electronics

MDPI

Article

A Novel Two-Channel Classification Approach Using Graph Attention Network with K-Nearest Neighbor

Yang Wang [1], Lifeng Yin [1,*], Xiaolong Wang [2], Guanghai Zheng [1] and Wu Deng [3,4]

[1] School of Rail Intelligent Engineering, Dalian Jiaotong University, Dalian 116028, China
[2] Oneg Robot Yinchuan Co., Ltd., Yinchuan 750021, China
[3] School of Electronic Information and Automation, Civil Aviation University of China, Tianjin 300300, China
[4] State Key Laboratory of Rail Transit Vehicle System, Southwest Jiaotong University, Chengdu 610031, China
* Correspondence: yinlifeng1030@djtu.edu.cn or yinlifeng1030@163.com

Abstract: Graph neural networks (GNNs) typically exhibit superior performance in shallow architectures. However, as the network depth increases, issues such as overfitting and oversmoothing of hidden vector representations arise, significantly diminishing model performance. To address these challenges, this paper proposes a Two-Channel Classification Algorithm Based on Graph Attention Network (TCC_GAT). Initially, nodes exhibiting similar interaction behaviors are identified through cosine similarity, thereby enhancing the foundational graph structure. Subsequently, an attention mechanism is employed to adaptively integrate neighborhood information within the enhanced graph structure, with a multi-head attention mechanism applied to mitigate overfitting. Furthermore, the K-nearest neighbors algorithm is adopted to reconstruct the basic graph structure, facilitating the learning of structural information and neighborhood features that are challenging to capture on interaction graphs. This approach addresses the difficulties associated with learning high-order neighborhood information. Finally, the embedding representations of identical nodes across different graph structures are fused to optimize model classification performance, significantly enhancing node embedding representations and effectively alleviating the over-smoothing issue. Semi-supervised experiments and ablation studies conducted on the Cora, Citeseer, and Pubmed datasets reveal an accuracy improvement ranging from 1.4% to 4.5% compared to existing node classification algorithms. The experimental outcomes demonstrate that the proposed TCC_GAT achieves superior classification results in node classification tasks.

Keywords: graph neural network; graph attention network; node classification; overfitting; oversmoothing; two-channel classification algorithm

Citation: Wang, Y.; Yin, L.; Wang, X.; Zheng, G.; Deng, W. A Novel Two-Channel Classification Approach Using Graph Attention Network with K-Nearest Neighbor. *Electronics* **2024**, *13*, 3985. https://doi.org/10.3390/electronics13203985

Academic Editor: Stefanos Kollias

Received: 9 September 2024
Revised: 29 September 2024
Accepted: 7 October 2024
Published: 10 October 2024

1. Introduction

As the digital era progresses, the volume of data generated daily is growing exponentially. Broadly, these data can be categorized into two types: Euclidean and non-Euclidean. Euclidean data, with its regular structure, is easy to quantify and can be processed and analyzed through conventional mathematical methods. This type of data typically includes images [1,2], videos [3], audio [4], and text [5–7], which can be directly or indirectly mapped onto two-dimensional or three-dimensional spaces. In contrast, non-Euclidean data, prevalent in citation networks [8], social networks [9], and recommendation systems [10–12], requires more complex and specialized methods for processing and analysis due to its intricate structure and network of relationships [13–17].

In the realm of Euclidean data, Convolutional Neural Networks (CNNs) [18] have undoubtedly shown exceptional performance, particularly achieving great success in image classification [19–24] and object detection [25–29]. However, CNNs face significant limitations when applied to non-Euclidean data [30,31], with the primary hurdle being their inability to adapt to the irregular structure of graph data. To address this, the introduction

of Graph Neural Network (GNN) architectures [32] has offered new perspectives and effective strategies for processing non-Euclidean data. GNNs have been widely applied in tasks such as node classification [33–37], graph classification [38–41], and link prediction [42–46], demonstrating formidable capabilities and broad applicability in managing complex structural data.

Graph Convolutional Networks (GCNs) [47] and their derivative models leverage graph structural information for aggregation, integrating features and labels of target nodes with those of their neighbors. This approach updates the embedding representations of target nodes, facilitating integrated learning of both graph structures and node features. This unique convolution operation has enabled GCNs to exhibit superior performance in node classification tasks involving irregular data.

However, GCNs face several limitations in practical applications. During the training phase, the model requires the entire graph structure (the whole graph) as input. This results in significant memory consumption when processing large graphs, consequently leading to increased usage costs as the size of the graph grows. Further, issues such as poor generalization have arisen. To overcome these challenges, Hamilton et al. [48] developed the GraphSAGE model, leveraging local neighboring information for message passing and thereby learning node embeddings. This method not only facilitates the processing of large-scale graph data but also adapts to newly appearing nodes, enabling inductive learning.

While GraphSAGE offers a new pathway to process large-scale graph data, it encounters issues such as prolonged training periods in practical applications. Moreover, its Laplacian aggregation operation cannot adaptively integrate neighborhood information, limiting model performance. The study of AM GCN [49] underscores the vital role of adaptively integrating neighborhood information in model performance, emphasizing that GCN models struggle to adaptively discern the intricate correlations between graph structures and node features, resulting in a performance that falls significantly short of the ideal.

Addressing these issues, the introduction of attention mechanisms has become particularly important, allowing GNNs to better focus on task-relevant edges and nodes, thereby improving training effectiveness and model accuracy. Hong et al. [50] utilized graph networks to explore the complex interplay between linguistic instructions and visual cues, introducing a two-level soft-attention mechanism. This approach conducted a meticulous analysis of the specific context of navigation instructions and their interrelationships, leading to significant advancements in semantic relationship modeling and algorithm performance optimization. In contrast, Graph Attention Networks (GATs) [51] prioritize enhancing the model's generalizability. By embedding attention mechanisms into GCNs, they achieve a dynamic amalgamation of neighboring node information, thereby elevating algorithm effectiveness.

However, GAT models typically rely on shallow depths and primary neighborhood information, failing to fully exploit higher-order features. When attempts are made to deepen the model in order to capture higher-order neighborhood information, it becomes susceptible to the over-smoothing phenomenon. This leads to increasingly ambiguous distinctions between nodes as the number of network layers grows, making it difficult to differentiate node embedding representations. This issue indicates that GNNs cannot straightforwardly increase their network depth in the same way as CNNs due to the risk of over-smoothing.

To address this challenge, Deep GCN [52] has constructed a high-depth graph neural network featuring a GNN architecture with 64 layers, thereby expanding the potential for increasing graph network depth and achieving significant performance improvements across various application tasks. This study introduced a generic form of aggregation function and compared different designs of residual connections within the GNN framework, aiming to elucidate the correlation between depth (higher-order neighborhood information) and performance. However, as the model depth increases, the number of parameters required for training also grows, leading to a substantial rise in GPU memory resource de-

mands. Consequently, despite Deep GCN making strides in enhancing GNN performance, efficiency remains a critical factor in evaluating algorithmic performance. In addition, several other methods or models have also recently been proposed [53–58].

Synthesizing the analysis above, the capability of GNNs for node classification heavily relies on the features of neighborhood nodes and the structural information of the graph. Models utilize the support of the graph's structure to aggregate the feature information of neighborhood nodes, thus updating their own node embedding representations to enhance node classification performance. However, the main constraints on their classification performance at this stage are also determined by two factors:

1. Acquisition of higher-order neighborhood information. GNNs can learn higher-order domain information by increasing network depth [59], thereby enhancing the discriminability of node embedding representations and achieving improved classification performance. However, the over-smoothing issue caused by deepening the network layers still restricts the performance of the model.
2. Acquisition of graph structural information. During the process of aggregating neighborhood information, GNNs also learn about the structural information of the graph. Relying solely on the interactions between nodes often fails to uncover the intrinsic connections among them, impeding the full exploration of their interrelations. Therefore, a profound understanding of graph structural information is crucial for enhancing the effectiveness of node classification.

To address these issues, this paper proposes a Two-Channel Classification Algorithm based on the Graph Attention Network (TCC_GAT) to improve the performance of GNN in node classification tasks. In order to enhance the model's ability to aggregate higher-order neighborhood information, a graph reconstruction module based on cosine similarity and the KNN algorithm is designed. This approach offers two types of graph structures for aggregating information in GNN: (1) an enhanced basic graph structure that facilitates and simplifies the capture of higher-order neighborhood information while mining node interaction behaviors, and (2) a behavior-based graph constructed via KNN, which provides a new perspective for mining structural information. Moreover, a parallel GAT network structure is utilized to construct the overall classification framework, integrating deeper node information to improve classification accuracy. Under the same dataset partition ratio, the classification performance of the model has been significantly enhanced, proving the effectiveness and correctness of the TCC_GAT.

The main contributions of this article are as follows:

1. A Two-Channel Classification Algorithm based on the Graph Attention Network is proposed.
2. A method for effectively acquiring higher-order neighborhood information is designed, enabling the acquisition of higher-order information at the same depth.
3. A graph reconstruction module is designed. Through K-nearest neighbors algorithms and cosine similarity, the graph structure is improved based on the similarity of interaction information, thereby diversifying the aggregation basis of graph neural networks and capturing rich graph structures and deep correlations between nodes to enhance algorithm performance.

The remainder of the paper is structured as follows: Section 2 briefly introduces related work; Section 3 presents the proposed innovative methodology, detailing the overall model framework, different functional modules, and specific model details; Section 4 explains the proposed algorithm through pseudocode and analyzes its complexity; Section 5 provides a comparative validation on multiple datasets, followed by a discussion of the results; the final section provides a brief conclusion and future research directions.

2. Related Work

2.1. Representation of Graph

This paper primarily focuses on undirected graphs, characterized by bidirectional edges. To begin, the representation and related concepts [60] are elucidated.

