Semi-Supervised Node Classification With Discriminable Squeeze Excitation Graph Convolutional Networks

NAN JIA, XIAOLIN TIAN, (Member, IEEE), YANG ZHANG, AND FENGGE WANG
Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Xidian University, Xian 710071, China
Corresponding author: Xiaolin Tian (xtianscript@yeah.net)
This work was supported by the National Natural Science Foundation of China under Grant 61977052.

Abstract In recent years, Graph Convolutional Networks (GCNs) have been increasingly and widely used in graph data representation and semi-supervised learning. GCNs can reveal and dig deep into irregular data with spatial topological structure. However, in the task of node classification, most models will be over-smoothing (indistinguishable representations of nodes in different classes) after stacking multilayer GCNs. To alleviate the issue of over-smoothing, we propose a novel Discriminable Squeeze and Excitation Graph Convolutional Network (D-SEGCN) based on the attention mechanism of features for semi-supervised node classification. In the proposed D-SEGCN model, Squeeze and Excitation (SE) module is fused to the Graph Convolutional Networks to form SEGCN, which can selectively enhance significant features and realize the adaptive calibration of feature dimensions, thus enhancing node features. Then, the feature representation is obtained through a hierarchical structure, it is put into a discriminable capsule layer which can judge the similarities between the node features to obtain new feature. Finally, the feature representation and new feature are fused to obtain the final feature to strengthen the discriminability of nodes. Furthermore, we demonstrate that D-SEGCN significantly outperforms the state-of-the-art methods on three citation networks datasets.

Index Terms Aggregation, capsule network, graph convolutional networks, squeeze and excitation.

I. INTRODUCTION

Graph structure is a non-linear data structure. Graphs have many examples in real life, such as transportation networks, subway networks, social networks, state execution (automata) in computers, etc., which can be abstracted into graph structure. Classification of graph structure data is the foundation in many fields. Besides, node classification is an important and basic task in graph data mining. GCNs are often used to learn the expression of each node and finally classify. In reality, due to the tremendous amount of graph data and high labeling cost, we usually face a semi-supervised node classification scenario. For example, for the classification of semi-supervised nodes in a citation network, where nodes represent articles and edges represent citation relationships, the task is to predict the label of each article with a few labeled data.

The associate editor coordinating the review of this manuscript and approving it for publication was Hazrat Ali.

In recent years, as a new graph learning technology, Graph Neural Networks (GNNs) have attracted much attention. The concept of GNNs was first proposed in Gori et al. [1], and further was elaborated in Scarselli et al [2]. The early GNN mainly solved graph theory problems, such as the classification of molecular structure. But in fact, Euclidean space (such as image) or sequence (such as text), many common scenes can also be converted into graph, and then use GNNs technology to model. Based on graphic signal processing, Bruner et al. [3] proposed convolution neural network based on Spectral-domain and Spatial-domain.

The initiative of Spectral-domain Graph Convolution Networks is to implement convolution operations on topological graphs which are accomplished by Graph Fourier Transform. The Spectral-domain graph convolution network has been increasingly improved and expanded, including Spectral CNN [3], Chebyshev Spectrum CNN(ChebNet) [4], 1stChebNet [5], [6], and [7]. Significantly, Kipf and Welling et al. [5] determined the convolution network
structure through the local first-order approximation of Spectral-domain GCN, then node embedding is generated by the combined neighborhood information based on the scheme of model.

Since spectral methods are usually difficult to deal with large-scale graphs, Spatial-domain graph convolutional networks have developed rapidly recently. Spatial convolution can be compared to convolution performed directly on image pixels. Based on Spatial-domain convolution, its operation is directly defined on the connection relationship of each node, which is more similar to the traditional Convolutional Neural Networks. Micheli et al. [8] proposed the Neural Network for Graphs (NN4G), which was the first GCNs based on the Spatial-domain. NN4G was realized the convolution of the graph by directly accumulating the neighborhood information of the node, and was memorized the information of each layer through the residual connection. Based on the adjacency matrix exploration, Li et al. [9] proposed a model to learn the unknown hidden structural relationship through the adjacency matrix, and to construct a residual graph adjacency matrix by using a reasonable distance function with two nodes. Zhuang et al. [10] proposed the dual graph convolution network (DGCN) model, two convolutional neural networks were respectively devised to embed the local-consistency-based and global-consistency-based knowledge. In addition, Veličković et al. [11] proposed Graph Attention Networks (GAT), which used an aggregation function to fuse adjacent nodes, random walks, and candidate models in the graph to learn new representations. Thekumparampil et al. [12] proposed a semi-supervised GCN based on attention mechanism, and pointed out that the key of the extended model was the communication layer, not the perception layer. The attention mechanism tends to select neighbor nodes of the same category as the center node, and embedding it into the model will give the model strong correlation and interpretability. The Diffusion Convolution Neural Network (DGCN) was proposed by Atwood et al. [13], it regarded graph convolution as a diffusion process, which assumed that information transfers from one node to another with a certain transfer probability, so that the information distribution will reach equilibrium after several rounds. Peng et al. [15] proposed a novel clustering method by minimizing the discrepancy between pairwise sample assignments for each data point. As explained in [9], each GCN layer acted as a form of Laplacian smoothing in essence, which made the node features in the same connected component similar. The GraphSAGE model was proposed by Hamilton et al. [14] modified GCN from two aspects. On the one hand, the strategy of sampling neighbors transformed GCN with a full-graph training model to a node-centric small-batch training model; on the other hand, this algorithm extended the operation of aggregating neighbors. Peng et al. [16] proposed a novel subspace clustering approach by introducing a new deep model-Structured AutoEncoder (StructAE). The StructAE learns a set of explicit transformations to progressively map input data points into nonlinear latent spaces while preserving the local and global subspace structure. Thereby, adding too many convolutional layers will result in the output features being over-smoothing and render them indistinguishable. Some recent methods are proposed by Chen et al. [17], Xu et al. [18], and Ying et al. [19] try to get the global information through deeper models, some of which are unsupervised models and some need many training examples.

