Topological graph convolutional network based on complex network characteristics

HE GAO\textsuperscript{1}, XIANG YU\textsuperscript{1}, YI SUI\textsuperscript{1}, FENGJING SHAO\textsuperscript{1} and RENCHENG SUN\textsuperscript{1}

\textsuperscript{1}College of Computer Science and Technology, Qingdao University, Qingdao, China

Corresponding author: Rencheng Sun (e-mail: src@qdu.edu.cn).

ABSTRACT Graph convolutional neural networks have received a lot of attention in various tasks dealing with graph data by aggregating information from neighboring nodes and passing node information. Many recent studies have looked at the impact of topological features on node classification tasks by altering aggregation based on degree values or incorporating topological data analysis into graph convolutional neural networks; however, graph data itself has many topological characteristics in complex networks. In most circumstances, the graph’s topological characteristics reveal the nodes’ similarity and facilitate the node classification task. This paper proposes a topological structure feature extraction method based on the concept of complex topological characteristics, which can obtain deeper topological features in the graph structure and use node features to obtain feature information that is more important for classification from both feature spaces. Evidence from experimental studies has established that the topological structure obtained by the method in this paper can be used as input to the GCN, and good results can be achieved on the classification task even without any external information on the nodes. In the graph dataset of connected topologies, the method exhibits a very large increase in accuracy and macro F1-score when compared to the state-of-the-art baseline model after mixing the node features.

INDEX TERMS Graph Convolutional Networks, Network Representation Learning, Topological Structure, Complex Networks

I. INTRODUCTION

Machine learning and deep learning have evolved rapidly, especially in the area of processing structured data (such as speech, image and natural language, etc.), and have made great strides. However, the vast majority of things in the real world are better represented by graph than by a sequence or a grid (such as social networks, knowledge graphs, complex file systems, etc.) [1]. In order to better handle such unstructured data, graph neural networks were created.

Graph neural networks have made significant progress in recent years. It has been used to perform various tasks on graph data, and existing studies have demonstrated its great potential in applications such as node classification [2] [3], graph classification [4], link prediction [5] [6] and recommendation [7] [8]. This paper is most concerned with graph convolutional neural networks, where feature information is passed through the network topology to nodes by aggregating neighbor nodes and message passing, and the learned node embeddings are used in classification tasks [9]. For graph structure data, the features of each node are important as graph structure data, as are the topological features of the entire network structure, and they are all key features in the node classification task. The graph data itself possesses many topological features. We start from the topological features of the graph data itself. How to make good use of the graph topological features in classification tasks is a problem worthy of continued research.

In this work, the goal of this paper is to make full use of the complex topological characteristics of the graph itself to provide more features for the node classification task. Essentially, the challenge of this paper is to explore the topological features of graph-structured data. There are many network topological characteristics of the graph data itself, such as Degree Centrality, Clustering Coefficient [10], Betweenness Centrality [11], K-core, etc. Based on the topology of the graph network itself, this paper proposes a graph structure topological feature learning method, which constructs the topological feature graph of the original graph by using the
topological characteristics of the network itself, and obtains the node embedding with the node feature graph, the original graph and its combination (T-GCN). Technically speaking, in order to make better use of obtaining topological features of graph structures, this paper obtains graph topological feature graphs by clustering coefficient topological characteristics. Meanwhile, this paper uses the k-nearest neighbor graph formed by the node features as the node feature graph in order to fully utilize the information in the feature space and learn the node features. The node features are propagated over the topological feature graph, the node feature graph, and the original graph, resulting in the extraction of three specific node embeddings in the three feature spaces via specialized convolution. This paper utilizes a convolution with the same parameters to jointly extract the common embeddings in the three feature spaces, and then uses an attention mechanism to learn the weights of the different embeddings and fuse them adaptively to extract the most useful information, taking into account the common relationship between the two feature graphs and the original graph.