- Graph (G): Consists of a finite, non-empty set of vertices and a set of edges between these vertices, generally denoted as $G = (V, E)$, where G represents a graph, V is the set of vertices in graph G, and E is the set of edges in graph G. The vertex set $V = \{v_1, v_2, v_3, \cdots, v_n\}$ and the edge set $E = \{e_1, e_2, e_3, \cdots, e_m\}$, n and m represent the number of vertices and edges, respectively. If the edge between vertices v_i and v_j is undirected, it is represented as (v_i, v_j) or (v_j, v_i)
- Adjacency Matrix (A): Constructs the adjacency matrix A to represent the neighboring relationships between vertices in the graph. For undirected graphs, the adjacency matrix is symmetric. If there is a link between vertices v_i and v_j, then a_{ij} and a_{ji} are 1; otherwise, they are 0.
- Adjacency Matrix with Self-loops $(\widetilde{A})$: Refers to the adjacency matrix A to which an identity matrix I is added, denoted as $\widetilde{A} = A + I$. Herein, a self-loop implies that the starting and ending points are the same vertex. Within the adjacency matrix of an undirected graph, such self-loops are located at diagonal positions, generally indicated as $\widetilde{a_{ii}} = 1$.
- Degree Matrix (D): The degree of a vertex is defined by the sum of the elements in its corresponding row within the adjacency matrix. Specifically, the degree of vertex v_i represents the number of edges connected to that vertex, which also equals the number of neighbors of v_i. In the degree matrix D, the element on the diagonal $d_{ii} = \sum_j a_{ij}$ represents the degree of vertex v_i, while all other elements in the matrix are set to 0.
- Similarity Matrix (S): The similarity matrix S quantifies the similarity between different vertices based on their features. The similarity value s_{ij} between vertices v_i and v_j is calculated using the cosine similarity Formula (1), where n is the dimension of vertex features.

$$s_{ij} = \frac{v_i \cdot v_j}{\|v_i\| \|v_j\|} = \frac{\sum_{k=1}^{n} \left(v_{ik} \times v_{jk}\right)}{\sqrt{\sum_{k=1}^{n} (v_{ik})^2} \times \sqrt{\sum_{k=1}^{n} \left(v_{jk}\right)^2}} \tag{1}$$

- Adjacency Matrix of Similar Behavior (A_s): Constructed utilizing the K-nearest neighbors algorithm based on cosine similarity. This graph structure captures the similarity in interaction behaviors between vertices and their neighbors. The calculation process is illustrated in Figure 1, is represented through the row vectors of the adjacency matrix with self-loops $(\widetilde{A})$, where $\widetilde{a_i}$ denotes the i-th row of matrix $\widetilde{A}$, indicating the feature attributes of the node v_i. Following Equation (1), the similarity matrix S is computed by traversing all nodes, and subsequently, the adjacency matrix of similar behavior A_s is constructed using the K-nearest neighbors algorithm based on the similarity matrix S.

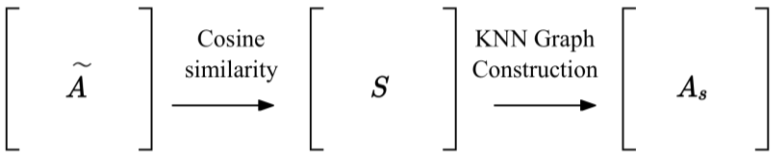


Figure 1. Process of constructing adjacency matrix of similar behavior.

2.2. Graph Attention Network

The introduction of the Self-Attention mechanism within the Transformer framework [61], via its Encoder–Decoder architecture, has led to marked improvements in

performance, significantly contributing to the widespread adoption and importance of attention mechanisms across diverse domains. Deeply exploring the Transformer architecture, Phan et al. [62] proposed a Structural Attention mechanism specifically for medical images. This mechanism allows for a more precise capture of the correlations among features, thereby significantly enhancing the overall performance of the model. In parallel, Chongjian Ge et al. [63] introduced a novel Neural Attention Learning Method (NEAL), which has significantly boosted the performance of models in the realm of object detection, further affirming the effectiveness of attention mechanisms.

These models are primarily oriented towards data in Euclidean spaces (such as images), relying on fixed positional relationships between data elements to calculate correlations or attention distributions, thus achieving high efficiency in processing image or textual data. However, the core characteristic of graph-structured data lies in the dynamic and non-Euclidean nature of its elements (nodes) and connections (edges), implying that the relationships between nodes possess significant flexibility and diversity. Therefore, applying traditional attention mechanisms to graph data encounters numerous challenges.

To address these challenges within graph data, the Graph Attention Network (GAT) integrates attention mechanisms with GCN, utilizing the attention mechanism to aggregate information from neighborhoods. This approach allows the model to adaptively allocate weights based on the significance of neighborhood information for a specific task, thereby optimizing parameters and ultimately obtaining embedded representations of the nodes.

In the GAT model, the collection of node features, $h = \{h_1, h_2, h_3, \cdots, h_N\}$ (with each $h_i \in \mathbb{R}^F$ representing the features of an individual node, N denoting the number of nodes, and F indicating the dimensionality of the node features), serves as the input information. After processing through the graph attention layer, the model produces a new set of feature vectors, $h' = \{h'_1, h'_2, h'_3, \cdots, h'_N\}$, where each $h'_i \in \mathbb{R}^{F'}$ (potentially altering to a different feature quantity F'), which then acts as input for subsequent modules.

The primary task of the graph attention layer is to utilize the weight matrix $W \in \mathbb{R}^{F \times F'}$ to perform linear transformations on the features of both the central node and its neighboring nodes, implementing element-wise feature processing [64]. GAT employs a shared attention mechanism that concatenates the linearly transformed features of the central node and its neighbors and multiplies them with the attention vector $a \in \mathbb{R}^{2F'}$. This mechanism evaluates the significance of neighboring nodes relative to the central node, generating an attention score e_{ij} among the nodes. The computation of these attention scores is illustrated in Equation (2) as:

$$e_{ij} = \text{LeakyReLU}\left(a^T \left[Wh_i || Wh_j\right]\right) \; j \in N_i \tag{2}$$

Herein, the '||' symbol stands for the operation of vector concatenation. The LeakyReLU function, acting as the activation function, provides a nonzero slope for all negative inputs, enhancing the modeling of nonlinear relationships.

Normalization of e_{ij} is conducted using the Softmax function, thus computing the attention weights α_{ij} that each node assigns to its neighbors. This ensures that the sum of attention a central node directs towards all its neighboring nodes equates to 1. This is expressed in Equation (3):

$$\alpha_{ij} = \text{Softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \tag{3}$$

The comprehensive calculation of attention weight is presented in Equation (4):

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(a^T \left[Wh_i || Wh_j\right]\right)\right)}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(a^T [Wh_i || Wh_k]))} \tag{4}$$

By this method, the model computes the weight coefficients α_{ij}, which are then utilized to update each node's embedding representation through a weighted sum operation. Utilizing the ELU(Exponential Linear Unit) activation function, the final embedding representation for each node is acquired:

$$h_i' = \mathrm{ELU}\left(\sum_{j \in N_i} \alpha_{ij} W h_j\right) \tag{5}$$

Equation (5) elucidates the calculation method for single-head attention. To boost the model's generalizability and ensure learning stability, this study incorporates a multi-head attention mechanism, as illustrated in Equation (6). This mechanism, by concurrently executing K independent single-head attentions and concatenating their outcomes, augments the model's processing capability and robustness:

$$h_i' = ||_{k=1}^{K} \mathrm{ELU}\left(\sum_{j \in N_i} \alpha_{ij}^k W^k h_j\right) \tag{6}$$

In Equation (6), K stands for the count of attention heads, with this study employing 8 heads. The α_{ij}^k represents the attention weight coefficient on the kth head, and W^k is the linear transformation matrix corresponding to the kth head. Following multi-head attention processing, this study opts for concatenation to generate h_i' as the final node feature representation, thus the ultimate node output h_i' embodies KF' node features.

These processes are illustrated in Figure 2, where h_2 to h_5 serve as neighbor nodes to h_1. The diagram utilizes distinct arrow shapes to represent the distinct attention computation processes.

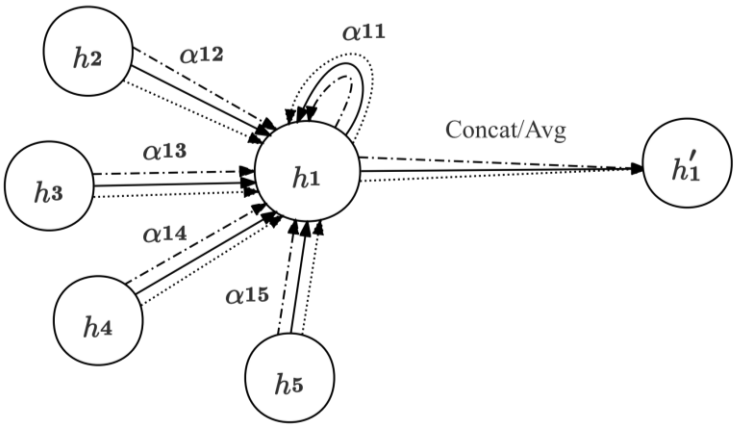


Figure 2. Multi-head attention.

3. TCC_GAT Model

3.1. The Overall Network Architecture

This article introduces a Two-Channel Classification Algorithm Based on Graph Attention Network (TCC_GAT), designed to optimize the performance of GNN in node classification tasks. The algorithm mainly acquires high-order neighborhood information through two approaches, utilizing parallel graph attention networks to effectively capture neighborhood structure and feature information, thereby significantly enhancing the embedding representation of target nodes. This method effectively overcomes the shortcomings of traditional GNN models in aggregating high-order neighborhood information and capturing structural information. The overall framework of the model consists of three modules: the graph reconstruction module, the graph data feature mining module, and the final result prediction module, with specific details shown in Figure 3.