However, these GCN-based models rarely paid attention to the over-smoothing issue. In the node classification task, over-smoothing means similarity of representations among different classes’ nodes. The reason for over-smoothing is that multiple layers of GCNs are stacked. If the model is stacked with multiple GCN layers, the more neighbors the node aggregates. The interaction between higher-order neighbors brings noise and dilutes useful information, resulting in the final representation of all nodes tending to be consistent and difficult to distinguish.

In order to alleviate the over-smoothing issue, in this paper, we propose a novel architecture of Discriminative Squeeze and Excitation Graph Convolutional Networks, D-SEGCN for brevity, for node classification on graphs. Inspired by the flourish of applying deep architectures and the pooling mechanism into image classification tasks, we design a model with feature dimensions based on attention mechanism to enhance the node features. The model is a deep hierarchical network, the main idea of D-SEGCN is to acquire feature dimensions based on the attention mechanism, which can highlight the important features of nodes. As illustrated in Fig. 1, D-SEGCN mainly consists of eight SEGcn modules (each SEGcn module contains two SEGcn layers), four coarsening layers, four refining layers, a capsule network layer, and a softmax classifier layer. Taking the adjacency matrix $A$ and the feature matrix $X$ of graph $G$ as the network inputs.

The main advantages of the proposed D-SEGCN for semi-supervised node classification tasks are summarized as follows.

Firstly, compared with the previous work proposed by Hu et al. [18], the proposed model in the paper is fused SE module to GCN. By incorporating Squeeze and Excitation module to GCN, the network can use global information to selectively enhance the significant features and suppress the useless features. The proposed model realize the adaptive calibration of feature dimensions, and enhance the features of nodes. These operations can alleviate over-smoothing during the multi-level neighborhood aggregation.

Secondly, the feature representation is obtained through hierarchical structure and new feature from the discriminable capsule layer are fused, which can remembering the rich information in the graph structure well judging the similarities among features, to strengthen the discriminability of nodes. Thus effectively alleviating the over-smoothing problem.

Thirdly, the performance of the proposed model is tested on three citation networks datasets. The results show that
the D-SEGCN model is advanced and effective compared to state-of-the-art models.

II. PRELIMINARIES AND RELATED WORK

Let \( G = (V, E) \) be the input graph (weighted or unweighted), where \( V \) and \( E \) are the node set and edge set, respectively. Let \( A \) be the \( |V| \times |V| \) adjacency matrix of the graph with each entry \( A_{ij} \) denoting the weight of the edge between \( i \) and \( j \).

We denote the adjacency matrix by

\[
A_{ij} = \begin{cases} 
1 & \text{if } (v_i, v_j) \in E \\
0 & \text{otherwise}
\end{cases}
\]

which is nonnegative.

Using \( D = \text{diag}(d_1, d_2, \ldots, d_n) \) to define the degree matrix of \( A \), where \( d_i = \sum_j A_{ij} \) is the degree of node \( i \). Let \( X = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^{d \times n} \) be the node feature matrix, where \( d \) is the dimension of the attributive features. We use edge weights to indicate connection strength between nodes.

In the paper, we input undirected graph \( G_1 = (v_1, e_1) \), where \( v_1 \) and \( e_1 \) are respectively the set of \( n_1 \) nodes and \( e_1 \) edges, let \( A_1 \in \mathbb{R}^{n_1 \times n_1} \) be the adjacency matrix describing edge weights. For the D-SEGCN model, the graph sent to the \( j \)-th layer is represented by \( Q_j \) with \( n_j \) nodes. The adjacency matrix and hidden representation matrix of \( Q_j \) are represented by \( A_j \in \mathbb{R}^{n_j \times n_j} \) and \( P_j \in \mathbb{R}^{n_j \times d} \), respectively. Given the labeled node set \( V_L \) containing \( m \) far less than \( n_1 \) nodes, where each node \( v_j \in V_L \) is combination with a label \( y_j \in Y \), our object is to predict labels of \( V \setminus V_L \).

A. GRAPH CONVOLUTIONAL NETWORKS

In the past few years, there has been a surge of applying convolutions on graphs. GCN contains one input layer, several propagation (hidden) layers and one final perceptron layer. Given an input feature matrix \( X^{(0)} = X \) and adjacency matrix \( A \), GCN conduct the following layer-wise propagation in hidden layers as formula (1):

\[
X^{k+1} = \sigma(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X^{k}W^{k})
\]

where \( k = 0, 1, \ldots, K - 1 \) and \( D = \text{diag}(d_1, d_2, \ldots, d_n) \) is a diagonal matrix with \( d_i = \sum_j A_{ij} \), \( W^k \in \mathbb{R}^{d \times d_{k+1}} \), \( d_0 = p \) is a layer-specific weight matrix needing to be trained. We denote \( \sigma(\cdot) \) as an activation function, such as \( \text{ReLU}(\cdot) = \max(0, \cdot) \), and \( X^{k+1} \in \mathbb{R}^{n \times d_{k+1}} \) as the output of activations in the \( k \)-th layer. For semi-supervised classification, GCN [5] defines the final perceptron layer as formula (2):

\[
Z = \text{softmax}(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X^{K}W^{(K)})
\]

where \( W^{(K)} \in \mathbb{R}^{d \times o} \) and \( o \) denotes the number of classes. The final output \( Z \in \mathbb{R}^{n \times o} \) denotes the label prediction for all data \( X \) in which each row \( Z_i \) denotes the label prediction for the \( i \)-th node.

B. SQUEEZE AND excitation NETWORKS

Squeeze and Excitation Network (SENet) by Hu et al. [21] proposed a better feature representation structure, which can be utilized to represent features on input by learning branch structure. Structurally, branches are used to learn how to evaluate the correlation between channels, and then applied to the original feature map to achieve input calibration. In order to make the network measure the channel association through the global information, the global pooling is used to capture the global information in the structure, and then two full connection layers are connected to recalibrate the input, which can make the network learn a better representation. Squeeze and Excitation operations are described below.

