Adaptive Aggregation-Transformation Decoupled Graph Convolutional Network for Semi-Supervised Learning

DEZHI SUN, MAN HU, AND ZHENYU LI
Beijing Institute of Graphic Communication. Beijing 100000, China
Corresponding author: Man Hu (e-mail: human190707@163.com)
This work was supported by initial funding for the Doctoral Program of BIGC (27170120003/022), BIGC Project (Eb202004, Ec202007).

ABSTRACT Graph Convolutional Network (GCN) has achieved significant success in many graph representation learning tasks. GCN usually learns graph representations by performing Neighbor Aggregation (NA) and Feature Transformation (FT) operations. Deep Adaptive Graph Neural Network (DAGNN) improves NA operation in the aggregation-transformation decoupled GCN, which enables the model to obtain a large receiver field. However, the problem with GCN is that when the model is trained to a deeper level, the performance decreases. In particular, the influence of NA and FT on model degradation has not been fully considered. In this work, we propose a new decoupled GCN architecture to enhance the performance of deep GCN. First, we conduct an experimental analysis of the impact of NA and FT operations on the degradation of the deep GCN model. Subsequently, we propose an Adaptive Aggregation-Transform Decoupled Graph Convolutional Network (AATD-GCN) which divides the model into two depths $D_{NA}$ and $D_{FT}$, and proposes improved approaches in NA and FT, respectively. The AATD-GCN reduces the influence of NA and FT operations on performance degradation of deep GCN, and obtains a large receiver field while extracting complex feature information. We investigate the requirements of model depth $D_{NA}$ and $D_{FT}$ for graphs of different structures and sizes. Finally, the effectiveness of the proposed architecture is verified through extensive experiments on real-world datasets, the experimental results show that AATD-GCN is superior in terms of accuracy and robustness.

INDEX TERMS deep learning, graph convolutional networks, graph representation learning, node classification

I. INTRODUCTION

A graph is composed of nodes and edges, representing entities and the relationships between them. Graphs are widely used in social networks, knowledge graphs, and protein networks. Many researchers have recently attempted to apply deep learning methods to graph data, which has accelerated the development of graph neural networks. The graph neural network method based on deep learning has been successful in many applications such as link prediction, graph classification, and node classification tasks. The graph convolution operation learns the representation of a node by iteratively propagating neighbor information to the target node [1]. The most representative method is Graph Convolutional Network (GCN) [2], which learns the representation of nodes by iteratively aggregating the representations of neighboring nodes. However, GCN and its variants all face a common limitation in that one-layer graph convolution operations can only aggregate 1-hop neighbor nodes information. Stacking multiple layers to improve the receptive fields to aggregate multihop neighbor node information causes a significant performance decline. Several recent studies have attributed this decline in performance to over-smoothing [3], model degradation [4], and the coupling of Neighbor Aggregation (NA) and Feature Transformation (FT) in the graph convolution operation [5]. These studies state that different classes of node features become indistinguishable after multiple neighbor aggregations, multiple feature transformations on a small dataset will weaken the generalization ability of the network, and the coupling of aggregations and transformations of each layer of graph convolution will cause performance decline. Deep Adaptive Graph Neural Network (DAGNN) [5] is proposed to reduce the model degradation and over-smoothing prob-
lems in deep GCNs. There are other networks such as APPNP [6], SGC [7], AP-GCN [8], and DGMLP [4]. In this study, we considered the impact of NA and FT on the performance decline in deep GCNs.

GCN has been successfully applied in node classification, link prediction, and graph classification. GCN use neighbor aggregation to learn node representations while considering the node feature information and graph topology. GCN can expand receptive fields to gather more distant neighbor information as the number of layers increases. However, the performance of the model decrease with deeper GC layers. To solve the degradation of model performance in graph representation learning by the GCN when the number of layers increases, we performed the following work.