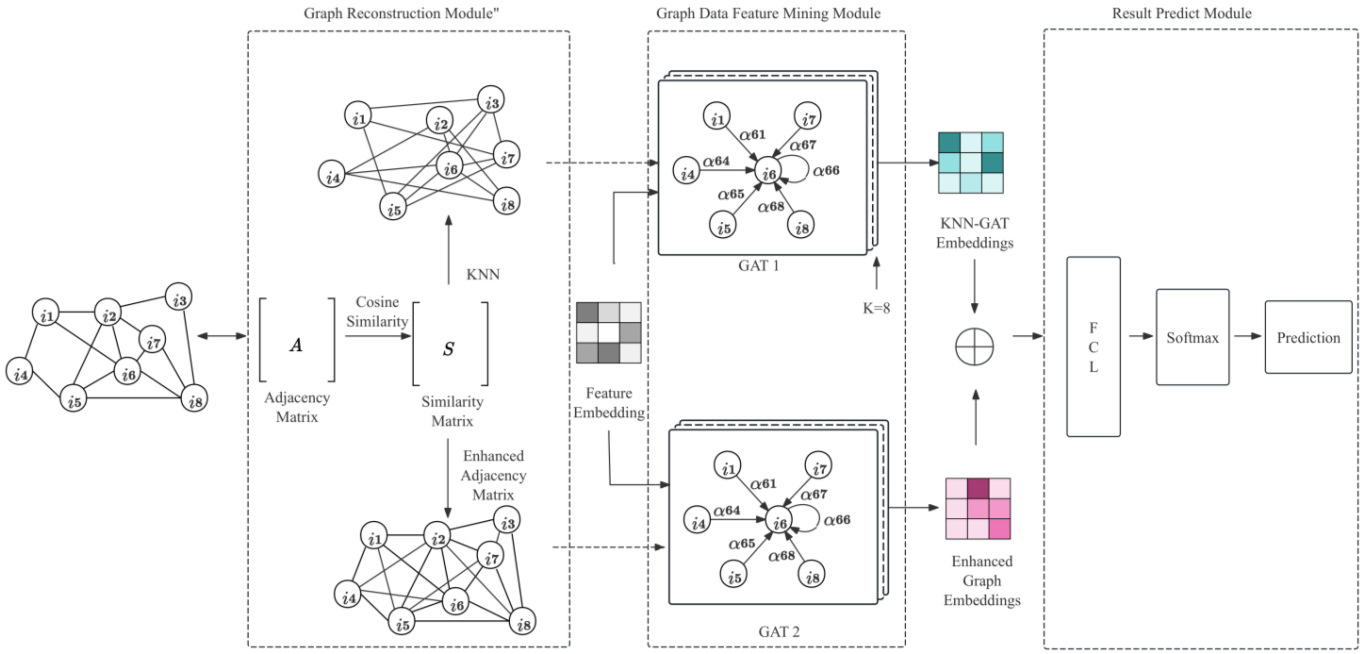


Figure 3. Overall framework diagram of the TCC_GAT.

Firstly, the graph reconstruction module exploits cosine similarity to mine the similarity of interactive behaviors between nodes, identifying strongly related nodes to reinforce the basic graph structure while increasing its capacity to aggregate high-order neighborhood information. Subsequently, relying on the similarity matrix, this module utilizes the K-nearest neighbors algorithm to restructure the graph, thereby generating a graph structure that reflects their interaction behavior similarity, enhancing the model's ability to aggregate structural information and high-order neighborhood information. Next, the outputs from the graph construction module, along with the node feature information, are inputted into the graph data feature mining module. This module updates the node embedding representations based on the newly constructed graph structure as the basis for neighborhood aggregation, using different GAT models. The embedding information of the same node under different graph structures is integrated, serving as the basis for model prediction, with node categories predicted through a fully connected layer and Softmax. Finally, the model adjusts network parameters based on gradient information calculated by the loss function, thereby improving the model's classification performance.

3.2. Graph Reconstruction Module

In the study of recommendation algorithms, collaborative filtering techniques have been extensively applied [64]. Consequently, this paper incorporates the concept of collaborative filtering, designs a graph reconstruction module, and integrates it into the GNN framework to enhance the basic graph structure and, thus, improve the model's performance in node classification tasks.

Figure 4 displays node i_2 and its interactive behavior (i.e., local information within the graph structure). The GAT allocates attention weight coefficients to the first-order neighborhood exclusively, thereby accomplishing the task of neighborhood information aggregation. However, when conducting a commonality analysis of nodes i_2 and others such as i_1, i_3, i_6, i_7, i_8, it is observed that nodes i_2 and i_8 exhibit a higher similarity in interactive behaviors, suggesting a higher likelihood of them belonging to the same category. In the process of feature fusion, the importance of i_8 relative to i_2 should be considered greater than that of other nodes.

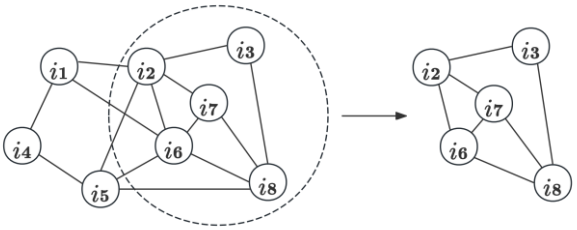


Figure 4. Local interaction diagram.

However, for a GAT to capture high-order neighborhood information similar to that of node i_8, it necessitates an increase in the depth of the network model. That is, multiple convolution operations are required to achieve the capability of aggregating high-order neighborhood information. Yet, with the decay of feature information during network propagation, even the acquisition of high-order features exerts a relatively limited impact on the target node. An excessive number of network layers leads to embedding features learned by different nodes becoming overly alike, making it difficult to distinguish effectively, thus causing the oversmoothing phenomenon and consequently restricting performance improvement. Therefore, this paper proposes the introduction of the collaborative filtering concept, through reconstructing the graph structure, to enhance the probability of target nodes directly aggregating neighbors with similar traits. The objective is to effectively capture the information of nodes in deep neighborhoods without increasing the number of network layers, thereby mitigating the issue of oversmoothing and enhancing the model's ability to classify.

The graph reconstruction module utilizes the interactive behaviors between nodes as the basis for constructing the graph. However, the adjacency matrix generated by these interactive behaviors often exhibits highly sparse characteristics, which are not conducive to subsequent similarity calculations. To mitigate this issue, this paper adopts an undirected graph approach, forming a diagonal matrix to improve the sparsity problem to a certain extent. For the problem of isolated nodes, to ensure the effectiveness of the computation, an identity matrix is added to the adjacency matrix, creating an adjacency matrix with self-loops, $\widetilde{A}$ which acts as the feature matrix for the nodes. $\widetilde{a_i}$ represents the ith row of matrix $\widetilde{A}$, indicating the feature information of node i. Using cosine similarity, calculate the differences in interaction behaviors between different nodes, thereby laying the groundwork for subsequent graph construction.

This paper chooses cosine similarity to calculate the differences in interactive behaviors between different nodes. Due to the characteristics of the data itself, when measuring differences using distance metrics, more emphasis is placed on relative differences. The cosine angle effectively avoids the variances in the individual perception of similarities, focusing more on the differences between dimensions rather than absolute numerical differences. For example, when analyzing the viewing behaviors of two dramas by users A and B, with user A's viewing vector being (0,1) and user B's (1,0), their cosine similarity is significant while the Euclidean distance is minimal. Analyzing the two users' preferences for different videos, it is clear that cosine similarity, which focuses more on relative differences, should be used.

By continuously traversing all nodes between the current node and all nodes through Formula (1), the similarity matrix S is obtained, where s_{ij} represents the degree of similarity between nodes i and j. The higher the value, the more similar the interactive behaviors between the two nodes are. After calculating the similarity matrix S, nodes with the highest similarity are added to the basic graph structure, increasing the number of neighbors for the target node and generating an enhanced graph (represented by the enhanced graph adjacency matrix A_e). This facilitates the model's ability to integrate high-order neighborhood information during the convolution process, thereby improving the overall performance of the model.

Figure 5 shows the enhanced graph structure, with red lines representing the edges added compared to the basic graph structure. This method extends the model's connectivity on the first-order neighborhood, facilitating subsequent feature aggregation.

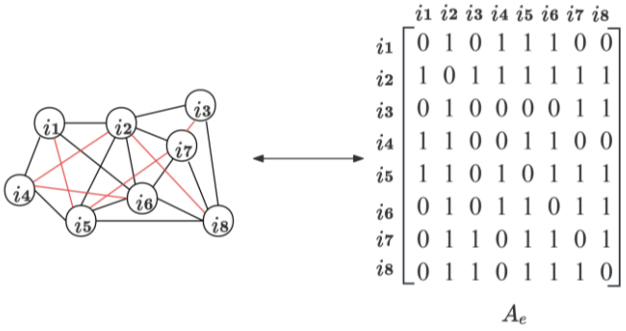


Figure 5. Enhanced graph.

However, this approach serves merely as an expansion of the basic graph structure, and its impact is relatively limited; the structural information that the model can capture in the feature aggregation phase is also quite singular. Therefore, this paper reconstructs the basic graph structure through the KNN algorithm. By utilizing the similarity in interactive behaviors between different nodes, the top K most similar nodes are selected as neighbors for the target node, thereby constructing a new graph structure (a similar behavior graph) that reflects the similarity in interactive behaviors. This serves as the basis for downstream model aggregation, represented by the similar behavior adjacency matrix A_s. This method allows the model to capture more comprehensive structural and neighborhood information, thereby improving the effectiveness of the embedding representation. Figure 6 shows the new graph structure constructed using the KNN algorithm.

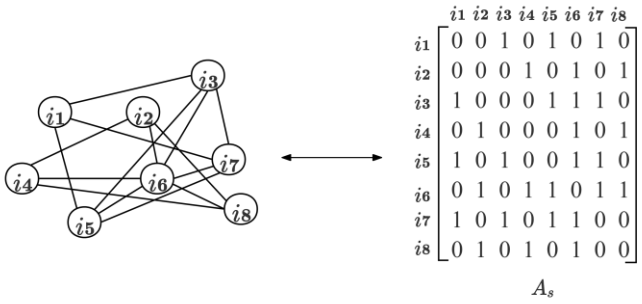


Figure 6. Similar behavior graph.

Ultimately, the graph reconstruction module outputs two types of graph structures (A_e and A_s), which are used for learning the embedding representations of identical nodes, facilitating the processing of subsequent tasks.

3.3. Graph Data Feature Mining Module

This paper employs the GAT as the foundational model for feature fusion. By integrating self-attention layers, GAT effectively overcomes the limitations of traditional GCN and their related approximations. It utilizes an attention mechanism to allocate unique weights to different nodes within neighborhoods. This strategy significantly enhances the model's sensitivity to newly added neighbor nodes, thereby facilitating a deeper understanding and learning of neighborhood information.