C. SQUEEZE OPERATION

Squeeze operation first compresses the feature with dimension of \( H \ast W \ast C \) to features of \( 1 \ast 1 \ast C \) using global pooling to obtain the global information. Using formulas (3) and (4) for the specific operation, where \( v_c \) is the \( c \)-th convolution kernel, \( x^s \) represents the \( s \)-th input, \( u_c \in \mathbb{R}^{W \times H \times C} \), \( u_c \) represents the \( c \)-th \( H \ast W \) matrix in \( u \), and the subscript \( c \) represents the number of channels. The squeeze operation \( F_{sq}(\cdot) \) is a global average pooling, therefore, squeeze operation converts the input of \( H \ast W \ast C \) into the output of \( 1 \ast 1 \ast C \).

\[
u_c = v_c \ast x = \sum_{s=1}^{c} v_c^s \ast x^s \quad (3)
\]

\[
Z_c = F_{sq}(u_c) = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} u_c(i,j) \quad (4)
\]

D. EXCITATION OPERATION

Specifically, the excitation operation is to learn the importance of each feature channel automatically, and then enhance significant features and suppress the useless features for the current task according to the importance, similar to the attention mechanism. Generally speaking, the network can use global information to enhance the significant feature channels and suppress the useless feature channels selectively, so as to realize the adaptive calibration of feature channels. Using formula (5) for the specific operation, \( F_{ex} \) is the excitation function, i.e. the function of adaptive recalibration, \( \delta \) is ReLU.

Here, \( W_1 \) is the parameter of dimension reduction layer, and \( W_2 \) is the the parameter of the dimension-increasing layer, the result after previous squeeze operation is \( z, W_1z \) is a fully connected layer operation, \( W_1 \in \mathbb{R}^{r \times C}, r \) is a scaling parameter. The purpose of this parameter is to reduce the number of channels and thus reduce the amount of calculation. Here \( z \in \mathbb{R}^{1 \times 1 \ast C}, W_1z \in \mathbb{R}^{1 \times 1 \ast \frac{C}{r}} \), after passing a ReLU layer, the output dimension is unchanged. Then the dimension of \( W_2 \) is \( C \ast \frac{C}{r}, \) the dimension of the output is \( 1 \ast 1 \ast C \). Finally, the sigmoid function is used to obtain \( S \).

\[
S = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2\delta(W_1z)) \quad (5)
\]

III. DISCRIMINABLE SQUEEZE AND excitation GRAPH CONVOLUTIONAL NETWORK

A. THE OVERALL ARCHITECTURE

For a D-SEGCN network of \( l \) layers, the \( j \)-th SEGCN module layer first conducts a squeeze and excitation (SE) operation.
as formulated in formulas\((6)\), \((7)\) and \((8)\), then we perform GCN operation using formula\((9)\), finally we aggregate nodes with similar structure into super-nodes to generate a coarser graph \(G_{j+1}\) with fewer nodes and node embedding matrix \(P_{j+1}\). As illustrated in Fig. 1, D-SEGCN mainly consists of eight SEGCN modules (each SEGCN module contains two SEGCN layers), four coarsening layers, four refining layers, a capsule network layer, and a softmax classifier layer. Taking the adjacency matrix \(A\) and the feature matrix \(X\) of graph \(G\) as the network inputs. The SEGCN layer is first conducted to learn node representations. Due to the attention mechanism based on feature dimensions, some significant node features will be strengthened. Then a coarsening operation is performed to aggregate structurally similar nodes into super-nodes. After the coarsening operation, each super-node represents the native structure of the original graph, which can easily make use of the global structure of the graph. Along with the coarsening layers, the symmetric graph refining layers are applied to restore the original graph structure for node classification tasks. Finally, on the discriminable Capsule layer, the feature matrix is performed dynamic routing operation to form a new feature matrix. The acquired features and the final new features are fused to enhance the distinguishability of the nodes, thus alleviating over-smoothing issue by the continuous merging of nodes. It is worth noting that \(\oplus\) is an add operation. The graph \(G_{j+1}\) with \(p\)-dimensional potential representation of \(n_{j+1}\) super-node is generated in the coarsening layer of layer \(j\), refining layer is just the opposite. The corresponding adjacent matrix \(A_{j+1}\) and \(P_{j+1}\) will be sent to the \((j + 1)\)-th layer. Symmetrically, before the graph refining operation is executed, we firstly conduct SE operation and then refine the coarsened graph to restore the finer graph structure. In order to promote optimization in a deeper networks, we add shortcut connections [22] spanning each coarsened graph and its corresponding refined parts.

**B. SQUEEZE AND EXCITATION GRAPH CONVOLUTIONAL NETWORKS**

Graph Convolutional Networks have achieved promising generalization in various tasks, and our work is based on...
1) SQUEEZE OPERATION
Squeeze operation uses the global pooling operation, first compresses the \(n \times d\) to a \(1 \times d\), so as to obtain the information of global receptive field. See formula (6) for the specific operation, where \(X_j \in R^{N_j \times d_j}\), where \(d_j\) is the dimension of the attributable features in the \(j\)-th layer. \(X_i\) represents the \(i\)-th row of \(X\), that is the feature vector for the \(i\)-th node, \(N_j\) is the number of nodes for the \(j\)-th layer, \(Z_j \in R^{1 \times d_j}\).

\[
Z_j = F_{sq}(X_j) = \frac{1}{N_j} \sum_{i=1}^{N_j} X_i
\]  

2) EXCITATION OPERATION
Then there is just an excitation operation, which is akin to a gate mechanism in a recurrent neural network, giving each part a relevant weight value (learned), similar to the attention mechanism, see formula (7) for the specific operation. After getting the squeezed feature map, we use sigmoid function to excite it. Firstly, the dimension is reduced by a fully connected layer, i.e. \(W_1 Z\) in the following formula (7), after passing a ReLU function, we can obtain \(\delta(W_1 Z)\), the output dimensions are unchanged. Then we through the full connection to upgrade the dimension, i.e. \(W_2 \delta(W_1 Z)\) and finally sigmoid function is used to obtain \(S\).

