost by formula (12). Notably, the main updating parameters of a CNN model are the weight of the map functions in each layer. This part denotes the weight parameters as w.

$$\begin{aligned} E\left(y, y'\right) &= -\sum_i y_i \log y_i' \\ Cost &= E\left(y, y'\right) + \frac{\lambda}{2n}\sum_w w^2 \end{aligned} \tag{12}$$

where y is the target label and y' means the predicted label. In this paper, y_i represents one-hot representation and y_i' denotes the predicted probability of each class, as defined in formula (7). The second term of *Cost* function is a regularization factor, being known as L2. L2 is an effective regularization method for avoiding overfitting. The overfitting problem, common in optimizing the deep learning models, means the model fits training set very well but does not work on testing set. As we all know, without a regularization term, the weight values will be relatively large, causing the cost function to fluctuate greatly in some small intervals. L2 regularization avoids overfitting by decaying weights of the cost function (12). Furthermore, another regularization method dropout is used in training our CNN models. Dropout avoids the overfitting problem by ignoring a proportion of hidden layer neurons randomly. The Minibatch Gradient Descent (MBGD) [20] method is adopted in minimizing the objective function. MBGD updates the network parameters with a batch samples one time. Our CNN model utilizes MBGD method for its fast training speed and high probability of discovering the global optimal value. The training process will stop under the condition that the loss of the objective function is stable at its minimal value.

Table 1: Characteristics of the dataset.

Dataset	Normal account	Fraud account
No. of accounts	9273	875
No. of edges	188359	39744
No. of transactions	6732730	275804

4. Experiments

4.1. Environmental Settings. Considering the volume of sample data is large and the number of deep learning parameter is numerous, GPU is adopted to improve the effect of matrix multiplication and convolution computation for its massively parallel processing. Since performing the former CNN framework on a single CPU is very time-consuming, to build up the experiment hardware environment, we use a deep learning server with GPU NVIDA LESLA P100 and 128G memory. TensorFlow is adapted to train the three CNN models.

4.2. Data Set. In recent years, we have been exploring computational models to classify bank accounts in combating illegal pyramid selling. The department of economic investigation provides us with plenty of transaction data of real bank accounts. We sample out the transaction records belonging to 10145 bank accounts to form out dataset for training our models. There are 9270 normal accounts and 875 accounts involving a multilayer marketing (MLM) organization, respectively. These MLM members are manually annotated as "abnormal" by economic investigators. Each record includes card id, timestamp of transaction, amount of a transaction, and direction of a transaction, i.e., revenue or expenditure. Before training the models, firstly, we filtered out some noisy data, i.e., deleting the duplicate records, incomplete records and the records whose transaction amounts no more than 50. And secondly, the fivefold cross-validation method is used to evaluate the trained network in all of the experiments. Table 1 summarizes the characteristics of the dataset.

As is usually adopted in neural network training, our models are trained with Minibatch Gradient Descent (MBGD) method. In each iteration, MBGD method takes a batch of samples, e.g., 100 accounts, randomly and updates the parameters with the average gradient value. Therefore, only the seed of MBGD will change when the input order changes, which means the performance of our model will keep stable within a reasonable range.

4.3. Experiments Settings. The determination of the best CNN values of parameters needs various experiments. In this paper, we refer to both related literatures and the certain dataset features to set and optimize those parameters.

Table 2: Characteristics of the dataset.

model name	input image size
NTD-CNN	460 × 200
TTD-CNN	571 × 1359
HDF-CNN	1031 × 1359

(1) The value of structural parameters in Figure 1: experimental results show that the classification property has little change when the length of network vector varies from 200 to 500. Thus, we set d as 200. In the three CNN models, the number of each kind of n-vertex convolution filter (i.e., fk_1 and k_2) is set as 100, and the number of neurons (i.e., the length of V^{re}) of the fully connected layer is set as 150. In our real transaction network, the maximum in-degree and out-degree (i.e., d_{in}, d_{out}) are 338 and 420, respectively. The maximum value of $d_{in} + d_{out}$ and $d_{in} \bigcup d_{out}$ are 571 and 459, respectively. In addition, d_c is set as 1359, which is also obtained from the statistical data. Therefore, the sizes of the input image of the three CNN models are listed in Table 2.

(2) Regularization parameter and dropout rate: these are two important parameters used to avoid model overfitting. In this paper, we adopt the most used regularization method named L2 and set its coefficient λ as 10e-4, as many CNN researches. Similarly, the value of dropout rate is set as a default value 0.5.