We first decoupling NA and FT operations in GCN and design an NA-GCN and FT-GCN, followed by a comparative analysis of the NA and FT impacts on the performance decline when the GCN goes deeper. We observed and argued that performance declines when the GCN goes deeper, not only relate to over-smoothing caused by multiple neighbor aggregations, but also to model degradation caused by multiple feature transformations. Through theoretical analysis and experimental verification, we found that when graph structure data are sparse, we need to increase the number of neighbor aggregations to obtain larger receptive fields and improve the performance of deep GCNs. When graph structure data are large, we need to increase the number of feature transformations to better extract features and improve the performance of deep GCNs. We propose a more robust deep graph neural network called the Adaptive Aggregation-Transformation Decoupled Graph Convolutional Network (AATD-GCN). First, it adaptively extracts different feature information from the input graph feature information through multilayer feature transformation, and then adaptively aggregates the extracted feature information from large receptive fields to obtain the final node representation. We perform an extensive experiment on the citation, co-authorship, and co-purchase datasets to demonstrate that our model performs better in terms of accuracy and robustness.

The contributions of this study are three-fold:

- We conduct analytical experiments on Feature Transformation (FT) operations and Neighbor Aggregation (NA) operations in GCN using two models, FT-GCN and NA-GCN. We empirically find that the number of NA and FT operations required varies between different graph structure data.

- We propose an Adaptive Aggregation-Transformation Decoupled Graph Convolutional Network (AATD-GCN), which has two depths $D_{NA}$ and $D_{FT}$, and can be set with different NA and FT layers according to different datasets to make the model more flexible. The application of techniques such as the adaptive adjustment mechanism also makes the AATD-GCN more robust to faces with different $D_{NA}$ and $D_{FT}$ values.

- The proposed AATD-GCN is a well-resolved model degradation problem when the $D_{NA}$ and $D_{FT}$ becomes deeper. It can extract more complex feature information while obtaining a larger received field and achieving a superior performance on the benchmark dataset.

II. RELATED WORK

Notation: Given an unweighted and undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the node set consisting of nodes $\{v_1, \cdots, v_n\}$, $n$ represents the total number of nodes. $\mathcal{E}$ is the set of edges between node $i$ and node $j$, and the total number of edges is $m$. Let $A \in \mathbb{R}^{n \times n}$ denote the adjacency matrix of graph $G$, where $A_{i,j} = 1$ if an edge exists between node $i$ and node $j$; otherwise, it is 0. The diagonal degree matrix of $G$ is defined as $D \in \mathbb{R}^{n \times n}$, where $D_{i,i} = \sum_j A_{i,j} N_i$ denotes the neighboring node set of node $i$. The node features matrix is denoted as $X = [x_1, \cdots, x_n] \in \mathbb{R}^{n \times d}$, each node $v$ is associate with a $d$-dimensional feature vector $x_v$.

A. GRAPH CONVOLUTIONAL NETWORK (GCN)

Currently, networks based on graph convolution operations are generally built on a message-passing framework [9]. Graph convolution operations usually adopt the neighborhood aggregation method to obtain node representation, first aggregation representation of its neighbors, and then apply feature transformation [5]. Similar to CNN and MLP, GCN pass through multiple layers to learn the node feature representation, which is then passed as an input to a linear classifier. The L-layer GCN model is composed of L-layer feed-forward Graph Convolution (GC) layers. Formally, the $l$-th GC layer can be represented as:

$$X^{(l)} = \sigma \left( \tilde{A} X^{(l-1)} W^{(l)} \right)$$  \hspace{1cm} (1)

Where $X^{(l-1)} \in \mathbb{R}^{n \times d_{l-1}}$ and $W^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ denote the node feature matrix and the shared learnable weight matrix of this layer. The $i$-th row ($i \in \{1, \cdots, n\}$) of $X^{(l)}$, denoted as $x^{(l)}_i$, represents the output features of node $i$ of the $l$-th layer. $\tilde{A} = D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}}$, where $\hat{A} = A + I$ is an adjacency matrix with added self-connections. $D_{i,i} = \sum_j \hat{A}_{i,j}$ is the diagonal node degree matrix of $\hat{A}$. $\sigma$ is a nonlinear activation function similar to that of the ReLU [10].