II. RELATED WORK
Over the past few years, several researchers have increasingly begun to extend deep learning to graph structured data. One of the results is the graph neural network. The existing graph neural networks are mainly divided into two categories, one is the spectral domain-based graph convolution network [12] [13] , and the other is the spatial-based graph convolution network [3]. Spectral-domain graph convolution is based on spectral graph theory and convolution theorem, converting data from the spatial domain to the spectral domain for processing, which has a more solid theoretical foundation. Bruna et al. [14] proposed a graph convolution using a learnable diagonal matrix instead of a convolution kernel in the spectral domain. Since the eigenvalue decomposition of the Laplace matrix needs to be computed in the process, this is a very time-consuming operation and the parameter complexity of the model is large. For this reason, Michaël Defferrard et al. [15] used Chebyshev polynomials instead of convolution kernels in the spectral domain, simplifying the operation of convolution. Thomas N. Kipf et al. [3] proposed a graph convolutional network (GCN), which can be considered as a further simplification of ChebNet, considering only 1st order Chebyshev polynomials and only one parameter per convolution kernel. Although GCN is a simplified spectral domain approach, it can also be seen as a spatial domain approach. From the node’s point of view, GCN aggregates information from its neighbors when updating the node’s representation. There are also many recent approaches to design aggregation and delivery of messages based on spatial domains, Yotam Hechtlinger et al. [16] use a convolutional approach that fixes the number of neighbor nodes and orders the fixed nodes. William L. Hamilton et al. [12] used a convolutional approach of sampling and information aggregation; Petar Velickovic et al. [17] used an attention mechanism for differentiated aggregation of neighborhood nodes. The process of aggregation in the above conventional graph convolution approach is a topology learning process in the spectral and spatial domains, but he does not directly take into account the existing topological aspects of the graph data itself.

In recent studies, a number of researchers have taken into account the importance of topological features and applied them to models, and DEMO-Net [18] proposes a degree specific multi-task graph convolution function to learn node representations. AM-GCN [19] is considering node features and topology and adaptive fusion of both features, but AM-GCN is acquiring topological features directly from the original graph. Max Horn et al. [20] used persistent homology to integrate global topological information of graphs and compute topological features of structured datasets, and designed a TOGL layer that can be easily integrated into any type of GNN. Z-GCNets [21] use zigzag persistence as a topological feature and combine it in the time dimension to propose a new topological summary-a new topological summary, zigzag persistence image. Recent work has considered topological features in several ways, and in fact, several topological features already exist for graph data itself. Can deeper topological features be obtained if the topological features of the graph data itself are used and combined into the graph convolutional network? Based on this, we look at topological characteristics of complex networks and do some work on combining node features.

III. THE PROPOSED MODEL
In this section, the key issue is topological features and node features to be considered in the graph convolutional network, and there are many topological features in the graph network itself, such as clustering coefficient, Betweenness Centrality, etc. The topological feature extraction approach suggested in this research calculates the clustering coefficient of each node to generate the topological feature graph, utilizes the k-nearest neighbor graph as the node feature graph, and adaptively fuses all the features. The following first introduces key concepts and definitions.

A graph is defined as \( G = (A, X) \), where \( A \in \mathbb{R}^{n \times n} \) is the...
adjacency matrix, \( n \) denotes the node, and \( X \in \mathbb{R}^{n \times d} \) is the node feature matrix, \( d \) is the dimension of the node features. Specifically, \( A_{i,j} = 1 \) means there exists an edge between nodes \( i \) and \( j \). Conversely, \( A_{i,j} = 0 \). We consider each node as belonging to one of the classes \( C \).

Figure 1 depicts the framework diagram presented in this paper. The main aim is to extract topological feature information that is more relevant to the topology’s node labels and to make optimal use of the node characteristics. We independently design a graph structure topology feature extraction method that mines the original graph structure for features more relevant to the label. With two specific convolution modules, \( X \) is able to propagate over the topological feature graph and the node feature graph to learn two specific node embeddings, \( Z_T \) and \( Z_F \), respectively, while in order to retain the features on the original graph structure, feature learning is also performed using specific convolution modules to get the node embedding \( Z_P \). To consider common features in the topological feature graph, node feature graph, and original graph, we use a common convolution module with shared weight parameters to jointly learn the common embeddings of nodes \( Z_{CT}, Z_{CF} \) and \( Z_{CP} \). Furthermore, considering the relevance of several features, we used an attention mechanism to adaptively fuse these node embedding learning weights to obtain the most relevant information \( Z \) for the final classification task.

A. TOPOLOGY GRAPH CONVOLUTION MODULE

First, in order to obtain the topological feature graph in the graph structure, we design a method to obtain the topological feature graph \( G_T = (A_T, X) \) from the original graph \( G_p = (A_p, X) \), where \( X \) is the node feature matrix and the adjacency matrix of the topological feature graph obtained by \( A_T \). Specifically, we use the clustering coefficient of each node in the graph to obtain topological features to design the topology capture function, and calculate the clustering coefficient of each node for the original graph structure

\[
C(i) = \frac{2|\{ e_{jk} : v_j, v_k \in L(i), e_{jk} \in E \}|}{k_i (k_i - 1)} \tag{1}
\]

where \( v_j, v_k \) denote the neighboring nodes of node \( i, e_{jk} \) denotes the connected edges of \( v_j \) and \( v_k \), and \( k_i \) denotes the number of edges connected to \( v_i \).