(3) Fixed learning rate: the convergence process of training will be too slow with a very small MBGD learning rate. Meanwhile, a too large learning rate will make the objective function fluctuate. It is the fact that the rate value is significant for CNN model converges to the global minimum. Therefore, our CNN models employ a dynamic strategy in learning rate selection. The strategy means the learning rate changes with the epoch time goes on. The value of learning is large on the beginning of training and is finally reduced to a minimum one. This paper trains the three CNN models with 100 epochs in order for them to have the same convergence time to attain the convergence condition. Therefore, during the top 80 epochs, the learning rate reduces 0.01 each 10 epochs. On arriving at the 80th epoch, if the current learning is larger than 0.01, the value will be reset to 0.01. And then, HDF-CNN decays once per epoch, using an exponential schedule. This paper tests the starting value from the set {0.01, 0.05, 0.09, 0.13, 0.17, 0.21, 0.25} in an decreasing order.

The experimental results show that HDF-CNN obtains the most stable convergence procedure.

We test the three CNN models on the dataset from two aspects, (1) comparing their classification performances with the traditional abnormal detection methods and (2) assessing the influence of additional convolution layers.

4.3.1. Comparing Methods in Classification Performance. We design four groups of experiments to compare the classification performance as follows.

(i) There are four traditional abnormal detection methods, i.e., isolation forest (IForest) [16], local outlier factor (LOF) [21], one-class SVM (One-SVM), and robust covariance (RC) [22]. These four are the most common methods for detecting financial fraudulent customers. We employ these four methods on the real dataset by using features extracted in [15]. To be specific, those features can be grouped into three categories: transaction statistical features, network behavioral features, and periodic behavioral features. Three values of a necessary threshold parameter, i.e., 0.01, 0.08, 0.1, meaning the outlier factor are selected in our experiments. The optimum factors for each method are shown in Table 3.

(ii) NTD-CNN model: the input is local topological feature matrices of bank accounts.

(iii) TTD-CNN model: the input is time series transaction feature matrices of bank accounts.

(iv) HDF-CNN model: the input is integrative feature matrices of bank accounts. An input matrix is obtained by concatenating the former two matrices, which are generated from two kinds of heterogenous information dependently.

In our experiments, both the hidden layer structure and the neuron number of each layer are all same in the three CNN models. The traditional methods are regarded as the baseline. The three CNN models are tested with many different parameter values and finally the best parameters are selected. The classification results on the test set are illustrated in Table 3.

As shown in Table 3, for the traditional automatic abnormal detection methods, IForest obtains the best results when the outlier factor evaluates to 0.08. It is indicated that the traditional automatic abnormal detection methods are not applicable on our dataset. This may be caused by the special fraud scenario. The fraudulent accounts act out different trading behaviors owning to their different rules in the MLM organization. Moreover, in this scenario, most of the abnormal accounts are normal when considering from their own angle, but are obviously abnormal by group perspective. Therefore, the four kinds of linear classification methods are too course-grained to find out the optimum classification boundary. Besides, the used features are heuristic, leading its performance to depend on expert knowledge. NTD-CNN maintains higher performances than IForest and TTD-CNN. Obviously, the features, being extracted from local topological structure data, are more useful than that of time series data. Meanwhile, it is illustrated that the proposed Directed-Walk algorithm is effective in representing the network vector of bank accounts. Utilizing the same type of data, TTD-CNN and IForest obtain similar classification performances. And, HDF-CNN achieves the best results, demonstrating our assumption that the two kinds of heterogeneous data are complementary. For a MLM member, involving in the real dataset, its fraudulent trading behaviors are hidden in its transaction relationships and its trading details. Considering that the only difference between the three CNN models is the input feature types, we can draw a conclusion that the integrative feature describes the category of an account

Table 3: Results of the classification methods.

Method	No.	Optimum factor	Precision(%)	Sensitivity (%)	F1-score(%)
LOF	1	0.1	23.71	25.42	24.54
IForest	2	0.08	64.39	57.46	60.73
One-SVM	3	0.08	32.75	41.87	36.75
RC	4	0.08	61.05	54.53	57.61
NTD-CNN	5	\	81.48	73.33	77.19
TTD-CNN	6	\	78.10	71.33	74.56
HDF-CNN	7	\	85.14	76.8	80.76

Table 4: The confusion matrix of HDF-CNN.