B. DEEP GCNs

Generally, by stacking more GC layers, node information can be propagated to farther distance nodes, thereby improving the performance of deep GCNs. However, some practices [1], [11]–[15] have shown that stacking more GC layers causes serious performance degradation. Several studies attribute the performance decline caused by over-smoothing problem, which means that different classes of node presentation converge to indistinguishable when there is a multitude of neighbor aggregation activities. The authors attempted to explore the role of over-smoothing in the GCN model [1]. The authors first demonstrated that the propagation process of the GCN model is a special symmetric form of Laplacian smoothing [16], which makes the representations of nodes in the same class similar, making classification
much easier. They showed that repeatedly stacking many convolution layers may cause an over-smoothing problem. Recently, SGC [7] proposed a simplified graph convolution operation that removed unnecessary complex operations. The authors showed that SGC is equivalent to a low-pass filter in the spectral domain that extracts smoothing features from the graph. Another author [11] demonstrated that the essence of the most popular graph convolution operation is smoothing features and that reasonable smoothing makes graph convolution work well, whereas excessive smoothing causes performance degradation. DropEdge [14] proposed an extension of dropout to graph neural networks. Unlike dropout, which randomly removed specific graph structure data attributes, DropEdge randomly removed some edges from the adjacency matrix $A$, and then used the remaining adjacency matrix for GCN training. To maintain a consistent overall distance, PairNorm [15] centered and re-stretched the output of each GCN layer.

In addition to the performance degradation caused by over-smoothing, some studies have pointed out that multiple feature transformations can also cause a series of problems, and thus, performance degradation. The authors have demonstrated this [17], the feature transformation in GCN amplifies node-wise feature variation and has a greater effect on model performance degradation than neighbor aggregation. Node Normalization (Node Norm), a variance-controlling approach that modifies each node’s features using its standard deviation, was then proposed. The author proposed Jumping Knowledge Networks [18] employ a layer aggregation mechanism to combine the outputs of each GCN layer as the final output, and the aggregation function used to combine the outputs of each GCN layer can be concatenated, maximum pooling, or LSTM [19].

### C. DECOUPLED GRAPH CONVOLUTIONAL NETWORK

In this subsection, we explain the meaning of aggregation-transformation decoupled GCN, define the Neighbor Aggregation (NA) and Feature Transformation (FT) operations in GCN with formulas, and show them vividly with Fig. 1. Finally, we introduce the work related to aggregation-transformation decoupled GCN.

In the original GCN [2] and part of its variants [20]–[22], NA and FT are coupled in each layer. In other words, when the number of layers in one of NA and FT increases, the other must increase as well. However, some recent works show that this coupling design is not necessary and propose to separate NA and FT operations so that they can be added separately without being influenced by each other.

According to Fig. 1, one GC layer is divided into two basic operations: Neighbor Aggregation (NA) and Feature Transformation (FT) [4], [7]. The operation of the $l$-th layer in graph convolution can be described as:

\[
\begin{align*}
\hat{X}^{(l-1)} &= \hat{A}X^{(l-1)} \quad \text{(NA)} \\
X^{(l)} &= \sigma \left( \hat{X}^{(l-1)}W^{(l)} \right) \quad \text{(FT)}
\end{align*}
\]

The notation letters are consistent with (1). The NA operation propagates and aggregates node features from the 1-hop neighbors of each node, whereas the FT operation transforms the aggregated features via linear transformation, followed by a non-linear ReLU activation function. After decoupling the NA and FT operations in the graph convolution layer, aggregation-transformation decoupled GCN has the flexibility to change the number of NA and FT layers without affecting each other.

Fig. 1 shows a two-layer GCN structure. By replacing $\hat{A}$ with an identity matrix, the GCN will degenerate to a Multi-layer Perceptron (MLP), which is equivalent to removing the NA operation from each layer of the GCN.

DAGNN [5] has shown that the key factor of degradation of the deep GCNs model is the entanglement of neighbor aggregation and feature transformation. The model decoupling NA and FT operations, making it possible to learn the representation of graph nodes from a larger receiver field.