Multiply the clustering coefficient of the nodes with connected edges

\[
A_{ij} = C(i) \times C(j) \tag{2}
\]

The topological feature adjacency matrix \( A_T \) is obtained. If the clustering coefficient of the current node is 0, we use the clustering coefficient of the multi-order neighbors of the node with clustering coefficient of 0 as a substitute value. If the clustering coefficient of the current node’s multi-order neighbors is 0, we use the Z-score normalization of the current node degree to replace the case where the clustering coefficient is 0.

Then, we take \( G_t = (A_t, X) \) as the input feature space, and the output \( Z_t^{(l)} \) of the \( l \)th layer can be expressed as

\[
Z_t^{(l)} = \text{ReLU} \left( \tilde{D}_t^{-\frac{1}{2}} \tilde{A}_t \tilde{D}_t^{-\frac{1}{2}} Z_t^{(l-1)} W_t^{(l)} \right) \tag{3}
\]

where \( W_t^{(l)} \) is the weight matrix of the \( l \)th layer in GCN, ReLU is the ReLU activation function and initializes \( Z_t^{(0)} = X \), \( \tilde{A}_t = A_t + I_t \) is the topological feature graph adjacency matrix, \( \tilde{D}_t \) is the diagonal degree matrix of the adjacency matrix \( A_t \). In this way we can obtain the information \( Z_T \) of the topological feature space by convolution.

B. NODE FEATURE GRAPH CONVOLUTION MODULE

This module, we in order to construct the node feature graph in the graph structure and capture the hidden features in the node feature space, we first construct the similarity matrix \( S \in \mathbb{R}^{n \times n} \) from \( n \) nodes using cosine similarity, and then the node feature graph \( G_f = (A_f, X) \) is constructed by a predetermined number of k-nearest neighbors, where \( X \) is then the feature matrix of the nodes and \( A_f \) is the adjacency matrix in the KNN graph.

Cosine similarity can be measured by the cosine of the angle between two vectors

\[
S_{jk} = \frac{x_j \cdot x_k}{|x_j| |x_k|} \tag{4}
\]

where \( x_j \) and \( x_k \) are the feature vectors of node \( j \) and node \( k \).

Next, we take \( G_f = (A_f, X) \) as the input feature space, and the output \( Z_f^{(l)} \) of the \( l \)th layer can be expressed as

\[
Z_f^{(l)} = \text{ReLU} \left( \tilde{D}_f^{-\frac{1}{2}} \tilde{A}_f \tilde{D}_f^{-\frac{1}{2}} Z_f^{(l-1)} W_f^{(l)} \right) \tag{5}
\]

where \( \tilde{A}_f = A_f + I_f \) is the node feature graph adjacency matrix, initialized \( Z_f^{(0)} = X \), and the other parameters are the same as in Eq. 3. In this way we can obtain the information \( Z_F \) in the feature space of the node.

C. ORIGINAL GRAPH CONVOLUTION MODULE

This module, keeping the basic features in the original graph, takes as input the original graph \( G_p = (A_p, X) \), where \( A \) is the adjacency matrix of the original graph, and obtains the information \( Z_P \) in the feature space of the original graph by the above convolution form.

D. COMMON GRAPH CONVOLUTION MODULE

In this module, the features of the several feature graphs we obtained above are not completely unrelated. For the node classification task, he relies on topological features as well as node features, and their common features are equally important. Therefore, we adopt a parameter sharing approach to design a common GCN module that allows us to extract features that are common to all three graph structures.

First, we extract the node embedding \( Z_{ct} \) by GCN from the topological feature graph \( G_f = (A_f, X) \):

\[
Z_{ct}^{(l)} = \text{ReLU} \left( \tilde{D}_t^{-\frac{1}{2}} \tilde{A}_t \tilde{D}_t^{-\frac{1}{2}} Z_t^{(l-1)} W_c^{(l)} \right) \tag{6}
\]
where $W_c^{(l)}$ is the common weight matrix of the $l$th layer and $Z_c^{(0)} = X$.

For the node feature graph $G_f = (A_f, X)$, the node embedding $Z_c^{(l)}$ is also extracted by GCN:

$$Z_c^{(l)} = \text{ReLU} \left( D_f^{-\frac{1}{2}} A_f D_f^{-\frac{1}{2}} Z_c^{(l-1)} W_c^{(l)} \right)$$

where $W_c^{(l)}$ is the common weight matrix of the $l$th layer and $Z_c^{(0)} = X$.