\[
S_j = F_{ex}(Z_j, W) = \sigma(g(Z_j, W)) = \sigma(W_2 \delta(W_1 Z))
\]

3) REWEIGHT OPERATION
We regard the weight of the output of excitation operation as the importance of each feature after feature dimension selection, and then used to weight the previous feature dimension by dimension through multiplication to complete the recalibration of the original feature on the dimension. See formula (8) for the specific operation.

\[
\hat{X}_j = F_{scale}(X_j, S_j) = S_j X_j
\]

In this paper, at layer \(j\), let \(\hat{X}_j = P_j\), taking graph adjacency matrix \(A_j\) and feature matrix \(P_j\) as input, each SEGCN layer outputs a hidden representation matrix \(Q_j \in R^{n_j \times d_{j+1}}\), which is described as:

\[
Q_j = ReLU(D_{\hat{A}}^{-\frac{1}{2}} \hat{X}_j \hat{D}_{\hat{A}}^{-\frac{1}{2}} P_j W_j)
\]

where \(P_1 = \hat{X}^{(0)}, ReLU(\cdot) = \max(0, \cdot), \) adjacency matrix with self-loop \(\hat{A}_j = A_j + I, \hat{D}_j\) is the degree matrix of \(\hat{A}_j,\) and \(W_j \in R^{d_j \times d_{j+1}}\) is a trainable weight matrix.

C. GRAPH COARSENING
In the phase, the input graph \(G(G_0)\) is repeatedly coarsened into a series smaller graphs \(G_0, G_1, G_2, \ldots G_m\) such that \(|V_0| > |V_1| > |V_2| > \ldots > |V_m|\). For the graph coarsening procedure, we design the following two super clustering strategies to assign nodes with similar structures into a super-node in the coarsen graph. We first conduct structural equivalence clustering, followed by normalized heavy edge clustering. The number of coarsening layer is \(q\). Some work has been proposed for learning hierarchical information on graphs. Liang et al. [23] and Hu et al. [20] uses a coarsening procedure to build a coarsened graph of small size.

1) STRUCTURAL EQUIVALENCE CLUSTERING(SEC)
If two nodes share the same set of neighbors, they are considered to be structurally equivalent. We then assign these two nodes to be a super-node. For example, as showed in Fig. 3, nodes \(B\) and \(E\) are structurally equivalent, so these two nodes are merged in a super-node. We mark all structurally equivalent nodes and leave other nodes unmarked.

2) NORMALIZED HEAVY EDGE CLUSTERING(NHEC)
Heavy edge matching is a popular matching method for graph coarsening [24]. For an unmarked node \(u_k\), its heavy edge clustering is a pair of nodes \((u_k, u_m)\) such that the weight of the edge between \(u_k\) and \(u_m\) is the largest. Then, we propose normalizing the edge weight between the unmarked node pair.
Since it remains unmarked.

For one super-node \( v_j \), its edge weight to \( v_j \) is the summation over edge weights of \( v_j \)'s neighbor nodes contained in \( v_j \). The updated node weights and edge weights will be used in formula (10) in the next coarsening layer. In order to help restore the coarsened graph to the original graph, we preserve the matching relationship between nodes and their corresponding super-nodes in a matrix \( M_j \in \mathbb{R}^{n_1 \times n_{j+1}} \). Formally, at layer \( j \), entry \( m_{km} \) in the matching matrix \( M_j \) is calculated as the formula (11):

\[
m_{km} = \begin{cases} 1, & \text{if } v_k \text{ in } G_i \text{ is matched into } v_m \text{ in } G_{i+1}; \\ 0, & \text{otherwise.} \end{cases}
\]  

(11)

Note that \( m_{33} = 1 \) in this illustration, since node \( C \) constitutes its super-node by itself. Next, the hidden node feature matrix is determined as the formula (12):

\[
P_{j+1} = M_j^T Q_j
\]  

(12)

Then, we can generate a coarsen graph \( G_{j+1} \) whose adjacency matrix can be calculated as the formula (13):

\[
A_{j+1} = M_j^T A_j M_j
\]  

(13)

The coarsen graph \( G_{j+1} \) along with the resulting representation matrix \( P_{j+1} \) will be fed into the next layer as input. The graph coarsening procedure is summarized in Algorithm 1.

### D. The Graph Refining Layer

To restore the original topology of the graph and further facilitate node classification, we heap up the same numbers of graph refining layers as coarsening layers. As for the coarsening process, each refining layer consists of two steps: generating the node embedding vector and restoring the node representation.

Since we have saved the matching matrix during the coarsening process, we use \( M_{j-1} \) to restore the refined node representation matrix of layer \( j \). We further employ residual connections between the two corresponding coarsening and

\[
S(u_k, u_m) = \frac{A_{km}}{\sqrt{D(u_k)} \cdot \sqrt{D(u_m)}}
\]  

(10)

where \( A_{km} \) is the edge weight between \( u_k \) and \( u_m \), and \( D(\cdot) \) is the node weight.

We iteratively extract an unlabeled node \( u_k \) and calculate its normalized connection strength with all unlabeled neighbors. For the example in Fig. 3, node \( A \) is equally likely to be merged with node \( C \) and node \( D \), without edge weight normalization. With normalization, node \( A \) will be merged with \( D \), which is a better merging since \( A \) is more structurally similar to \( D \). Node pair \((A, D)\) are merged together to constitute a super-node. After that, since only node \( C \) remains unmarked, it is a super-node by itself.

3) A Hybrid Clustering Method

In this paper, we use a hybrid of two clustering methods above for graph coarsening. To construct \( G_{i+1} \) from \( G_i \), we first find out all the structural equivalence clustering (SEC) \( n_1 \) of \( G_i \). Nodes in each clustering are then merged into a super-node in \( G_{i+1} \). Note that some nodes might not be merged at all and they will be directly copied to \( G_{i+1} \). Then we need to be able to acquire all the super-nodes. Fig. 3 provides a toy example for illustrating the process.