APPNP [6] also decoupling NA and FT operations and using a two-layer fully connected neural network to extract graph feature information. In addition, APPNP uses the relationship between GCN and PageRank [23] to propose a propagation mechanism based on personalize PageRank, which preserves the information of local nodes while aggregating the information of more distant neighbors.

### III. ANALYSIS OF DEEP GCNS

In general, neural networks perform better with increasing model depth [24], [25]. GCN, on the other hand, perform worse as model depth increases. Existing research has mostly focused on the impact of neighbor aggregation on this problem, with little focus on the role of feature transformations, which are other important operators that make up a GCN layer. Few studies have considered the effects of both NA and FT in the GCN and optimized both operations simultaneously. For example, DAGNN [5] proposed an adaptive adjustment mechanism for NA operation, but only set them to a fixed number of layers for FT operations, whereas SGC [7] considered that the effect of GCN originating from NA and brutally removing FT operations.

In this study, we begin with extensive ablation experiments to investigate the effects of FT and NA on deep GCNs performance degradation. For this purpose, we separate the...
two operations and created two GCN variants: 1) NA-GCN, in which the number of layers of NA can increase separately, and 2) FT-GCN, in which the number of layers of FT can increase separately. The NA-GCN and FT-GCN of the L-layer are defined as shown in (4) and (5). The FT operation is denoted as $T(\cdot)$, $X$ is the input feature matrix, and $\hat{A}$ is the normalized adjacency matrix. $H^{(L)}$ is the node representation obtained after the L layers.

\[
H^{(L)} = T(\hat{A}^{-2}T(X)) \quad (\text{NA - GCN}) \tag{4}
\]

\[
H^{(L)} = \sum_{L=2}^{T(L)} (\hat{A}T(X)) \quad (\text{FT - GCN}) \tag{5}
\]

On three benchmark datasets, Cora, CiteSeer [26], and PubMed, we trained three models with varying depths, GCN, NA-GCN, and FT-GCN, and displayed the test accuracy, as shown in Fig. 2. The model layer range from one to eight. It can be observed that the FT-GCN suffers from even more severe performance degradation than NA-GCN. This is different from the previous viewpoint [14], [15] that the degradation of the deep GCNs performance can be attributed to over-smoothing caused by multiple neighbor aggregations. Instead, we find that stacking in more FT layers introduce a larger performance decline, and that FT contribute more than NA in the lower layers. When the number of model layers is small, although the NA operation has a certain impact on the performance degradation of the model, it is not the main reason for model degradation. Owing to the coupling of FT and NA operations in the GCN, the performance of the model also suffers a significant decline when stacking multilayer graph convolutional layers. We can observe that, after decoupling, the two operations show a certain degree of a trend of first increasing and then decreasing.

### A. INFLUENCE BY NEIGHBOR AGGREGATION

Neighbor aggregation (NA) in single-layer GCN obtains node embeddings by propagating and aggregating information from 1-hop neighbors, whereas stack multilayer GCN can perform multiple neighbor aggregations to obtain information about more distant neighbors to generate better node embeddings. However, too many neighbor aggregations can cause all node embeddings to converge to the same. We use quantitative analysis methods to study the influence of NA in deep GCNs. For the aggregation- transformation decoupled GCN, we set several different values of feature transformation layers $D_{FT}$ and varied the value of neighbor aggregation layers $D_{NA}$ from one to eight. The experimental results for the three benchmark datasets are shown in Fig. 3.

From Fig. 3, it can be observe that the $D_{NA}$ require to achieve the best accuracy on different datasets is different. Using $\frac{n}{\pi^2}$ to calculate the density of the edges, we know that the density of the edges is arranged from highest to lowest as Cora, CiteSeer, and PubMed. We found that the smaller the edge density, the sparser the graph structure, which often requires larger $D_{NA}$ to achieve better accuracy.