With the same treatment, we extract the node embedding $Z_p^{(l)}$ from the original graph $G_p = (A_p, X)$ by GCN. By sharing the weights we can obtain the common features in the feature space of the three graph structures. According to different input graphs, we can get three output node embeddings $Z_{CT}$, $Z_{CF}$ and $Z_{CP}$. We obtain the final embedding $Z_C$ from the three node embeddings.

$$Z_C = \frac{Z_{CT} + Z_{CF} + Z_{CP}}{3}$$

### E. ATTENTION MECHANISM

In this module, we obtain four node embeddings $Z_T$, $Z_F$, $Z_P$ and $Z_C$. To make them a better integration, we use an attention mechanism that adaptively learns the corresponding weights. $\alpha_t, \alpha_f, \alpha_p, \alpha_c \in \mathbb{R}^{n \times 1}$ denotes the attention weights of $n$ nodes in $Z_T$, $Z_F$, $Z_P$, and $Z_C$, respectively.

Here we focus on node $i$, $Z_{CT} \in \mathbb{R}^{1 \times h}$ denotes the node embedding of node $i$ in $Z_T$, we use a nonlinear transformation and then use a shared attention $\alpha \in \mathbb{R}^{h \times 1}$ to get the attention value $w^i_T$ with the following equation.

$$w^i_T = a^T \cdot \tanh \left( Z^i_T \cdot W + b \right)$$

where $W \in \mathbb{R}^{h \times h}$ is the weight matrix and $b \in \mathbb{R}^{h \times 1}$ is the bias vector. Similarly, we can calculate $w^i_F, w^i_P, w^i_C$, who are the weights of $Z_F, Z_P$, and $Z_C$, respectively, according to the above formula. Finally we use the softmax function to normalize $w^i_T, w^i_F, w^i_P, w^i_C$ to get the final weights.

$$\alpha^i_T = \text{softmax}(\omega^i_T)$$

Similarly, $\alpha^i_F = \text{softmax}(\omega^i_F), \alpha^i_P = \text{softmax}(\omega^i_P), \alpha^i_C = \text{softmax}(\omega^i_C)$ can all be computed separately. For $n$ nodes, the embedding weights of each feature can then be expressed as $\alpha_T = \text{diag}(\alpha^1_T, \alpha^2_T, \cdots, \alpha^n_T), \alpha_F = \text{diag}(\alpha^1_F, \alpha^2_F, \cdots, \alpha^n_F), \alpha_P = \text{diag}(\alpha^1_P, \alpha^2_P, \cdots, \alpha^n_P), \alpha_C = \text{diag}(\alpha^1_C, \alpha^2_C, \cdots, \alpha^n_C)$. Finally, we fuse the four embeddings to obtain the final embedding $Z$.

$$Z = \alpha_T \cdot Z_T + \alpha_F \cdot Z_F + \alpha_P \cdot Z_P + \alpha_C \cdot Z_C$$

### F. OBJECTIVE FUNCTION

Our model achieves reasonably decent results without adding any additional loss functions. Our objective function is the cross-entropy error.

According to the final embedding $Z$ obtained by Eq. 11, we are able to obtain the prediction category $\hat{Y}$ for $n$ nodes.

$$\hat{Y} = \text{softmax}(W \cdot Z + b)$$

where softmax$(x)$ is a normalization of all classes.

If $L$ is said to be the training set, then for each $l \in L$ the true label is $Y_l$ and its predicted label is $\hat{Y}$. then the cross-entropy loss of all training nodes for node classification is denoted as $L_l$:

$$L_l = - \sum_{l \in L} \sum_{i=1}^{C} Y_{li} \ln \hat{Y}_{li}$$

### IV. EXPERIMENTS

#### A. EXPERIMENTAL SETUP

**Datasets:** T-GCN uses a representative dataset and three datasets with complex relationships for the experiments, where Table 1 summarizes the basic information of the dataset.

- **Citeseer** [3]: Citeseer is a citation network with nodes representing documents and edges representing citation relationships. Bag-of-words is used to obtain the feature vector of each document, and all nodes are divided into six classifications.
- **UA12010** [22]: With 3067 nodes and 28311 edges, we utilized this dataset. This dataset is used in [22] to detect communities.
- **ACM** [23]: The papers are separated into three categories (database, wireless communication, data mining), nodes signify papers, and if the authors of two papers are the same, the two papers are concatenated, and the feature vectors of nodes are represented by bag-of-words.
- **BlogCatalog** [24]: The BlogCatalog dataset is a social relationship network, the graph is composed of bloggers and his/her social relationships (e.g. friends), and the labels are the interests of the bloggers. There are 5196 nodes and 171743 edges, and all nodes are classified into 6 categories.
- **Flickr** [24]: This dataset is made up of linkages between publicly accessible Flickr photographs. Photographs from the same place, images uploaded to the same gallery, group, or collection, images with similar tags, images taken by friends, and so on create edges. All nodes are grouped into nine categories depending on user interests, with edges expressing the link between them.