In the implementation phase, the similar behavior adjacency matrix A_s (which achieves self-loops by adding an identity matrix to A_s along with the node features h, are input into the GAT1. The calculation is performed according to Equation (7), producing node

embedding representations Z_s that exhibit similar interactive behaviors, serving as the output from GAT1:

$$Z_s^l = ||_{k=1}^K \mathrm{ELU}\left(\sum_{j \in N_i(A_s)} \alpha_{ij}^k W^k Z_s^{l-1}\right) \tag{7}$$

In this process, feature extraction is realized through the use of linear transformation matrices W^k, facilitating the calculation of attention coefficients α_{ij}^k, which serve as the attention weights between node i and its neighbor j in the Similar behavior graph. By leveraging these weight metrics α_{ij}^k and employing the ELU as the activation function, a non-linear transformation is applied to the learned embedding representations, where K denotes the number of attention heads. The adoption of the multi-head attention mechanism not only stabilizes the learning process but also effectively mitigates the risk of overfitting. By aggregating the feature information from K attention heads, an enhanced node embedding representation Z_s^l on the graph of similar behaviors is constructed, thereby enhancing both the model's expressive power and its classification accuracy.

Subsequently, the enhanced graph adjacency matrix A_e, generated through cosine similarity, along with the base features h, are fed into the second GAT2. As indicated by Equation (8), although α_{ij}^k and W^k represent the attention parameters and feature extraction matrices within GAT2, respectively, and maintain the same form as those within GAT1, the two models operate independently and do not share parameters.

$$Z_e^l = ||_{k=1}^K \mathrm{ELU}\left(\sum\nolimits_{j \in N_i(A_e)} \alpha_{ij}^k W^k Z_e^{l-1}\right) \tag{8}$$

Ultimately, features of the same node Z_s and Z_e are captured independently by the two GAT networks from different graph structures. These features are then fused by setting the hyperparameter β to amalgamate node information from the disparate networks, culminating in the final node embedding representation Z. This representation is subsequently fed into the classification prediction module for multi-class prediction, as delineated in Equation (9):

$$Z = (\beta \times Z_s) + ((1 - \beta) \times Z_e) \tag{9}$$

The two-channel learning strategy implemented in this paper comprehensively captures the feature information and details of the two graph structures, reflecting them fully in the nodes' embedding representations, thereby significantly improving the model's classification performance.

3.4. Classification Result Prediction Module

Building upon the aforementioned framework, this study opts for cross-entropy as the loss function, suitable for addressing semi-supervised classification problems. The feature extraction matrix W is applied to the input node embedding representations for feature selection, and Softmax is utilized as the activation function to obtain the model's final output $\hat{y}$, as depicted in Equation (10):

$$\hat{y} = \mathrm{Softmax}(W \cdot Z + b) \tag{10}$$

The cross-entropy loss function is employed to measure the difference between predicted labels and true labels, calculating the classification loss function L_{class}:

$$L_{class} = -\sum\nolimits_{l \in L} \sum\nolimits_{c=1}^{M} y_{lc} \ln(\hat{y}_{lc}) \tag{11}$$

Here, M represents the number of categories, and y_{lm} denotes the true labels describing the real categories of the current nodes, directing predictions for all $l \in L$ under the guidance of the labeled training set L. Predicted outcomes are represented using $\hat{y}_{lm}$. Through backpropagation, network parameters are optimized and updated with the goal of minimizing the loss function, enabling the model to effectively learn and recognize

the significance differences between nodes, thereby significantly enhancing the overall performance and classification accuracy of the algorithm.

4. TCC_GAT Model

4.1. Pseudocode for TCC_GAT

In this paper, a Two-Channel Classification Algorithm Based on Graph Attention Network (TCC_GAT) is introduced. This algorithm merges the concept of collaborative filtering, constructs two distinct graph structures, and employs parallel graph attention networks to effectively aggregate neighborhood information from each structure, thereby significantly enhancing performance. The following is the pseudocode for the graph reconstruction module, which is one of the key components of the TCC_GAT Algorithm 1.

Algorithm 1: The pseudo code of the TCC_GAT

Input:
Number of nearest neighbors in KNN : K
Graph adjacency matrix : A
Number of nodes in the graph : N

Output:
Similarity behavior graph adjacency matrix : A_s
Enhanced graph adjacency matrix : A_e

Initialize A_s
$\widetilde{A} = A + I$;
for $i = 0$ to N $-$ 1 do
  for $j = 0$ to N $-$ 1 do
    $s_{ij} = \frac{\widetilde{a_i} \cdot \widetilde{a_j}}{\left\|\widetilde{a_i}\right\| \cdot \left\|\widetilde{a_j}\right\|} = \frac{\sum_{k=1}^{n}\left(\widetilde{a_{ik}} \times \widetilde{a_{jk}}\right)}{\sqrt{\sum_{k=1}^{n}\left(\widetilde{a_{ik}}\right)^2} \times \sqrt{\sum_{k=1}^{n}\left(\widetilde{a_{jk}}\right)^2}}$ //Cosine similarity
    $s_{ii} = 0$; //Zeroing self-similarity
  end for
end for
for $i = 0$ to N $-$ 1 do //Iterate over all nodes
  $dist$ = zip ($S[i]$, range (N));
//Storing similarities and indices of node i with all other nodes in a list
  $dist$ = sorted ($dist, key = lamda, x : -x[0]$);
//Sorting pairs in descending order by similarity
  for $m = 0$ to K $-$ 1 do
    $neighbourID = dist[m][1]$;
//Finding the indices of the top K most similar neighbors
  end for
  for each $j \in neighbourID$ do
    $A_s[i][j] = 1.0$;
    $A_s[i][j] = A_s[j][i]$; // Create adjacency matrixA_s
  end for
  for each $j \in neighbourID[0]$ do
    $A[i][j] = 1.0$;
    $A[i][j] = A[j][i]$; //Enhanced adjacency matrix A
  end for
end for
$A_e = A$; // Create adjacency matrixA_e
return A_e, A_s

Based on the pseudocode presented above, it is demonstrated how the graph reconstruction module within the TCC_GAT algorithm utilizes cosine similarity and the KNN method to improve the graph structures. Subsequently, by aggregating neighborhood information from different graph structures through parallel GAT models, the algorithm can generate embedding representations of the same node across various graph structures.

By integrating these diverse embeddings, a significant enhancement in node classification performance is ultimately achieved.

4.2. Analysis of Complexity for TCC_GAT

The time complexity of the TCC_GAT model is $O(|V|FF'+|E|F') \times K$, where F represents the initial feature dimensionality of the nodes, F' is the new feature dimensionality after passing through a graph attention layer, and K is the number of heads in the multi-head attention mechanism.

In the TCC_GAT model, the features h_i of the target node are initially passed through a feature extraction matrix W for feature transformation, with a time complexity of $O(FF\prime)$. For all nodes in the graph, the complexity is $O(|V|FF')$, where $|V|$ is the number of nodes in the graph. When calculating attention coefficients, the model utilizes a shared attention mechanism a, concatenating the vectors of any two nodes and mapping them to real space, which incurs a time complexity of $O(F\prime)$. Since this calculation is performed for each edge, the complexity for this part is $O(|E|F')$. The aggregation process, being a weighted summation operation, does not involve complex computations. By employing a graph reconstruction strategy, the model is capable of converging in fewer epochs. Overall, the proposed model demonstrates a clear advantage in complexity.

5. Experimental Results and Analysis

5.1. Dataset

To validate the performance of the algorithm proposed in this paper, experiments were conducted on three commonly used citation datasets for node classification: Cora, Citeseer, and Pubmed. Nodes in these datasets represent academic papers, while the citation relationships between these papers constitute the edge information of the datasets. The specific dataset statistics are presented in Table 1 below.

Table 1. Citation data set.

Dataset	Node	Edge	Feature	Label
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
Pubmed	19,717	44,338	500	3

The Cora dataset primarily comprises academic papers from the machine learning field and has been widely used in the deep learning domain in recent years. This dataset contains a total of 2708 paper samples, categorized into seven classes: Case-Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, and Theory. Each paper is represented by a 1433-dimensional word vector. Each dimension corresponds to a word from a dictionary of 1433 words; the value is 1 if the word appears in the paper, otherwise, it is 0.

The Citeseer dataset encompasses papers from six major academic fields: Agents, Artificial Intelligence, Database, Machine Learning, and Human-Computer Interaction. It contains a total of 3327 papers, which form the dataset's relationship network through mutual citations. After removing stop words and filtering out words that appear less than ten times in the corpus, 3703 unique words were extracted.

The Pubmed dataset is derived from the Pubmed database and consists of 19,717 scientific papers in the field of diabetes, classified into three categories: Diabetes Mellitus, Experimental; Diabetes Mellitus Type 1; Diabetes Mellitus Type 2. The papers in this dataset are linked through 44,338 citation relationships. Each paper is represented by a TF/IDF weighted word vector, extracted from a dictionary comprising 500 unique vocabulary terms.

5.2. Experimental Setup

The experimental environment for the model includes both hardware configuration and software versions. On the hardware side, the GPU model used is the GeForce RTX 2080 Ti, and the CPU is an Intel(R) Xeon(R) Gold 6130. The software environment relies on the Windows 10 operating system, with Python version 3.8.3, and PyTorch version 1.7.1. All experiments were conducted on the Jupyter Notebook platform.

Regarding dataset splitting, this paper follows the method outlined in the GAT paper. The proposed model was tested across various datasets and compared with other models for analysis. To ensure fairness in comparison, all reference models were configured with their optimal parameter settings. Following best practices cited in other related papers, adjustments were made to comparison models to achieve optimal results. In specific experimental settings, the learning rate for the model was set to 0.009, with the number of iterations (epochs) at 300, and the number of attention heads in the multi-head attention mechanism set to 8. The model employs the adaptive optimizer Adam [40], with a dropout rate of 0.6, a regularization coefficient of 0.001, and ReLU as the activation function. The experiments were conducted with a random seed set to 72 to ensure the validity and repeatability of the experiment results.