The figure displays in the process of applying Structural Equivalence Clustering(SEC) and Normalized Heavy Edge Clustering(NHEC) for graph coarsening.

For one super-node \( v_j \), its edge weight to \( v_j \) is the summation over edge weights of \( v_j \)'s neighbor nodes contained in \( v_j \). The updated node weights and edge weights will be used in formula (10) in the next coarsening layer. In order to help restore the coarsened graph to the original graph, we preserve the matching relationship between nodes and their corresponding super-nodes in a matrix \( M_j \in \mathbb{R}^{n_1 \times n_{j+1}} \). Formally, at layer \( j \), entry \( m_{km} \) in the matching matrix \( M_j \) is calculated as the formula (11):

\[
m_{km} = \begin{cases} 1, & \text{if } v_k \text{ in } G_i \text{ is matched into } v_m \text{ in } G_{i+1}; \\ 0, & \text{otherwise.} \end{cases}
\]  

(11)

Note that \( m_{33} = 1 \) in this illustration, since node \( C \) constitutes its super-node by itself. Next, the hidden node feature matrix is determined as the formula (12):

\[
P_{j+1} = M_j^T Q_j
\]  

(12)

Then, we can generate a coarsen graph \( G_{j+1} \) whose adjacency matrix can be calculated as the formula (13):

\[
A_{j+1} = M_j^T A_j M_j
\]  

(13)
Algorithm 1 Graph Coarsening

**Input:** A input Graph $G_j$, node feature matrix $P_j$, adjacency matrix $A_j$ and levels for coarsening $q$

**Output:** Coarsened graph $G_{j+1}$ and node representation $P_{j+1}$ and adjacency matrix $A_{j+1}, j \in [0, q − 1]$

1. for $j = 0, 1, \ldots, q − 1$
   2. $η_1 \leftarrow$ all the structural equivalence clustering $G_j$
   3. Mark nodes in $M_1$ as marked
   4. $η_2 = \emptyset$
   5. Sort all unmarked nodes in ascending order according to the number of neighbors
   6. for each unmarked node $u_k$ do
      7. for each unmarked node $u_k$ adjacent to $u_m$ do
         8. Calculate $S(u_k, u_m)$ according to formula (10)
         9. $η_2 = η_2 \cup (u_k, u_m)$ and mark both as matched
     10. Update node weights and edge weights
     11. Construct matching matrix $M_j$ according to formula (11) and based on $η_1, η_2$
     12. Calculate node representation $P_{j+1}$ according to formula (12)
   13. return coarsened graph $G_{j+1}$, adjacency matrix $A_{j+1}$ according to formula (13) and node representation $P_{j+1}$

refining layers. In summary, node representations are calculated by the formula (14):

$$P_j = M_{l-j} \cdot Q_j + Q_{l-j}$$

E. DISCRIMINABLE CAPSULE LAYER

Geoffrey Hinton is one of the founders of deep learning and the inventor of classic neural network algorithms such as back propagation. In order to solve CNN’s shortcomings, Hinton in [25] raised a more effective network for image processing Capsule Network, which combined the advantages of CNN and considered the relative position, angle and other information missing from CNN, so as to improve the effect of recognition.

A capsule is a group of neurons whose activities vector represents the instantiation parameters of a specific type of entity such as an object or an object part. Compared with the conventional convolution network, the advantages of the capsule network are as follows: firstly, the output of the convolution neural network is scalar, while the capsule network is vector with direction; secondly, each layer of the convolution neural network needs to do the same convolution operation, so it needs quite a lot of network data to study, which is time-consuming, inefficient and expensive. Capsule network requires the model to learn the feature variables in the capsule to maximize the retention of those valuable information, so it can use less training data to infer the possible variables to achieve the anticipated effect of CNN; thirdly, convolution neural network can not well deal with ambiguity, because continuous pooling will lose many important feature information, so that they are not sensitive to small changes. Therefore, when the complex tasks such as semantic segmentation are completed, we need to build complex architecture to solve the problem of information loss, Capsule Network is different, since each capsule is carrying a large amount of information, such as target location, rotation, and the size of the thickness and tilt. The information and other details have been saved and translated to the upper capsule and can use simple consistent architecture to cope with different visual tasks.

Algorithm 2 Routing Algorithm

1. $b^0_l = 0, b^0_2 = 0, \ldots, b^0_n = 0$
2. $T = n$
3. for $r = 1$ to $T$ do
   4. $c^r_1, c^r_2, \ldots, c^r_n = \text{softmax}(b^{r-1}_1, b^{r-1}_2, \ldots, b^{r-1}_n)$
      softmax computes formula (18)
   5. $u^r = v^r \times W^r, i \in [1, n]$
   6. $s^r = c^r_1u^1 + c^r_2u^2 + \ldots + c^r_nu^n$
   7. $a^r = \text{Squash}(s^r)$ squash computes formula (16)
   8. $b^{r+1}_i = b^{r+1}_i + a^r \cdot u^r, i \in [1, n]$
   9. $V = a^n$
10. return $a^1, a^2, \ldots, a^{n-1}, V$

We propose a discriminable capsule network, the network is adept in judging the similarities between features and is used to remember the rich information in the graph structure. In the layer $l − 2$, we use a SEGCN on $P_{l-2}$ to output (as in formula (15)):

$$P_{l-2} = \text{ReLU}(D^{l-2}_j \hat{A}^{l-3}_{l-3} \hat{A}^{l-2}_{l-2} P_{l-3}W_{l-2})$$