**Baseline:** The state-of-the-art two-class method, which includes two word embedding methods and six graph neural network approaches, is compared to our methodology.

| Dataset    | Node | Edges | Classes | Features | Training | Test  |
|------------|------|-------|---------|----------|----------|-------|
| Citeseer   | 3327 | 4732  | 6       | 3703     | 120      | 1000  |
| UA12010    | 3067 | 28311 | 19      | 4973     | 380      | 1000  |
| ACM        | 3025 | 13128 | 3       | 1870     | 60       | 1000  |
| BlogCatalog| 5196 | 171743| 6       | 8189     | 120      | 1000  |
| Flickr     | 7575 | 239738| 9       | 12047    | 180      | 1000  |
TABLE 2. Node classification results (%). (Bold: best)

| Dataset      | Metrics | DeepWalk | LINE | Chebyshev | GCN | KNN-GCN | Cluster-GCN | GAT | DEMO-Net | AM-GCM | T-GCN |
|--------------|---------|----------|------|-----------|-----|---------|-------------|-----|----------|--------|-------|
| Citeseer     | ACC     | 43.47    | 32.21| 69.80     | 70.30| 61.35   | 64.60        | 72.50| 69.50    | 73.70 | 72.70 |
|              | F1      | 38.09    | 31.75| 65.92     | 67.50| 58.86   | 62.78        | 68.14| 67.84    | 68.42 | 69.30 |
| UAI2010      | ACC     | 42.02    | 43.47| 50.02     | 49.88| 66.06   | 57.90        | 56.92| 70.10    | 70.80 | 70.50 |
|              | F1      | 32.93    | 37.01| 33.65     | 32.86| 52.43   | 47.43        | 39.61| 16.82    | 55.61 | 57.08 |
| ACM          | ACC     | 62.69    | 41.28| 75.24     | 87.80| 78.52   | 86.30        | 84.48| 90.40    | 90.80 | 90.70 |
|              | F1      | 62.11    | 40.12| 74.86     | 87.82| 78.14   | 86.37        | 84.16| 90.43    | 90.23 | 90.60 |
| BlogCatalog  | ACC     | 38.67    | 58.75| 38.08     | 69.84| 75.49   | 85.70        | 64.08| 54.19    | 81.98 | 90.23 |
|              | F1      | 34.96    | 57.75| 33.39     | 68.73| 72.53   | 85.71        | 63.38| 52.79    | 81.36 | 77.10 |
| Flicker      | ACC     | 24.33    | 33.25| 23.26     | 41.42| 69.28   | 51.70        | 38.52| 34.89    | 75.26 | 77.29 |
|              | F1      | 21.33    | 31.19| 21.27     | 39.95| 70.33   | 49.68        | 37.00| 33.53    | 74.63 | 77.29 |

- **DeepWalk** [25]: a network embedding approach that obtains contextual information through random walks and learns the network representation using the Skip-gram algorithm.
- **LINE** [26]: a large-scale network embedding approach that retains the network’s first-order and second-order closeness, respectively. Here we use LINE (first-order + second-order).
- **Chebyshev** [15]: a graph neural network method using Chebyshev filter.
- **GCN** [3]: a semi-supervised graph convolutional network model that learns node representations from their neighbors.
- **KNN-GCN**: As a contrast, instead of using the original graph as the input graph for GCN, we utilize the sparse k-nearest neighbor graph generated from the original graph adjacency matrix and describe it as KNN-GCN.
- **Cluster-GCN**: Also as a comparison, we employ the Cluster-GCN, which uses the topological feature graph generated from the original graph adjacency matrix as the input to the GCN.
- **GAT** [17]: a graph neural network model that aggregates node characteristics via an attention mechanism.
- **DEMO-Net** [18]: a degree-specific graph neural network for node classification.
- **AM-GCN** [19]: Specific and common embeddings are extracted from node features, topologies and their combinations, and the attention mechanism is used to learn the adaptive importance weights of the embeddings for fusion to obtain the final node embeddings.

**Parameter settings**: To further analyze and demonstrate the model’s performance, we randomly choose 1000 points as the test set and 20 nodes from each class in the dataset as the training set. All baselines are initialized using the same values provided in their work. We need to train six 2-level GCNs with the same input hidden dimension (nhid1) and output hidden dimension (nhid2) at the same time for our model, where nhid1 ∈ {512, 768} and nhid2 ∈ {32, 128, 256}. We utilize the Adam optimizer with a learning rate of 0.0001 ~ 0.005, a dropout rate of 0.5, weight decay ∈ {5e−3, 5e−4}, and k ∈ {2...9} for the km graph. After ten runs with the identical partitions, we summed the results for all approaches, and we assessed the models using accuracy (ACC) and marco F1-Score (F1).