5.3. Comparison Model Introduction

This experiment contrasts the accuracy of the model proposed in this study with other models to validate the effectiveness of the proposed model. The following provides a brief introduction to the models compared:

- Deep Walk [65]: A graph structure data mining method that combines Random Walk and the Word2Vec algorithm. It manages to learn the implicit structure information of networks and represents nodes in the graph as vectors containing latent structural information.
- Node2vec Model: A graph embedding method that considers neighborhood characteristics of Depth-First Search (DFS) and Breadth-First Search (BFS), acting as an extension of the Deep Walk algorithm.
- Graph Convolutional Network (GCN): GCN updates the embedding representation of each node by aggregating information from adjacent nodes, learning more complex feature representations by stacking multiple graph convolutional layers.
- GCNII [66]: Introduced to mitigate the oversmoothing phenomenon of GCN, the GCNII model enhances the model's generalization capacity by introducing adaptive aggregation and polynomial parameterization mechanisms, allowing for dynamic usage of different graph structures and tasks.
- Deep Graph Convolutional Neural Network (DeepGCN): Built upon stacking multiple layers of graph convolutional networks. It expands the range of information propagation by deepening graph convolution layers while effectively alleviating the oversmoothing issue through the introduction of identity mapping and residual structures.
- Graph Attention Network (GAT): The GAT model adaptively fuses neighborhood feature information through a self-attention mechanism and captures different feature subspaces through a multi-head attention mechanism, thereby enhancing the model's expressive power.

The experiments conducted on Cora, Citeseer, and Pubmed datasets, evenly split for training, validation, and testing, highlight the superior performance of the model proposed in this article.

5.4. Experimental Results

To ensure fairness in experimentation, the same strategy was employed for dataset division, conducting semi-supervised experiments on identical citation network datasets. The datasets were divided such that, for each category, 20 nodes were selected for training. The initial embeddings for training algorithms utilized the feature information of the

nodes. Moreover, 500 nodes were used for validation, and 1000 nodes were utilized for testing purposes. Table 2 below presents the performance results of different models on three datasets.

Table 2. Accuracy results of semi supervised classification experimental data.

Method	Cora	Citeseer	Pubmed
DeepWalk	67.2%	43.2%	65.3%
Node2vec	75.7%	64.7%	77.2%
GCN	81.5%	70.3%	79.0%
GCNII	82.6%	68.9%	78.8%
DeepGCN	84.3%	73.1%	79.2%
GAT	83.8%	72.0%	79.3%
TCC_GAT	86.1%	73.4%	83.8%

To offer a more vivid and direct visualization of the test results, these are presented in a bar chart in Figure 7.

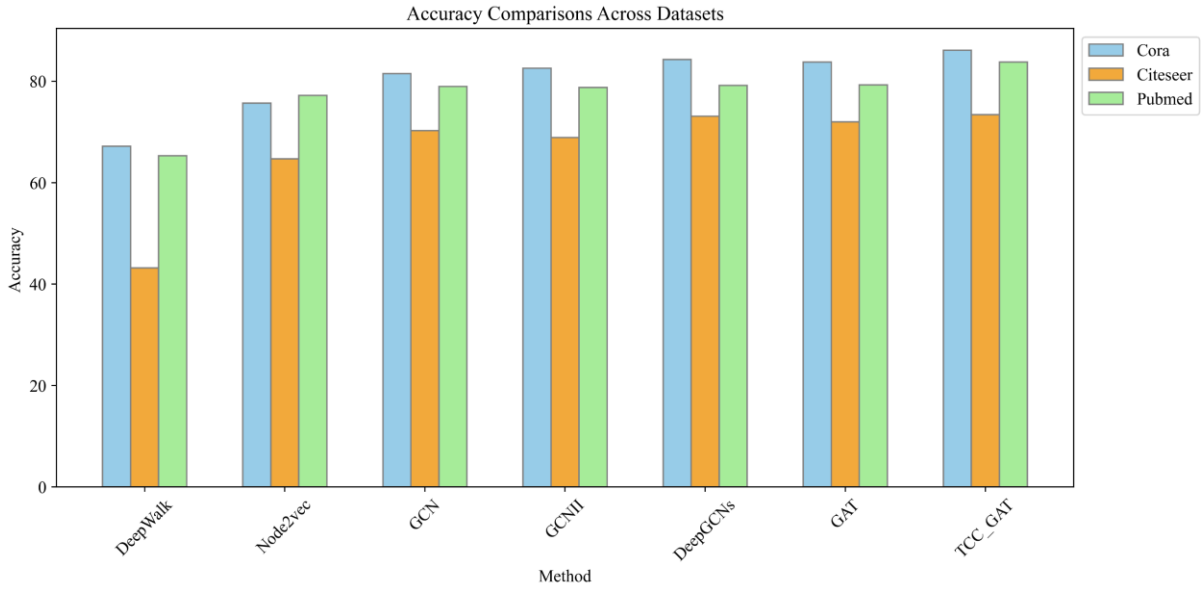


Figure 7. Model accuracy comparison across datasets.

It can be observed that the GCN model, which aggregates neighborhood information through the graph structure, shows a significant improvement in node classification performance compared to traditional methods based on random walks, confirming the superiority of graph convolutional networks in handling graph-structured data. The enhancement brought by GCNII indicates that improving the convolution method indeed aids in enhancing the understanding of graph structures, though this enhancement is relatively limited. More importantly, aggregating information from the graph structure fosters the generation of better node embeddings, thereby boosting the model's classification performance.

The DeepGCN model, through incorporating residual structures, extends the depth of the model and enhances its ability to aggregate higher-order neighborhood information, thereby achieving an improvement in overall performance. However, when handling large-scale datasets, the performance improvement remains marginal. In contrast, the GAT model, by introducing attention mechanisms, achieves significant performance enhancement with merely a shallow model structure. Comparative analysis between DeepGCN and GAT demonstrates that the traditional GAT model sufficiently captures crucial graph structural features at shallow levels, while attempts by DeepGCN to capture more hierarchical graph structural features by deepening the model do not result in significantly noticeable marginal benefits.

Building on this observation, the TCC_GAT model proposed in this work optimizes the graph structure on the basis of GAT. It effectively captures the graph structure and pays attention to higher-order neighborhood information. By employing a shallow model architecture, it avoids excessive reliance on deep-level features, thus achieving up to a 4.5% increase in classification performance as compared to the baseline GAT model.

5.5. Hyperparameter Discussions

This section addresses the hyperparameter settings, focusing on two crucial hyperparameters: learning rate and the choice of K for the K-nearest neighbors. The adjustment of the learning rate plays a pivotal role in the convergence speed of the algorithm: an excessively high learning rate may prevent the model from converging to the optimal point, whereas a learning rate that is too low could result in slow convergence, making it challenging to escape local optima, thereby directly impacting model accuracy. On the other hand, the choice of K in the TCC_GAT model proposed herein determines the number of neighboring nodes; selecting too many neighborhood nodes increases computational load, reducing convergence speed, while selecting too few may lead to insufficient learning of neighborhood information, affecting model efficiency.

To ensure the accuracy of the experimental data and the effectiveness of the experimental conditions, this study adopts a controlled variable approach to obtain experimental results. In the learning rate testing experiments, a fixed K value of 3 is used; similarly, in experiments to select the K value, the learning rate is fixed at 0.009 to ensure accurate experimental outcomes.

In the context of learning rate selection, the study conducted a series of tests by setting the learning rate within a numeric range from one to a hundred times 0.001. The experiment adhered to the parameter settings consistent with those mentioned earlier and varied model parameters one by one to assess the independent impact of each parameter. As shown in Figure 8, from a global perspective on data performance, the model exhibits peaks in performance at learning rate settings of 0.009 and 0.022. After conducting mean value tests ten times, it was observed that when the learning rate is set to 0.009, the model achieves the highest accuracy rate of 0.87, along with optimal stability.

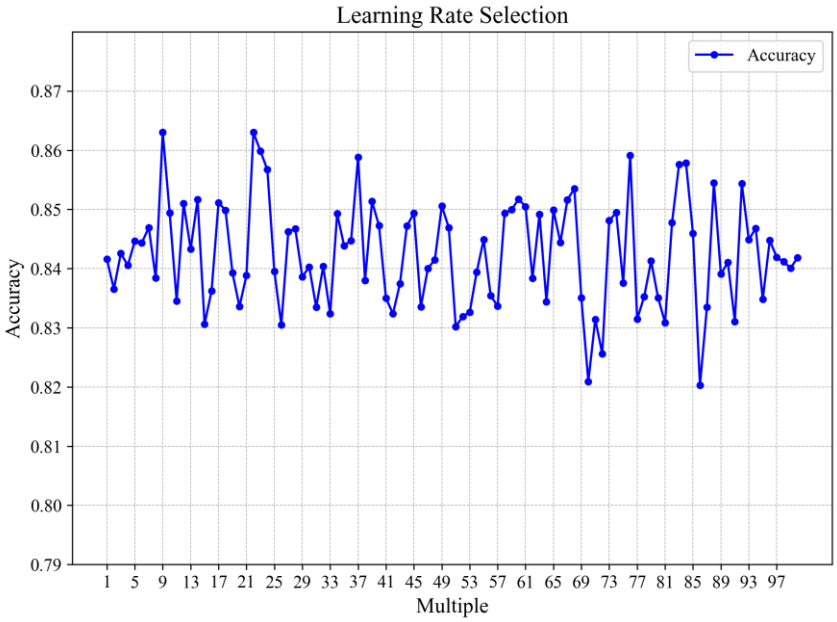


Figure 8. Learning rate selection.