$P_{l-2} \in R^{n \times o}$, where $o$ represents the number of classes, $n$ is the number of nodes. We feed $n$ node feature vectors $v^1, v^2, \ldots, v^n$ into the discriminable capsule network as $n$ capsules, as shown in Fig. 4. Taking $1$ to $n$ ($n$ is the number of nodes) rows of feature matrix $P_{l-2}$ ($v^1, v^2, \ldots, v^n$) and input them into the network as $n$ capsules. They are multiplied by $n$ weights $W^1, W^2, \ldots, W^n$, respectively. We get $u^1, u^2, \ldots, u^n$. Through dynamic routing procedure, we can get output $a^1, a^2, \ldots, a^{n-1}, a^n$. These new capsules (feature vectors) form the matrix $P_{l-1} \in R^{n \times o}$ ($o$ is the number of classes). The dynamic routing procedure is summarized in Algorithm 2. We want the length of the output vector of the capsule to represent the probability that the entity represented by the capsule appears in the current input. Therefore, we use a nonlinear squashing function (see formula (16) to ensure that the short vector is compressed to almost zero length, and the long vector is compressed to a length slightly less than 1.

$$a^r = \frac{||s^r||^2}{1 + ||s^r||^2}, s^r = \frac{s^r}{||s^r||}$$

where $a^r$ (see formula (17)) is the vector output of capsule $v^r$ and $s^r$ is its total input. For all but the first layer of capsules, the total input to a capsule $s^r$ is a weighted sum over all prediction vectors $u^r$ from the capsules in the layer below and is produced by multiplying the input $v^r$ of a capsule in the
layer below by a weight matrix $W^i$.

$$s^i = \sum_{i=1}^{n} c_i^r u^r \quad r \in [1, n] u^r = W^i v^r$$  \hspace{1cm} (17)

where the $c_i^r$ (as in formula (18)) are coupling coefficients that are determined by the iterative. Sum of coupling coefficient of capsule $v^r$ and all capsules in the above layer is 1, which is determined by routing softmax, and its initial logits $b_i^r$ are the log prior probabilities that capsule $v^r$ should be coupled to capsule $a_i^r$.

$$c_i^r = \frac{\exp(b_i^r)}{\sum_{r=1}^{n} \exp(b_i^r)}$$  \hspace{1cm} (18)

The log priors can learn differently from other ownership values at the same time. The initial coupling coefficient are added to the initial logit from the discriminative capsule layer to get the matrix $\bar{v}$ are calculated.

$$v_t = \sum_{i} v_i$$

The agreement is simply the scalar product $r_i \cdot v_t$. The protocol is treated as a log likelihood and the $\exp$ is the add operation as in formula (19):

$$P_{l-1} = P_{l-2} \oplus P_{l-1}$$  \hspace{1cm} (19)

\section{THE OUTPUT LAYER}

Finally, in the output layer $l$, we use a softmax classifier on $P_{l-1}$ to output probabilities of nodes (formula (20)):

$$P_l = \text{softmax}(ReLU(D_{l-1}^\hat{i} \cdot A_{l-1}^\hat{i} D_{l-1}^\hat{i} - \frac{1}{2} P_{l-1} W_l))$$  \hspace{1cm} (20)

where $W_l \in R^{l \times o}$ is a trainable weight matrix and $P_l \in R^{l \times o}$ denotes the probabilities of nodes pertain to each class.

The loss function is defined as the cross-entropy of predictions over the labeled nodes (see formula (21)):

$$L = - \sum_{i=1}^{m} \sum_{y=1}^{o} \Pi(p_i = y_i) \log(P_{i, y_i})$$  \hspace{1cm} (21)

where $\Pi(\cdot)$ is the indicator function, $y_i$ is the true label for $v_i$, $h_i$ is the prediction for labeled node $v_i$, and $P(p_i, y_i)$ is the predicted probability that $v_i$ is of class $y_i$, $o$ represents the number of classes.

\section{EXPERIMENTS AND ANALYSIS}

\subsection{DATA SETS}

To verify the effectiveness and benefit of the proposed D-SEGCN on semi-supervised learning tasks, we test it on three citation networks (Cora, Citeseer and Pubmed). Citation network is a network composed of papers and their relationships, including citation relationships, co-authors, etc. It has a natural graph structure, and the tasks for data sets is the general classification of papers and the prediction of connections. The details of these data sets and their usages in our experiments are introduced below. The statistics of these data sets are summarised in Table 1.

**Cora.** This data contains 2708 nodes and 5429 edges. Each node has a 1433 dimension feature descriptor and all the nodes are falling into seven classes.

**Citeseer.** This network data contains 3327 nodes and 4732 edges. The nodes are falling into six classes. Each node is divided into a 3703 dimension feature descriptor.

**Pubmed.** This data contains 19717 nodes and 44338 edges. Each node has a 500 dimension feature descriptor and all the nodes are divided into three classes.

\subsection{EXPERIMENTAL SETTING}

For Cora, Citeseer and Pubmed datasets, we follow the experimental setup of previous works by Kipf and Welling [5], and Velićkoivić et al. [11]. During training, only 20 labels per class are employed to each citation network. Besides, 500 nodes in each data set are selected randomly as the validation set.

\subsection{PARAMETER SETTINGS}

We train D-SEGCN model for a maximum of 1500 epochs (training iterations) using an ADAM algorithm [31] with learning rate 0.03. The dropout is applied to all features vectors with rates of 0.9. All the network weights $W$ are initialized using Glorot initialization [32]. Considering the different scales of the data sets, we set the total number of layers $l$ to eighteen for citation networks.

\subsection{COMPARISON WITH STATE-OF-THE-ART METHODS}

**Baseline Methods.** In order to evaluate the performance of D-SEGCN, we compared our method with the following representative methods. The specific classification accuracy is shown in Table 2.

**ManiReg** Belkin et al. [26] proposed a learning algorithm based on a new regularization form that can take advantage of the geometric properties of edge distributions. It focused on a semi-supervised framework that combines labeled and unlabeled data in general learners.