**B. NODE CLASSIFICATION**

The results on the node classification task are compared in Table 2, and the following are some of our observations:

- In comparison with the baseline method, T-GCN shows better performance in all datasets. As a result of T-GCN, we were able to improve accuracy by 8.71% and 1.84%, and F1 scores by 8.88% and 2.66% on two very complex datasets, BlogCatalog and Flickr. This demonstrates that T-GCN is effective.
- When comparing the performance of KNN-GCN with Cluster-GCN on the Citeseer, ACM, and BlogCatalog datasets, Cluster-GCN outperforms KNN-GCN on all three, and even outperforms AM-GCN on the BlogCatalog dataset. On the UAI2010, Flickr dataset, however, KNN-GCN performs better. This shows that both topological feature graphs and node feature graphs have unique features, and it is necessary to include topological feature graphs and node feature graphs in the traditional GCN. It also shows that the topological feature extraction method we designed is effective.
- In all datasets, T-GCN consistently beats KNN-GCN, Cluster-GCN, and GCN, indicating that it is capable of...
adaptive fusion and can extract more valuable information from all three feature spaces.

C. VISUALIZATION
We did the visualization task on the BlogCatalog dataset for a more visual comparison and to further demonstrate the effectiveness of our model, where we output the node embeddings in the test dataset and plotted the coloring of the BlogCatalog node classification by t-SNE [27] dimensionality reduction, as shown in Figure 2.

Figure 2 shows that GCN and GAT have poor classification, however AM-GCN can identify the classes more clearly, though the classification is not as excellent as T-GCN in certain areas. T-GCN outperforms the others in terms of visualization, as it learns node embeddings with a more compact structure, the highest intra-class similarity, and the most striking distinctions between various types.

D. THE RELATIONSHIP BETWEEN TOPOLOGY AND CLASSIFICATION
T-GCN performs better in terms of accuracy and F1 scores after the analysis in IV-B, however the improvement varies between datasets. In order to analyze the relationship between the methods we used and the classification task, we analyzed some basic topological characteristics of the experimental dataset as shown in Table 3.

We can see from Table 3 that for BlogCatalog and Flickr, which have large performance gains, they are fully connected graphs and are disassortative, while for Citeseer and ACM, which have smaller gains, they are non-connected graphs and are assortativity. In terms of clustering coefficient, the overall average clustering coefficient isn’t the determining factor in the effectiveness of T-GCN, but the number of points with a clustering coefficient of 0 has a greater impact on the method, implying that the stronger the data clustering, the better the effect. There are more connected subgraphs in Citeseer and ACM. To further analyze these two datasets, We drew the scale distribution of the connected subgraphs of both datasets Citeseer and ACM as shown in Figure 3.

Both datasets have the same one feature, both have a larger connected subgraph with many connected subgraphs of fewer nodes. The graph is more fractured as a whole, and the overall aggregation is low. The topological feature information collected using the approach described in this research is insufficient, resulting in a minor performance increase.

After studying numerous datasets, it’s clear that the strategy suggested in this research may significantly enhance performance on some well-connected and disassortative data, but it won’t help much with other basic datasets with low aggregation. This also demonstrates that the strategy suggested in this study is better suited to data sets with complicated topologies and high connectedness.

E. TOPOLOGICAL FEATURES STUDY
We utilized the clustering coefficient to get topological structural characteristics in this paper, but is the value of the clustering coefficient we used useful, and can we get decent performance by superimposing the original graph structure? We attempted to replace the topological feature graph with the original graph for the structure of Figure 3 based on this problem, and the experimental findings are displayed in Table 4 by comparing the overlaid original graph with our structure.

It is discovered that employing a topological feature graph outperforms merely overlaying the original graph, with the topological features obtained being deeper and the clustering coefficient values being valid. Topological characteristics cannot be represented simply by superimposing the structure of the original graph; instead, a more sophisticated method of acquisition is required.

### Table 3. Basic topological characteristics of the dataset.

| Dataset     | number of nodes | number of edges | Number of concatenated subgraphs | assortativity | average clustering | Number of clustering coefficients ≤ 0 |
|-------------|-----------------|-----------------|----------------------------------|---------------|--------------------|--------------------------------------|
| Citeseer    | 3327            | 4676            | 438                              | 0.1397        | 0.1414             | 2316                                 |
| UAI2010     | 3067            | 28314           | 8                                | -0.1100       | 0.3703             | 250                                  |
| ACM         | 2464            | 13128           | 156                              | 0.6589        | 0.6741             | 351                                  |
| BlogCatalog | 5196            | 171743          | 1                                | -0.0200       | 0.1223             | 1                                    |
| Flickr      | 7575            | 239738          | 1                                | -0.2275       | 0.3301             | 57                                   |