To assess the impact of the hyperparameter K on model performance, this study conducted a comprehensive evaluation across a range of K values from 1 to 10. This was undertaken to explore the specific influence of varying numbers of neighboring nodes on

the performance of the proposed model when constructing the graph structure. Figure 9 presents the results of this series of experiments, where the horizontal axis represents the number of neighboring nodes selected by the K-nearest neighbors algorithm in constructing the graph structure, and the vertical axis denotes the model's accuracy. The results indicate that the model achieved the highest accuracy rate when the number of neighboring nodes was set to 3. As the value of K gradually increased, the accuracy of the model began to slowly decline. This trend was similarly validated in experiments conducted on other datasets. Increasing the number of neighboring nodes indeed contributes to enhancing the model's classification effect; however, as the number of neighbors further increases, so does the accompanying noise information, which in turn leads to a decline in model performance. These findings emphasize the need to balance enhancing model performance and controlling noise when selecting the value of K.

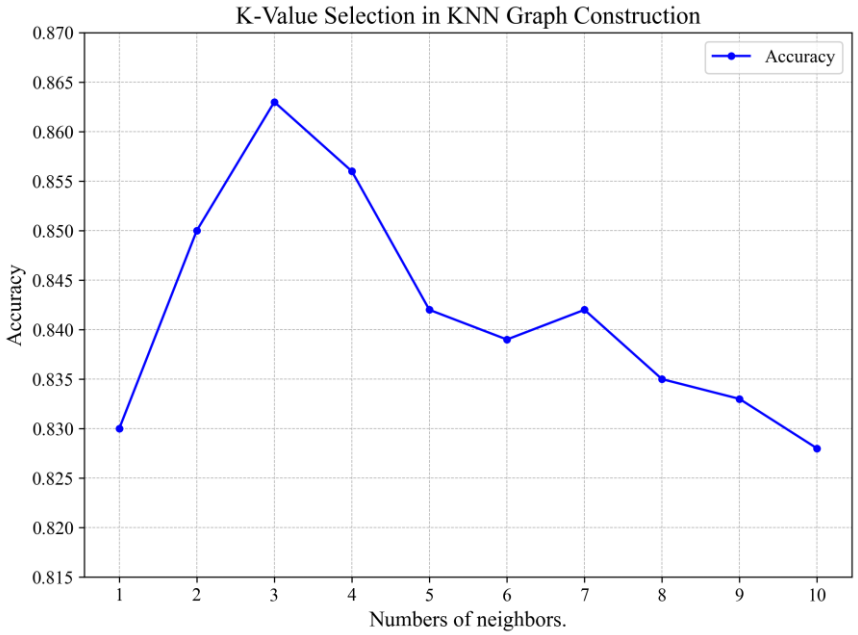


Figure 9. K numbers of neighbors.

Therefore, based on these experiments and analyses, this study ultimately determines to set the K value to 3 and the learning rate to 0.009 as the hyperparameter configuration for the model presented in this paper.

5.6. Time Complexity

Under identical experimental settings and dataset partition strategies, this study compared the TCC_GAT model proposed herein with the GAT and GCN models on the Cora, Citeseer, and Pubmed datasets in terms of training time. Given the distinct performances of these models on different graph structures, the improved graph structures proven effective for them were selected as the basis for model aggregation testing. Table 3 distinctly demonstrates that, compared to the GCN model with a simpler overall structure, the model presented in this paper exhibits certain discrepancies in convergence speed, primarily attributable to the advantages brought about by the smaller parameter volume of GCN. However, in the comparative analysis with the base model GAT, it is observed that the TCC_GAT algorithm designed in this paper, which improves graph structure information through parallel GAT, significantly reduces the model's training time.

Table 3. Time comparison table.

Method	Cora	Citeseer	Pubmed
GAT	67.5069 s	46.3596 s	162.0160 s
GCN	13.7267 s	16.2523 s	43.6845 s
TCC_GAT	23.0002 s	17.2497 s	55.5295 s

Specifically, for the Cora dataset, the training duration of TCC_GAT amounts to only 34% of that required by GAT. Similarly, on other datasets, this model also demonstrates enhanced time efficiency. These experimental results validate that the parallel strategy proposed in this study not only achieves effectiveness in enhancing model performance but also demonstrates the capability to improve time efficiency.

5.7. Ablation Experiment

To comprehensively assess the influence of each component of the model on the overall performance and its necessity, this paper designed and executed a series of ablation studies. The aim was to delve into the impact of different modules within the model on its overall performance by adjusting various components. In the comparative experiments, the GAT utilized the model's optimal parameter settings. To fully explore the model's potential, the maximum training epochs were set to 10,000, providing the model with ample learning time. Additionally, to enhance training efficiency and prevent overfitting, an early stopping mechanism was introduced. Training automatically ceases when there is no significant improvement in model performance over 100 consecutive training epochs. The obtained results of ablation experiments is shown in Table 4.

Table 4. Results of ablation experiments.

Dataset	Metrics	GAT	GAT-A	GAT- B	GAT-C	TCC_GAT
Cora	F1	82.25%	82.72%	80.11%	83.50%	85.19%
	precision	82.52%	82.99%	80.10%	82.74%	84.70%
	recall	82.15%	82.60%	80.55%	84.60%	85.70%
	accuracy	83.80%	84.30%	82.20%	84.70%	86.10%
Citeseer	F1	66.95%	66.96%	67.10%	67.02%	68.12%
	precision	68.60%	69.10%	64.60%	65.89%	70.00%
	recall	71.81%	71.88%	72.60%	72.66%	73.38%
	accuracy	71.81%	71.80%	72.60%	72.88%	73.40%
Pubmed	F1	78.30%	78.64%	81.10%	82.51%	83.17%
	precision	78.42%	78.89%	81.20%	82.62%	83.30%
	recall	78.18%	78.53%	81.56%	82.50%	83.12%
	accuracy	79.30%	79.91%	82.10%	83.00%	83.80%

Experimental results, as presented in Table 4, offer a comprehensive display of various model configurations' performance across different evaluation metrics. Through an in-depth analysis of these results, a clear understanding of each module's contribution to the overall model performance is achieved, providing an important basis for further optimization and improvement of the model.

Descriptions of models in Table 4:

- GAT: The GAT model represents a baseline single-channel setup, employing the original node interaction behavior graph as input. It serves as a benchmark in this study to assess the performance of other enhanced models.
- GAT-A: The GAT-A model, an improved single-channel GAT setup, utilizes cosine similarity to enhance the basic interaction behaviors, generating an enhanced graph. This graph forms the basis for model aggregation aimed at capturing richer inter-node relationships and assessing the effectiveness of high-order neighborhood information.

- GAT-B: The GAT-B model, another single-channel GAT setup, employs the K-nearest neighbors algorithm to construct a similarity-based graph structure for aggregation. This approach explores the impact of similarity-based graph structures on model performance.
- GAT-C: The GAT-C model is a dual-channel classification model combining two distinct graph structures. It initially performs GAT aggregation using the basic node interaction behavior graph, then fuses the embedding representations with the outputs from the GAT-B model. This design aims to assess whether different graph structures can complement each other's model deficiencies, thereby demonstrating complementary advantages.
- TCC_GAT: The TCC_GAT model is a Two-Channel Classification Algorithm Based on Graph Attention Network proposed in this study.

Analysis of Experimental Results:

Across all classification metrics, the GAT-A model exhibited superior performance compared to the baseline GAT model. This result strongly confirms the effectiveness of enhancing the basic graph structure using similar interaction behavior nodes. By introducing cosine similarity, the model is enabled to aggregate high-order neighborhood information directly, thus improving classification performance.

The GAT-B model exhibited inconsistent performance across different datasets. On the Cora dataset, GAT-B's performance was below that of models using the original graph structure. However, on datasets with richer citation information (such as Citeseer and Pubmed), GAT-B outperformed GAT using original citation behaviors. This phenomenon indicates that the effectiveness of reconstructing the paper citation graph structure using the K-nearest neighbors algorithm largely depends on the richness of edge information. On datasets with denser edge information, this reconstruction method can bring more significant performance improvements.

The primary objective of designing the GAT-C model was to validate a hypothesis: the combined effects of multiple models might surpass that of a singular model. Experimental results robustly support this hypothesis. By integrating features from both the basic GAT and GAT-B, the GAT-C model, despite the deficiencies in performance for both GAT and GAT-B in a single-channel setup, significantly outperformed single models on multiple metrics when integrated into a dual-channel setup.

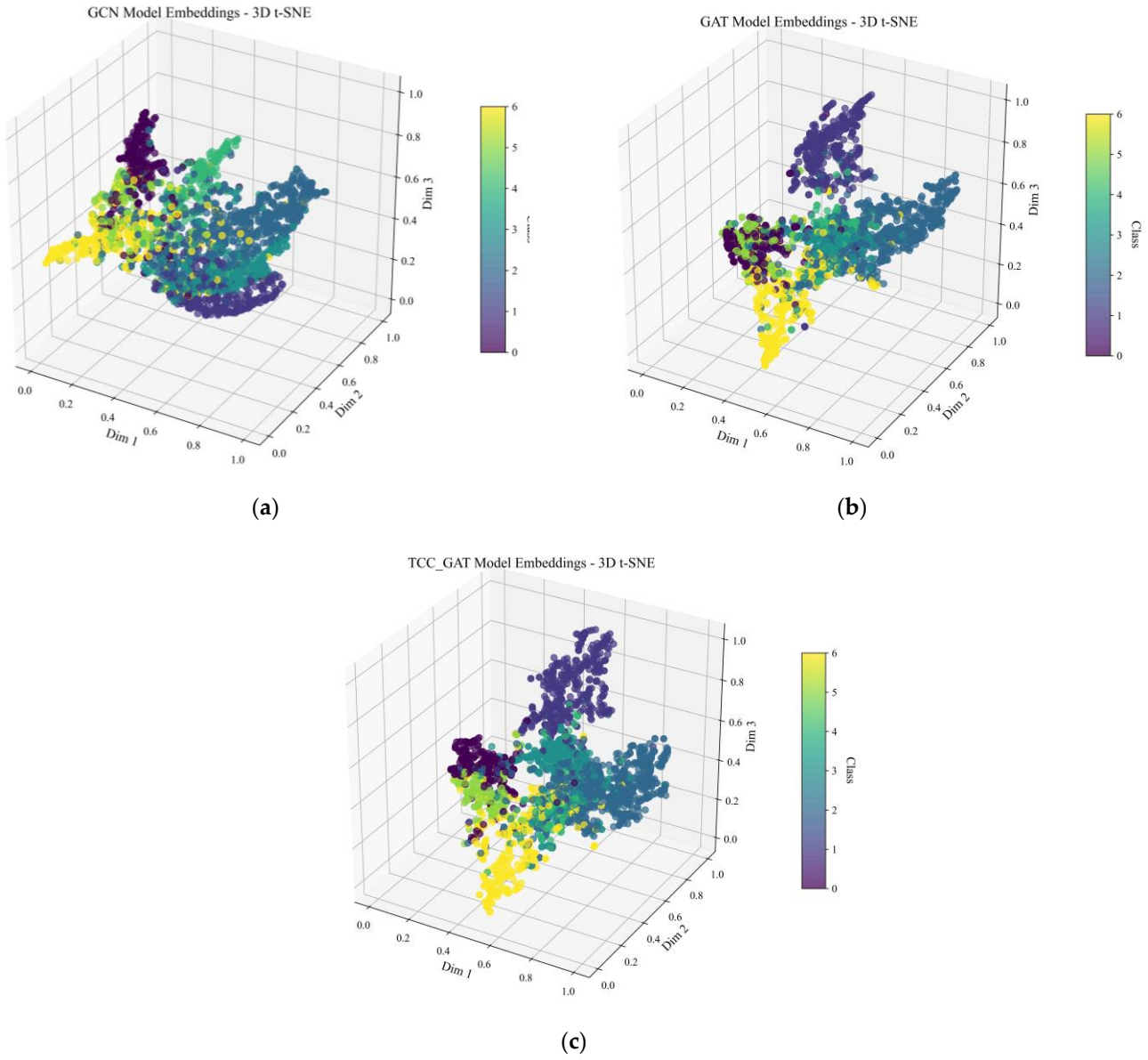
Finally, the TCC_GAT model proposed in this paper achieved the best performance across all three datasets. This result not only validates the effectiveness of the proposed method but also highlights its stability and adaptability across different data environments.