**DeepWalk** Perozzi et al. [27] generated node embeddings through unsupervised random walks, and then input the embedding vector into the SVM classifier for node classification.

\begin{table}[h]
\centering
\caption{Statistics of datasets used in our experiments.}
\begin{tabular}{|c|c|c|c|}
\hline
Task & Cora & Citeseer & Pubmed \\
\hline
# Nodes & 2708 & 3527 & 19717 \\
# Edges & 5429 & 4732 & 44338 \\
# Features/Node & 1433 & 3703 & 500 \\
# Classes & 7 & 6 & 3 \\
# Training Nodes & 140 & 120 & 60 \\
# Validation Nodes & 500 & 500 & 500 \\
# Test Nodes & 1000 & 1000 & 1000 \\
Labeling rate & 0.052 & 0.036 & 0.003 \\
\hline
\end{tabular}
\end{table}
FIGURE 4. The workflow of the discriminable Capsule layer. Taking 1 to n (n is the number of nodes) rows of feature matrix $P_{l-2}$ ($v^1, v^2, \ldots, v^n$) and input them into the network as n capsules.

TABLE 2. Comparison results of node classification on dataset Cora, Citeseer and Pubmed in terms of accuracy.

| Method      | Cora   | Citeseer | Pubmed |
|-------------|--------|----------|--------|
| ManiReg     | 59.5%  | 60.1%    | 70.7%  |
| DeepWalk    | 67.2%  | 43.2%    | 65.3%  |
| LP          | 68.0%  | 45.3%    | 63.0%  |
| Planetoid   | 75.7%  | 64.7%    | 77.2%  |
| Chebyshev   | 81.2%  | 69.8%    | 74.4%  |
| GCN         | 81.5%  | 70.3%    | 79%    |
| DGCN        | 83.3%  | 72.6%    | 79.3%  |
| GAT         | 83.0± 0.7% | 72.5± 0.7% | 79.0± 0.3% |
| H-GCN       | 84.5 ±0.5% | 72.8± 0.5% | 79.8± 0.4% |
| D-SEGCN     | 84.8%  | 74.0%    | 81.2%  |

LP. Zhu et al. [28] proposed a traditional graph based on a Gaussian random field model to semi-supervised learning methods.

Planetoid Yang et al. [29] proposed a semi-supervised learning framework based on graph embedding. The framework gave a graph between instances and trains the embedding of each instance to jointly predict the class labels and neighbor contexts in the graph.

Chebyshev Defferrard et al. [4] proposed a kind of convolutional network which extended traditional CNN to Non-Euclidean space.

GCN Kipf and Welling et al. [5] produced node embedding vectors by truncating the Chebyshev polynomial to the first-order neighborhoods.

DGCN Zhuang and Ma [10] used the adjacency matrix and positive mutual information matrix of graph to encode both local consistency and global consistency.

GAT Veličković et al. [11] generated a node embedding vector by modeling the difference between the node and its hop neighbor.

H-GCN Hu et al. [20] proposed a novel deep hierarchical graph convolutional network for semi-supervised node classification in order to increase the receptive field.

Results. Table 1 summarizes the basic situation of three citation network benchmark datasets (Cora, Citeseer and Pubmed [30]). Table 2 summarizes the comparison results of the three datasets. The best results are marked as bold in Table 2. Overall, we can note that D-SEGCN performs better than other graph based semi-supervised learning methods, such as ManiReg, Deepwalk and LP, which further demonstrates the effectiveness of D-SEGCN in the task of executive graph semi-supervised classification.

E. ABLATION STUDY

To verify the effectiveness of the proposed SEGCN module and discriminable Capsule layer to alleviate over-smoothing problems, the ablation study is conducted in the section. The results are shown in Table 3 and Table 4.

We take measures to verify that the proposed D-SEGCN can use global information to selectively enhance the significant features and suppress the useless features. Thereby strengthening the features of nodes. At first, the squeeze and excitation operation is not fused to GCN to get the classification accuracy of the three benchmark graph datasets (Cora, Citeseer, and Pubmed) are: 84.5%, 72.4%, and 79.8%, respectively (see Table 3). By incorporating squeeze and excitation operation to the first layer of GCN, the classification accuracy on the three benchmark graph datasets (Cora, Citeseer, and Pubmed) are: 84.2 ± 0.4%, 72.4%, and 80.2%, respectively. Then squeeze and excitation operation are fused to each GCN layer to obtain the classification accuracy are: 84.6%, 73.1%, 80.6%, respectively.
Another operation used to prove the discriminable capsule layer is adept in judging the similarities among features, and can be used to remember the rich information in the graph structure and enhance the discriminability of nodes. Therefore, the over-smoothing problem is alleviated. The addition of discriminable Capsule layer allows newly acquired features to fuse with previous features. As can be seen from Table 4, the classification accuracy of the model for the three data sets Cora, Citeseer, and Pubmed are increased: 0.2%, 0.9% and 0.6%, respectively.

V. CONCLUSION

In this paper, we have introduced a novel graph convolutional networks for semi-supervised classification on graph-structured data. The D-SEGCN model is composed of SEGCN layers, coarsening layers, symmetrical refining layers, a discriminable capsule layer and a softmax classifier layer. Compared to the previous work, the proposed model can selectively enhance significant features and suppress useless ones by adding squeeze and excitation modules. While the proposed model is also to realize the adaptive calibration of feature dimension, enhance the discriminability of nodes. The final features are fused with the features obtained from the discriminable capsule layer, which can well judge the similarities among features and can remember the rich information in the graph structure. Therefore, the over-smoothing can be effectively alleviated by aggregating the multi-level neighborhood. Comprehensive experiments have confirmed that the proposed method consistently outperformed other state-of-the-art methods. In particular, the proposed approach in this paper achieves more significant results when the labeled data is extremely scarce.