### Table 4. Topology graph and original graph Node classification results(%). (Bold: best)

| Dataset     | Metrics | Topology graph | Original graph |
|-------------|---------|----------------|----------------|
| Citeseer    | ACC     | 72.50          | 71.50          |
|             | F1      | 69.06          | 67.55          |
| UAI2010     | ACC     | 70.80          | 70.50          |
|             | F1      | 57.08          | 56.96          |
| ACM         | ACC     | 90.50          | 90.50          |
|             | F1      | 90.37          | 90.47          |
| BlogCatalog | ACC     | 88.50          | 84.40          |
|             | F1      | 87.97          | 83.96          |
| Flickr      | ACC     | 77.10          | 76.10          |
|             | F1      | 77.29          | 75.58          |
F. COMPUTATIONAL COMPLEXITY ANALYSIS

In this section, we analyze the computation of the clustering coefficient and the time consumption of the training process to analyze the advantages and disadvantages of the model in terms of computational complexity. The most commonly used method for computing clustering coefficients is intersecting adjacency lists, which has a time complexity upper bound of $O(V \cdot d_{\text{max}}^2)$, where $V$ denotes the number of nodes and $d_{\text{max}}$ is the vertex with the largest adjacency matrix. The core computational time complexity of our method is not high for the computation of clustering coefficients with approximately linear time complexity, which is the consideration of our attempt to use clustering coefficients to obtain topological features in terms of time complexity. For some global topology metrics, such as Betweenness Centrality, the computational time complexity is too high, thus discarding the use of such metrics.

We tried to calculate the time to build the topological feature graph and carved a graph of the time to build the topological feature graph versus the total time for the training dataset loss stabilization, as shown in Figure 4. The other times in the figure indicate the difference between the time when the training dataset loss is stable and the time when the topological feature graph is constructed. As can be seen from Figure 4, the process of constructing the topological feature graph is a relatively small percentage of time in the whole training and stabilization process. Of course, based on the analysis of the complexity of the clustering coefficients, the time consumption for constructing topological feature graphs is relatively large for larger-sized graphs, which is approximately linearly related to the number of edges in the dataset.

G. PARAMETER STUDY

In this section, we explore the sensitivity of the parameters on the Citeseer and BlogCatalog.

Analysis of the K-nearest neighbor construction node feature graph $k$: When constructing the node feature graph, in order to explore the influence of the top $k$ neighborhoods in KNN. In Figure 5, we investigate the performance of T-GCN at different K-number ranges from 2 to 9. For both Citeseer and BlogCatalog, the accuracy and F1 scores increase and then decrease, reaching higher values when $k = 9$. This could be because bigger values give the node feature graph more structural information.

Analyzing the effect of learning rate on the model: Different hyperparameter settings can have an impact on the performance of the model. In this paper, we explore the effect of not using learning rate on model performance without changing other hyperparameters. On Citeseer and BlogCatalog, the learning rates were set to 0.0001, 0.0003, 0.0005, 0.001, 0.005, 0.007, and 0.02, respectively. The experimental results are shown in Figure 6. As we can see in Figure 6, for the Citeseer dataset, the best model performance is achieved for a learning rate of 0.0005, when the values of both the accuracy and the F1 score are higher than the other learning rates. For the BlogCatalog dataset, a learning rate of 0.005 is the best model performance for a learning rate of 0.005 when the values of both the accuracy and the F1 score are higher than the other learning rates.

V. CONCLUSION

In this paper, we rethink the use of topological structural characteristics in graph convolutional neural networks. We examine how to use the topological structure information in the network data itself and create an extraction method for obtaining topological features using clustering coefficients based on this fundamental idea. And combined with the features of the graph nodes themselves are applied to each other in the graph convolution process, making it possible to make fuller use of topological and node features in the graph convolution. Our method showed outstanding performance in
some datasets with good network topology connection after employing the topological features produced by this method. For non-connected datasets, our approach has corresponding limitations, with only a small improvement in some metrics. To address this issue, we will continue to consider combining multiple topological characteristics metrics to obtain topological features later. Extensive trials have proven the approach’s superiority, with the method outperforming state-of-the-art models on real datasets.