5.8. Visualization Experiments

In node classification tasks within the realm of graph neural networks, models commonly leverage the wealth of graph structure information and neighborhood features to enhance node identification, ultimately aiming to improve classification accuracy. However, since node representations are typically high-dimensional data, assessing the quality of model embeddings in a straightforward manner is challenging. To evaluate the effectiveness of these generated embeddings, visualization techniques are employed. Nevertheless, directly evaluating the quality of model embeddings based on feature vectors, due to their high-dimensional nature, is not intuitive.

Therefore, this study conducted a visualization analysis of embeddings generated by the TCC_GAT model on the Cora dataset. Comparative models include GCN and GAT. The nonlinear dimensionality reduction algorithm t-SNE (t-distributed stochastic neighbor embedding) [67,68] was utilized to reduce high-dimensional data, projecting feature vectors of different nodes into a three-dimensional space for visualization. Through these visualization results, it becomes possible to more intuitively observe the distribution and clustering effects of embeddings generated by different models, thus assessing the performance of each model in terms of node representation learning.

Figure 10 illustrates the performance of three models on the Cora dataset, presented through equally proportioned axes in a three-dimensional space. Identical colors represent the same labels, and the color distribution clearly reveals the visual effects of the classification by the three models.



(**a**) (**b**)

(**c**)

Figure 10. 3D visualization of Cora dataset using t-SNE. (**a**) GCN model embeddings-3D t-SNE; (**b**) GAT model embeddings-3D t-SNE; (**c**) TCC_GAT model embeddings-3D t-SNE.

For the GCN model, although it manages to distinguish between different node categories to a certain extent, the boundaries between categories are not well-defined, resulting in numerous misclassifications. Additionally, the projection of all nodes in the three-dimensional space appears overly dense, indicating a lack of significant differentiation between the embeddings of different categories. On the other hand, the GAT model exhibits a clearer comparative effect in classification. By incorporating an attention mechanism, the model is better able to comprehend the characteristics of the graph structure, thus producing more distinct node embeddings and creating gaps between different clusters. Nonetheless, misclassifications still occur among categories with fewer data points (projecting into the same space), as evidenced by the difficulty in precisely distinguishing between categories 5 and 0 in the figure, lacking a clear division.

In comparison, the model proposed in this study renders a more coherent visualization, forming a congregated state. Within the three-dimensional space, different categories are situated in their respective areas with almost no significant overlap, showcasing the model's superiority. This also validates the effectiveness of graph structure information and higher-order neighborhood information in enhancing the embedding representations of nodes, thereby improving the model's performance on node classification tasks.

6. Conclusions

This study introduces a Two-Channel Classification Algorithm Based on Graph Attention Network (TCC_GAT), aimed at enhancing the performance of GNN in node classification tasks. Generally, GNN models attempt to improve model performance by incorporating high-order neighborhood structures and feature information. However, increasing the model's depth often leads to oversmoothing. To address this issue, this article proposes an improved strategy for acquiring high-order information. By exploiting cosine similarity to explore the similarity in interaction behaviors among nodes, the algorithm identifies nodes with strong correlations to reinforce the basic graph structure, thereby enhancing its ability to aggregate high-order neighborhood information while retaining the basic graph structure. Subsequently, the graph structure is reconstructed using the similarity matrix and the K-nearest neighbor algorithm, generating a new graph structure that reflects the similarity in interaction behaviors. This further enhances the model's ability to aggregate structural information and high-order neighborhood information. Utilizing parallel GAT architecture, this study effectively captures the embedding representations of identical nodes across different structures and updates the embedding representation of the target node through a merging process.

Comprehensive experimental evaluations, tested across three datasets, demonstrate that the proposed model significantly improves various performance metrics, including accuracy, when compared to baseline models. Notably, the model's convergence rate is more than double that of the conventional GAT model, reinforcing its superiority in performance.

Nonetheless, there are certain limitations in the generation of node embeddings. Specifically, the model only utilizes the attention coefficient of the first-order neighborhood to filter neighboring nodes, thus overlooking the importance of higher-order neighborhoods relative to the target node. This oversight may lead to a deficiency in node feature information, impacting the effectiveness of neighborhood information aggregation and consequently affecting classification performance. To overcome this challenge, future research will consider adjusting the importance coefficient of nodes based on the order of neighboring nodes and their aggregation, increasing the depth of the graph neural network to further optimize model performance, aiming to achieve more accurate and robust node classification results.

Author Contributions: Conceptualization, Y.W. and L.Y.; methodology, L.Y., X.W. and G.Z.; software, Y.W.; validation, Y.W., G.Z. and L.Y.; formal analysis, Y.W.; resources, Y.W.; data curation, Y.W; writing—original draft preparation, X.W. and Y.W.; writing—review and editing, L.Y. and W.D.; visualization, Y.W; funding acquisition, L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: Author Xiaolong Wang was employed by the company Oneg Robot Yinchuan Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Chitradevi, B.; Srimathi, P. An overview on image processing techniques. *Int. J. Innov. Res. Comput. Commun. Eng.* **2014**, *2*, 6466–6472.
2. Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 6999–7019. [CrossRef] [PubMed]
3. Zhou, T.; Porikli, F.; Crandall, D.J.; Van Gool, L.; Wang, W. A survey on deep learning technique for video segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 7099–7122. [CrossRef] [PubMed]
4. Michelsanti, D.; Tan, Z.H.; Zhang, S.X.; Xu, Y.; Yu, M.; Yu, D.; Jensen, J. An overview of deep-learning-based audio-visual speech enhancement and separation. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2021**, *29*, 1368–1396. [CrossRef]
5. Ran, X.J.; Suyaroj, N.; Tepsan, W.; Ma, J.H.; Zhou, X.B.; Deng, W. A hybrid genetic-fuzzy ant colony optimization algorithm for automatic K-means clustering in urban global positioning system. *Eng. Appl. Artiffcial Intell.* **2024**, *137*, 109237. [CrossRef]
6. Sarzynska-Wawer, J.; Wawer, A.; Pawlak, A.; Szymanowska, J.; Stefaniak, I.; Jarkiewicz, M.; Okruszek, L. Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Res.* **2021**, *304*, 114135. [CrossRef]
7. Lee, J.; Toutanova, K. Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
8. Deng, W.; Li, X.Y.; Xu, J.J.; Li, W.H.; Zhu, G.T.; Zhao, H.M. BFKD: Blockchain-based federated knowledge distillation for aviation Internet of Things. *IEEE T. Reliab.* **2024**.
9. Gao, L.; Wang, H.; Zhang, Z.; Zhuang, H.; Zhou, B. HetInf: Social influence prediction with heterogeneous graph neural network. *Front. Phys.* **2022**, *9*, 787185. [CrossRef]
10. Johnson, F.; Adebukola, O.; Ojo, O.; Alaba, A.; Victor, O. A task performance and fitness predictive model based on neuro-fuzzy modeling. *Artif. Intell. Appl.* **2024**, *2*, 66–72. [CrossRef]
11. Jiang, Y. *Information Fusion Recommendation Based on Convolutional Graph and Neural Collaborative Filtering*; Jilin University: Changchun, China, 2018.
12. Yin, L.; Chen, P.; Zheng, G. Session-Enhanced Graph Neural Network Recommendation Model (SE-GNNRM). *Appl. Sci.* **2022**, *12*, 4314. [CrossRef]
13. Zhao, H.; Gao, Y.; Deng, W. Defect detection using shuffle Net-CA-SSD lightweight network for turbine blades in IoT. *IEEE Internet Things J.* **2024**. [CrossRef]
14. Yin, L.; Chen, P.; Zheng, G. Recommendation Algorithm for Multi-Task Learning with Directed Graph Convolutional Networks. *Appl. Sci.* **2022**, *12*, 8956. [CrossRef]
15. Chen, H.; Ru, J.; Long, H.; He, J.; Chen, T.; Deng, W. Semi-supervised adaptive pseudo-label feature learning for hyperspectral image classification in internet of things. *IEEE Internet Things J.* **2024**, *11*, 30754–30768. [CrossRef]
16. Li, W.; Liu, D.; Li, Y.; Hou, M.; Liu, J.; Zhao, Z.; Guo, A.; Zhao, H.; Deng, W. Fault diagnosis using variational autoencoder GAN and focal loss CNN under unbalanced data. *Struct. Health Monit.* **2024**. [CrossRef]
17. Bhosle, K.; Musande, V. Evaluation of deep learning CNN Model for recognition of Devanagari digit. *Artif. Intell. Appl.* **2023**, *1*, 114–118. [CrossRef]
18. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
19. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1–9. [CrossRef]
20. Lin, Y.; Guo, D.; Wu, Y.; Li, L.; Wu, E.Q.; Ge, W. Fuel consumption prediction for pre-departure flights using attention-based multi-modal fusion. *Inf. Fusion* **2024**, *101*, 101983. [CrossRef]
21. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
22. Preethi, P.; Mamatha, H.R. Region-based convolutional neural network for segmenting text in epigraphical images. *Artif. Intell. Appl.* **2023**, *1*, 119–127. [CrossRef]
23. Yan, S.; Shao, H.; Wang, J.; Zheng, X.; Liu, B. LiConvFormer: A lightweight fault diagnosis framework using separable multiscale convolution and broadcast self-attention. *Expert Syst. Appl.* **2024**, *237*, 121338. [CrossRef]
24. Guo, D.; Wu, E.Q.; Wu, Y.; Zhang, J.; Law, R.; Lin, Y. FlightBERT: Binary Encoding Representation for Flight Trajectory Prediction. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 1828–1842. [CrossRef]
25. Wang, Z.; Wang, Q.; Liu, Z.; Wu, T. A deep learning interpretable model for river dissolved oxygen multi-step and interval prediction based on multi-source data fusion. *J. Hydrol.* **2024**, *629*, 130637. [CrossRef]
26. Li, T.Y.; Shu, X.Y.; Wu, J.; Zheng, Q.X.; Lv, X.; Xu, J.X. Adaptive weighted ensemble clustering via kernel learning and local information preservation. *Knowl.-Based Syst.* **2024**, *294*, 111793. [CrossRef]
27. Yan, Z.; Yang, H.; Guo, D.; Lin, Y. Improving airport arrival flow prediction considering heterogeneous and dynamic network dependencies. *Inf. Fusion.* **2023**, *100*, 101924. [CrossRef]
28. Li, M.; Lv, Z.; Cao, Q.; Gao, J.; Hu, B. Automatic assessment method and device for depression symptom severity based on emotional facial expression and pupil-wave. *IEEE Trans. Instrum. Meas.* **2024**, *20*, 42. [CrossRef]
29. Li, X.; Zhao, H.; Deng, W. IOFL: Intelligent-optimization-based federated learning for Non-IID data. *IEEE Internet Things J.* **2024**, *11*, 16693–16699. [CrossRef]