		Predicted class	
		abnormal	normal
Actual class	abnormal	149	26
	normal	45	1809

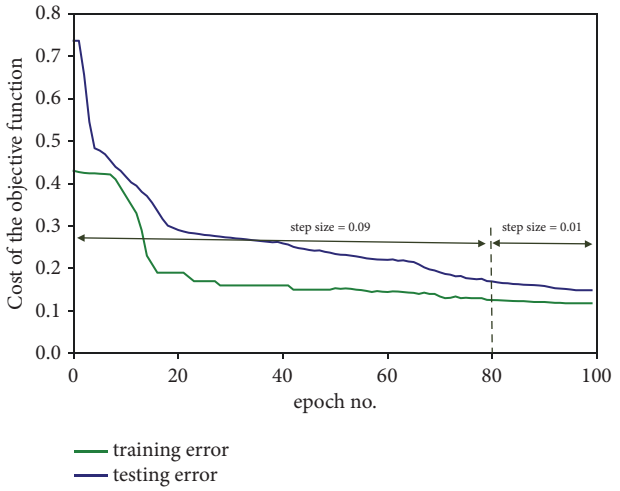


Figure 3: Training and testing errors for 100 epochs of HDF-CNN.

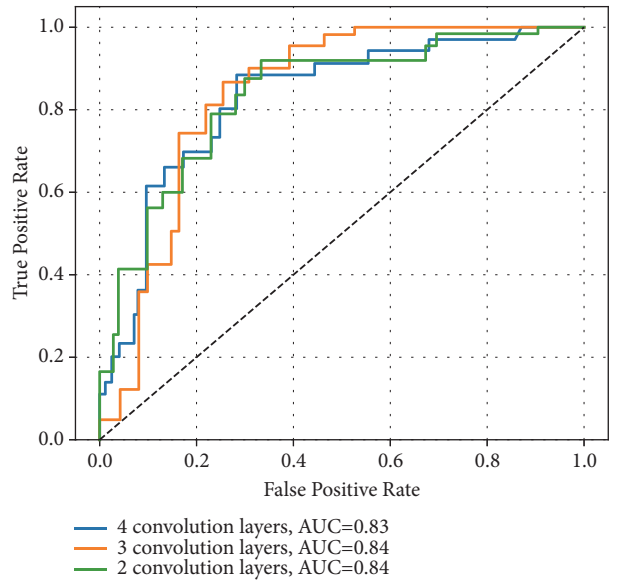


Figure 4: The ROC curves of the HDF-CNN model with different convolution layers.

best. It is worth noting that there exist significant differences among the precision and sensitivity in all of the four methods. Technically, the result means the number of false negative (FN) sample is more than that of false positive (FP). The confusion matrix of HDF-CNN is shown in Table 4.

Moreover, we input different random seeds in different orders in our experiments. It is proved that the order of input accounts does not affect the performance of CNN.

The changes of training and testing errors of HDF-CNN with increasing epoch times are depicted in Figure 3. The testing error starts to converge after 80 epochs. The training error keeps decreasing during the previous 80 epochs but almost converges to a constant value in the later 20 epochs. The testing error diminishes sharply originally and then fluctuates smoothly before the 80th epoch. With a smaller step size, the fluctuations almost disappear both on training error and testing error. Hence, the procedure to obtain smoothing testing error is to reduce the step size after a specific number of epochs.

4.3.2. Assessing Different Convolution Layers. As shown in Figure 1, HDF-CNN contains 2 convolution-pooling layers. This part investigates the effect of adding extra convolution layers. That means, the number of convolution-pooling layer of HDF-CNN is set as {2, 3, 4} for testing. The one or two additional layers are added following the last pooling layer in Figure 1. The ROC curves of HDF-CNN with different convolution layers are shown in Figure 4.

The value of AUC means the area under the curve. As well known, the greater the AUC value is, the better the classification performance of the model is. As shown in Figure 4, the classification effects of the HDF-CNN model achieve the best results with two and three convolution layers. It is found that the performance decreases when adopting four convolution-pooling layer structures. That means a more complex convolution structure will not improve the classification performance significantly. We conclude that the specific data features of transaction records, e.g., the amount of money, the number of counterparties, and local topological structure, are the reasons why a shallow structure is enough for HDF-CNN.

5. Conclusion and Discussion

In this paper, the CNN framework is applied in FAD problem. This paper proposes three CNN classification models, feeding with feature matrix getting from the historical transaction records, to detect the fraudulent accounts. The three CNN models are same in framework structure, but different in types of input matrices. This paper devises a DirectedWalk algorithm to learn the network vector of vertex, which is used for generating the local topological feature. DirectedWalk embeds the local network structure of an account into a high dimensional vector. It is found that the local topological feature is efficient in classifying fraudulent accounts. And the two kinds of heterogenous features, i.e., local topological feature and time series feature, are complementary in describing the classification category of an account. The experimental results on real dataset show that HDF-CNN achieves significant improvements when compared with other CNN models in classification performance.