REFERENCES
[1] J. Gao and L. Hao, “Graph neural network and its applications,” in Journal of Physics: Conference Series, vol. 194, no. 1. IOP Publishing, 2021, p. 012004.
[2] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galtstyan, “Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing,” in international conference on machine learning. PMLR, 2019, pp. 21–29.
[3] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” arXiv preprint arXiv:1609.02907, 2016.
[4] H. Gao and S. Ji, “Graph u-nets,” in international conference on machine learning. PMLR, 2019, pp. 2083–2092.
[5] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” arXiv preprint arXiv:1611.07308, 2016.
[6] M. Cogun and M. Koyuturk, “Node similarity-based graph convolution for link prediction in biological networks,” Bioinformatics, vol. 37, no. 23, pp. 4501–4508, 2021.
[7] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, “Graph neural networks for social recommendation,” in The World Wide Web Conference, 2019, pp. 417–426.
[8] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems,” in Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 974–983.
[9] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in International conference on machine learning. PMLR, 2017, pp. 1263–1272.
[10] D. J. Watts and S. H. Strogatz, “Collective dynamics of ’small-world’ networks,” nature, vol. 393, no. 6684, pp. 440–442, 1998.
[11] L. C. Freeman, “A set of measures of centrality based on betweenness,” Sociometry, pp. 35–41, 1977.
[12] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025–1035.
[13] M. Ding, J. Tang, and J. Zhang, “Semi-supervised learning on graphs with generative adversarial nets,” in Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 913–922.
[14] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” arXiv preprint arXiv:1312.6203, 2013.
[15] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” Advances in neural information processing systems, vol. 29, pp. 3844–3852, 2016.
[16] Y. Hechtlinger, P. Chakravarti, and J. Qin, “A generalization of convolutional neural networks to graph-structured data,” arXiv preprint arXiv:1704.08163, 2017.
[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” arXiv preprint arXiv:1710.10903, 2017.
[18] J. Wu, J. He, and J. Xu, “Net: Degree-specific graph neural networks for node and graph classification,” in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 406–415.
[19] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, “Am-gcn: Adaptive multi-channel graph convolutional networks,” in Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining, 2020, pp. 1243–1253.
[20] M. Horn, E. De Brouwer, M. Moor, Y. Moreau, B. Rieck, and K. Borgwardt, “Topological graph neural networks,” arXiv preprint arXiv:2102.07835, 2021.
[21] Y. Chen, I. Segovia, and Y. R. Gel, “Z-genets: Time zigzags at graph convolutional networks for time series forecasting,” in International Conference on Machine Learning. PMLR, 2021, pp. 1684–1694.
[22] W. Wang, X. Liu, P. Jiao, X. Chen, and D. Jin, “A unified weakly supervised framework for community detection and semantic matching,” in Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 2018, pp. 218–230.
[23] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, “Heterogeneous graph attention network,” in The World Wide Web Conference, 2019, pp. 2022–2032.
[24] Z. Meng, S. Liang, H. Bao, and X. Zhang, “Co-embedding attributed networks,” in Proceedings of the twelfth ACM international conference on web search and data mining, 2019, pp. 393–401.
[25] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.
[26] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in Proceedings of the 24th international conference on world wide web, 2015, pp. 1067–1077.
[27] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.” Journal of machine learning research, vol. 9, no. 11, 2008.

HE GAO received the B.Eng degree in Computer Science and Technology from Qilu University of Technology (Shandong Academy of Sciences) in Jinan, China, in 2019. He is currently pursuing the M.S. degree in School of Computer Science and Technology at Qingdao University. His current research interests are deep learning and graph neural networks.

XIANG YU received the M.S. degree in computer technology from the Qingdao University, Qingdao, China, in 2020. He is currently pursuing the Ph.D. degree with the Software Engineering, Qingdao University of China. His current research interests include big data analysis, deep learning and remote sensing.

YI SUl is currently an associate professor at College of Computer Science Technology, Qingdao University, China. She received the Ph.D. degree in system theory from Qingdao University in 2012. From 2018 to 2019, she worked in Center for Spatial Information Science, The University of Tokyo as a visiting researcher. In the past five years, she led and participated in many important projects as principal investigator or primary actor, such as Young Scientists Fund of the National Natural Science Foundation of China, National Statistical Science Research Project of China, Major Research Plan of the National Natural Science Foundation of China, and The National Special Research Fund for Non-Profit Sector. Her research interests are big data, artificial intelligence and complex network.
FENGJING SHAO was born in Qingdao, China in 1955. She received Bachelor degree of Computer Science in Shandong University in 1982. In 1988 she had Master’s in Business Administration in Shiga University. Prof. She had Doctorate in Technology at Osaka University in 1991. She was employed as a researcher in central research center of Panasonic Corporation between 1991-1994. Since 2017, she was nominated as fellow of Developing World Academy of Engineering and Technology (AETDEW). She is the author of 4 books, more than 150 articles, and more than 10 inventions. Her interest areas in research included protocol automatic verification, data warehouse and data mining, machine learning, complex network, and etc.

RENCHEENG SUN received the M.S. degree in Computer software and theory from the Qingdao University, Qingdao, China, in 2003, and the PH.D. degree in system theory from the Qingdao University, Qingdao, China, in 2010. Since 2003, he has been engaged in teaching and research in Qingdao University. His research interests include Machine Learning, Deep Learning and Complex Networks.

Mr. Sun awards and honors include the First Prize for Science technology Development/Achievement of the Ministry of Education, Second prize of scientific and technological progress of Shandong Province, etc.