30. Bhatti, U.A.; Tang, H.; Wu, G.; Marjan, S.; Hussain, A. Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence. *Int. J. Intell. Syst.* **2023**, *2023*, 8342104. [CrossRef]
31. Lu, Y.; Chen, Y.; Zhao, D.; Liu, B.; Lai, Z.; Chen, J. CNN-G: Convolutional neural network combined with graph for image segmentation with theoretical analysis. *IEEE Trans. Cogn. Dev. Syst.* **2020**, *13*, 631–644. [CrossRef]
32. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
33. Wu, Z.; Zhan, M.; Zhang, H.; Luo, Q.; Tang, K. MTGCN: A multi-task approach for node classification and link prediction in graph data. *Inf. Process. Manag.* **2022**, *59*, 102902. [CrossRef]
34. Sun, Q.; Chen, J.; Zhou, L.; Ding, S.; Han, S. A study on ice resistance prediction based on deep learning data generation method. *Ocean. Eng.* **2024**, *301*, 117467. [CrossRef]
35. Shao, H.; Zhou, X.; Lin, J.; Liu, B. Few-shot cross-domain fault diagnosis of bearing driven by Task-supervised ANIL. *IEEE Internet Things J.* **2024**, *11*, 22892–22902. [CrossRef]
36. Li, B.; Wu, J.; Pi, D.; Lin, Y. Dual mutual robust graph convolutional network for weakly supervised node classification in social networks of Internet of People. *IEEE Internet Things J.* **2021**, *10*, 14798–14809. [CrossRef]
37. Song, Y.J.; Han, L.H.; Zhang, B.; Deng, W. A dual-time dual-population multi-objective evolutionary algorithm with application to the portfolio optimization problem. *Eng. Appl. Artiffcial Intell.* **2024**, *133*, 108638. [CrossRef]
38. Xie, Y.; Yao, C.; Gong, M.; Chen, C.; Qin, A. Graph convolutional networks with multi-level coarsening for graph classification. *Knowl.-Based Syst.* **2020**, *194*, 105578. [CrossRef]
39. Xu, J.; Li, T.; Zhang, D.; Wu, J. Ensemble clustering via fusing global and local structure information. *Expert Syst. Appl.* **2024**, *237*, 121557. [CrossRef]
40. Li, M.; Wang, Y.Q.; Yang, C.; Lu, Z.; Chen, J. Automatic diagnosis of depression based on facial expression information and deep convolutional neural network. *IEEE Trans. Comput. Soc. Syst.* **2024**, 1–12. [CrossRef]
41. Xiao, Y.; Shao, H.; Lin, J.; Huo, Z.; Liu, B. BCE-FL: A secure and privacy-preserving federated learning system for device fault diagnosis under Non-IID Condition in IIoT. *IEEE Internet Things J.* **2024**, *11*, 14241–14252. [CrossRef]
42. Wang, J.; Shao, H.; Peng, Y.; Liu, B. PSparseFormer: Enhancing fault feature extraction based on parallel sparse self-attention and multiscale broadcast feed-forward block. *IEEE Internet Things J.* **2024**, *11*, 22982–22991. [CrossRef]
43. Ma, Y.; Wang, S.; Aggarwal, C.C.; Tang, J. Graph convolutional networks with eigenpooling. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 723–731.
44. Li, Q.; Han, Z.; Wu, X.M. Deeper insights into graph convolutional networks for semi-supervised learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
45. Zhu, Z.; Zhang, Z.; Xhonneux, L.P.; Tang, J. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 29476–29490.
46. Long, Y.; Wu, M.; Liu, Y.; Fang, Y.; Kwoh, C.K.; Chen, J.; Luo, J.; Li, X. Pre-training graph neural networks for link prediction in biomedical networks. *Bioinformatics* **2022**, *38*, 2254–2262. [CrossRef] [PubMed]
47. Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; Weinberger, K. Simplifying graph convolutional networks. *Int. Conf. Mach. Learn. PMLR* **2019**, *97*, 6861–6871.
48. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
49. Wang, X.; Zhu, M.; Bo, D.; Cui, P.; Shi, C.; Pei, J. Am-gcn: Adaptive multi-channel graph convolutional networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 6–10 July 2020; pp. 1243–1253.
50. Hong, Y.; Rodriguez, C.; Qi, Y.; Wu, Q.; Gould, S. Language and visual entity relationship graph for agent navigation. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7685–7696.
51. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
52. Li, G.; Muller, M.; Thabet, A.; Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9267–9276.
53. Li, F.; Chen, J.; Zhou, L.; Kujala, P. Investigation of ice wedge bearing capacity based on an anisotropic beam analogy. *Ocean. Eng.* **2024**, *302*, 117611. [CrossRef]
54. Zhao, H.; Wang, L.; Zhao, Z.; Deng, W. A new fault diagnosis approach using parameterized time-reassigned multisynchrosqueezing transform for rolling bearings. *IEEE Trans. Reliab.* **2024**, 1–10. [CrossRef]
55. Xie, P.; Deng, L.; Ma, Y.; Deng, W.Q. EV-Call 120: A new-generation emergency medical service system in China. *J. Transl. Intern. Med.* **2024**, *12*, 209–212. [CrossRef]
56. Deng, W.; Chen, X.; Li, X.; Zhao, H. Adaptive federated learning with negative inner product aggregation. *IEEE Internet Things J.* **2023**, *11*, 6570–6581. [CrossRef]
57. Gao, J.; Wang, Z.; Jin, T.; Cheng, J.; Lei, Z.; Gao, S. Information gain ratio-based subfeature grouping empowers particle swarm optimization for feature selection. *Knowl.-Based Syst.* **2024**, *286*, 111380. [CrossRef]
58. Huang, C.; Wu, D.Q.; Zhou, X.B.; Song, Y.J.; Chen, H.L.; Deng, W. Competitive swarm optimizer with dynamic multi-competitions and convergence accelerator for large-scale optimization problems. *Appl. Soft Comput.* **2024**, *167*, 112252. [CrossRef]
59. Rongmei, Z.; Jiahui, Z. Research review of graph neural network technology. *J. Hebei Acad. Sci.* **2022**, *39*, 1–13.

60. Vaswani, A. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
61. Phan, V.M.H.; Xie, Y.; Zhang, B.; Qi, Y.; Liao, Z.; Perperidis, A.; Phung, S.L.; Verjans, J.W.; To, M.-S. Structural Attention: Rethinking Transformer for Unpaired Medical Image Synthesis. *arXiv* **2024**, arXiv:2406.18967.
62. Ge, C.; Song, Y.; Ma, C.; Qi, Y.; Luo, P. Rethinking attentive object detection via neural attention learning. *IEEE Trans. Image Process.* **2023**, *33*, 1726–1739. [CrossRef] [PubMed]
63. Lee, J.; Sun, M.; Lebanon, G. A comparative study of collaborative filtering algorithms. *arXiv* **2012**, arXiv:1205.3193.
64. Kingma, D.P. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
65. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
66. Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; Li, Y. Simple and deep graph convolutional networks. *Int. Conf. Mach. Learn. PMLR* **2020**, *119*, 1725–1735.
67. Malmqvist, L.; Yuan, T.; Manandhar, S. Visualising argumentation graphs with graph embeddings and t-SNE. *arXiv* **2021**, arXiv:2107.00528.
68. Long, H.; Chen, T.; Chen, H.; Zhou, X.; Deng, W. Principal space approximation ensemble discriminative marginalized least-squares regression for hyperspectral image classification. *Eng. Appl. Artif. Intell.* **2024**, *133*, 108031. [CrossRef]