6. Future Work

In recent years, automatic FAD has become one of the most concerning but challenging research topics. The two kinds of available information from local topological structure and time series transaction records are independent but interrelated. The way to better represent both kinds of information to reflect the account behavioral features more accurate is one of our future goals.

Long Short-Term Memory (LSTM) model has been proved very suitable for classifying time series data. Considering natural time series characteristic of transaction data, we have also adopted the LSTM model to do another experiment but achieved unsatisfactory classification results more than CNN with some unknown reasons. In the future, we will try to study and improve the LSTM model in order to adopt it to detect the fraudulent accounts.

Moreover, to solve the problem of less labelled samples, in the next step, we aim to add semisupervised strategy in our future work.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by National Key Research and Development Plan (No. 2016YFB0800802 and No. 2017YFB0801804), Frontier Science and Technology Innovation of China (No. 2016QY05X1002-2), and Key Research and Development Program of Shandong Province (No. 2017CXGC0706 and No. 2016ZDJS01A04).

References

[1] Cornell University Law School, "White-Collar Crime: an overview," http://topics.law.cornell.edu/wex/White-collar_crime.

[2] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, pp. 106–154, 1962.

[3] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, Boston, Mass, USA, June 2015.

[4] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics(ACL 2014)*, pp. 655–665, Baltimore, USA, June 2014.

[5] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the SIGKDD18: Proceedings of the 24th ACM SIGKDD International Conference*, London, UK, 2018.

[6] T. Zhu, "Suspicious financial transaction detection based on empirical mode decomposition method," in *Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)*, pp. 300–304, Guangzhou, China, December 2006.

[7] S. Wang and J. Yang, "A money laundering risk evaluation method based on decision tree," in *Proceedings of the 2007 International Conference on Machine Learning and Cybernetics*, pp. 283–286, Hong Kong, China, August 2007.

[8] L. Yang, "Based on social network crime organization relation mining and central figure determining," in *Proceedings of the 2012 IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS)*, pp. 55–58, Beijing, China, June 2012.

[9] J. Cao, H. Lu, W. Wang, and J. Wang, "A novel five-category loan-risk evaluation model using multiclass LS-SVM by PSO," *International Journal of Information Technology & Decision Making*, vol. 11, no. 4, pp. 857–874, 2012.

[10] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond*, The MIT Press, Cambridge, Mass, USA, 2003.

[11] Y. Ling, Q. Cao, and H. Zhang, "Credit scoring using multikernel support vector machine and chaos particle swarm optimization," *International Journal of Computational Intelligence and Applications*, vol. 11, no. 3, 2012.

[12] A. Karaa and A. Krichene, "Credit-risk assessment using support vectors machine and multilayer neural network models: a comparative study case of a tunisian bank," *Journal of Accounting and Management Information Systems*, vol. 11, pp. 587–620, 2012.

[13] A. Correa Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Feature engineering strategies for credit card fraud detection," *Expert Systems with Applications*, vol. 51, pp. 134–142, 2016.

[14] Li. YL, T. Zhou, and WB. Xie, "Uncovering the Illegal pyramid Networks by Big Data," *Big Data*, vol. 3, no. 5, pp. 106–112, 2017.

[15] F. Lv, J. Huang, W. Wang, G. Xin, and B. Wang, "Detecting pyramid scheme accounts with time series financial transactions," in *Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 722–728, Guangzhou, China, June 2018.

[16] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, pp. 413–422, Pisa, Italy, December 2008.

[17] J. Ma, "Complex funds network relationship discovery algorithm and its applifation in banking industry," *Financial Computer of China*, vol. 08, pp. 48–54, 2017.

[18] 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 (KDD 2014)*, pp. 701–710, New York, NY, USA, August 2014.

[19] T. Mikolov, K. Chen, and G. Corrado, "Efficient estimation of word representations in vector space," Computer Science, 2013.

[20] MD. Zeiler, "ADADELTA: an adaptive learning rate method," Computer Science, 2012.

[21] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '00)*, pp. 93–104, Tex, USA, June 2000.

[22] P. J. Rousseeuw and K. Van Driessen, "A fast algorithm for the minimum covariance determinant estimator," *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.