REFERENCES

[1] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in Proc. IEEE Int. Joint Conf. Neural Netw., Jul/Aug. 2005, pp. 7729–7734.
[2] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” IEEE Trans. Neural Netw., vol. 20, no. 1, pp. 61–80, Jan. 2009.
[3] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” Tech. Rep., 2013.
[4] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3844–3852.
[5] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Represent., 2017, pp. 1–13.
[6] R. Li, W. Sheng, F. Zhu, and J. Huang, “Adaptive graph convolutional neural networks,” in Proc. AAAI Conf. Artif. Intell., 2018, pp. 3546–3553.
[7] M. Henaff, J. Bruna, and Y. LeCun, “Deep convolutional networks on graph-structured data,” 2015, arXiv:1506.05163. [Online]. Available: http://arxiv.org/abs/1506.05163.
[8] A. Micheli, “Neural network for graphs: A contextual constructive approach,” IEEE Trans. Neural Netw., vol. 20, no. 3, pp. 498–511, Mar. 2009.
[9] Q. Li, Z. Han, and X. M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in Proc. 32nd AAAI Conf. Artif. Intell. (AAAI), Menlo Park, CA, USA: Association Advance Artificial Intelligence, 2018, pp. 3538–3545.
[10] C. Zhuang and Q. Ma, “Dual graph convolutional networks for graph-based semi-supervised classification,” Tech. Rep., 2018.
[11] P. V. kovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in Proc. Int. Conf. Learn. Represent., 2017, pp. 1–12.
[12] K. K. Thekumparampil, C. Wang, S. Oh, and L. J. Li, “Attention-based graph neural network for semi-supervised learning,” in Proc. Int. Conf. Learn. Represent., 2018, p. 17.
[13] J. Atwood and D. Towsley, “Diffusion-convolutional neural networks,” in Proc. Conf. Workshop Neural Inf. Process. Syst., 2016, pp. 1993–2001.
[14] L. W. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1024–1034.
[15] X. Peng, H. Zhu, J. Feng, C. Shen, H. Zhang, and J. T. Zhou, “Deep clustering with sample-assignment invariance prior,” IEEE Trans. Neural Netw. Learn. Syst., early access, Dec. 31, 2019, doi: 10.1109/TNNLS.2019.2958324.
[16] X. Peng, J. Feng, S. Xiao, W.-Y. Yau, J. T. Zhou, and S. Yang, “Structured AutoEncoders for subspace clustering,” IEEE Trans. Image Process., vol. 27, no. 10, pp. 5076–5086, Oct. 2018.
[17] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, “HARP: Hierarchical representation learning for networks,” in Proc. 32nd AAAI Conf. Artif. Intell., Menlo Park, CA, USA: Association Advance Artificial Intelligence, 2017, pp. 1–9.
[18] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 5453–5462.
[19] R. Ying, J. You, C. Morris, X. Ren, L. W. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling.”
[20] F. Hu, Y. Zhu, S. Wu, L. Wang, and T. Tan, “Hierarchical graph convolutional networks for semi-supervised node classification,” in Proc. 28th Int. Joint Conf. Artif. Intell., Aug. 2019, pp. 4532–4539.
[21] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze- and-excitation networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 8, pp. 2011–2023, Aug. 2020.
[22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–777.
[23] J. Liang, S. Gurukar, and S. Parthasarathy, “MILE: A multi-level framework for scalable graph embedding,” 2018, arXiv:1802.09612. [Online]. Available: http://arxiv.org/abs/1802.09612.
[24] G. Karypis and V. Kumar, “Multilevel K-way partitioning scheme for irregular graphs,” Tech. Rep., 2005.
[25] S. Sabour, N. Frost, and G. E. Hinton, “Dynamic routing between capsules,” Tech. Rep., 2017.
[26] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples,” J. Mach. Learn. Res., vol. 7, pp. 2399–2434, Nov. 2006.
[27] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2014, pp. 701–710.
[28] X. Zhu, Z. Ghahramani, and J. D. Lafferty, “Semi-supervised learning using Gaussian fields and harmonic functions,” in Proc. 20th Int. Conf. Mach. Learn. (ICML), 2003, pp. 912–919.
[29] Z. Yang, W. Cohen, and R. Salakhudinov, “Revisiting semi-supervised learning with graph embeddings,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 40–48.
[30] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and A. T. Eliassi-Rad, “Collective classification in network data articles,” AI Mag., vol. 29, no. 3, pp. 93–106, 2008.
[31] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Represent., 2015, pp. 1–15.
[32] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” J. Mach. Learn. Res., vol. 9, pp. 249–256, May 2010.
N. Jia et al. received the B.S. degree in mathematics and applied mathematics from the School of Mathematics and Computer Science, Modern College of Humanities and Sciences, Shanxi Normal University, Shanxi, China, in 2013, and the M.S. degree in applied mathematics from the School of Mathematics and Statistics, Qinghai Normal University, Xining, China, in 2018. She is currently pursuing the Ph.D. degree in electronic science and technology from the School of Artificial Intelligence, Xidian University, Xian, China. Her current research interests include computer vision, image processing, with a focus on graph neural network and graph data mining.

Xiaolin Tian (Member, IEEE) received the Ph.D. degree in intelligent information processing from the School of Artificial Intelligence, Xidian University, Xian, China, in December 2008. He is currently a Professor and a Doctoral Supervisor with the School of Artificial Intelligence, Xidian University. He was a Visiting Scholar with the Vision Lab, Computer Science Department, UCLA. He is also a member of the Institute of Intelligent Information Processing and a member of the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Xidian University. He is a member of the innovation team of Yangtze River scholars and innovation group development plan of the Ministry of education, a member of the national 111 plan innovation and talent introduction base, and a member of China computer society. At present, he is mainly engaged in image understanding and interpretation, computer vision, intelligent algorithm system implementation, and other research work. He has presided over and participated in more than ten national projects, many provincial and ministerial projects, published more than 20 SCI retrieval articles in international mainstream journals, and obtained more than 30 national invention patents. In 2012, he won one Shaanxi Science and Technology Award, and one Shaanxi Teaching Achievement Award, in 2018.

Yang Zhang received the B.S. degree in intelligence science and technology from the School of Artificial Intelligence, Xidian University, Xian, China, in 2018, where he is currently pursuing the M.S. degree in circuit and system. His current research interests include machine learning, deep learning, and computer vision, with a focus on visual object tracking.

Fengge Wang received the B.S. degree in electronic information science and technology from the School of Science, Xi’an University of Architecture and Technology, Xian, China, in 2018. She is currently pursuing the M.S. degree in circuit and system from the School of Artificial Intelligence, Xidian University, Xian. Her current research interests include machine learning, deep learning, and computer vision, with a focus on image classification.