### B. INFLUENCE BY FEATURE TRANSFORMATION

Feature transformations (FT) are performed by linear transformations followed by non-linear activation functions on the embedding of nodes. As the number of feature transformation layers $D_{FT}$ increases, more complex data distributions and hidden information can be extracted. However, an excessively large $D_{FT}$ can lead to problems, such as overfitting and model degradation. Fig. 2 show that blindly increasing the $D_{FT}$ can cause a dramatic drop in model performance. It is natural to refer to residual connections [24] from CNNs to GNNs to solve the problem of model degradation caused by the increase in $D_{FT}$. Residual connectivity is a well-known concept whose primary purpose is to reduce gradient disappearance and gradient explosion during deep convolutional network training. Simultaneously, disruption of the symmetry of the neural network enhances its expression. In PubMed, we conduct comparative studies using the residual connectivity technique on the MLP and GCN.

From Fig. 4, we observe that when the number of model layers increases, the residual connection strategy appears to prevent performance degradation to some extent. The performance of the graph convolutional network, however, continues to deteriorate as the number of model layers increases. Subsequently, an experiment was conducted to determine an appropriate $D_{FT}$ value.

We use quantitative analysis methods to explain the impact of FT times on node classification performance through experiments on the Cora and ogbn-arxiv datasets. Based on the previous analysis, we fix a suitable neighbor aggregation number and analyse its effect on the deep GCN by varying the number of feature transformations. Specifically, we replace the MLP in the DAGNN with ResMLP and increase the value of feature transformations from 1 to 5. Fig. 5 shows that the node classification accuracy on ogbn-arxiv and Cora increase the number of feature transformations layers from 1 to 5.

From Fig. 5, we observe that when the number of feature transformations is equal to 2, the node features can be properly extract from the small graph Cora. However, for large graph data ogbn-arxiv, more feature transformation layers are required for feature extraction to generate high-quality node embeddings. We found that larger graphs tend to require larger $D_{FT}$ to achieve better accuracy.

Through the above analysis, we show that the FT operation and NA operation after GCN decoupling affect performance degradation to varying extents. The $D_{FT}$ and $D_{NA}$ values required for different graph datasets often differ. Therefore, we need to consider these two operations comprehensively instead of considering one of them separately.

### IV. DEEP DECOUPLED GRAPH NEURAL NETWORK

Motivated by the above experimental analysis, we propose a new decoupled GCN architecture to solve the problem of model degradation when the GCN becomes deeper. The effectiveness and robust network termed Adaptive Aggregation-Transformation Decoupled Graph Convolutional Network (AATD-GCN). Fig. 6 shows the feature
transformation part of our proposed AATD-GCN in detail, and Fig. 7 shows the workflow between feature transformation, neighbor aggregation and adaptive adjustment, which introduces the entire framework of AATD-GCN. Our AATD-GCN comprehensively considers the impact of FT and NA on deep GCNs. First, in the FT part, we use a residual connection technique to avoid model degradation when the number of layers increases, and an adaptive adjustment mechanism to balance the information from layers of different depths. Then, in the NA part, we employ initial residuals to preserve the initial information and an adaptive adjustment mechanism to balance the information from local and global.

Feature Transformation: Fig. 5 shows that larger graphs require more feature transformation layers to improve the model performance. To adapt the model to graph structural data of different sizes, we set the number of feature transformation layers $D_{FT}$ of the model as hyperparameters. Users can set different $D_{FT}$ based on the different graph sizes. In addition, Fig. 4 shows that both residual connections can alleviate the model degradation problem, thus helping to train a GNN with a large $D_{FT}$. In this study, we select a residual connection technique to train the large $D_{FT}$. We refer to our transformation layer in the following format:

$$X^{(l+1)} = \sigma(X^{(l)}W^{(l)}) + X^{(l)}, l = 1, 2, \ldots, L \in \mathbb{R}^{n \times c}$$

(6)

$$X = \text{stack}(X^{(0)}, X^{(1)}, \ldots, X^{(L)}) \in \mathbb{R}^{n \times (L+1) \times c}$$

(7)
FIGURE 6. The feature transformation part of the proposed Adaptive Aggregation-Transformation Decoupled Graph Convolutional Network (AATD-GCN). For clarity, we show the pipeline to generate the embedding for one node. The notation letters are consistent with (6)-(10), where lowercase bold indicates vectors. \( s \) is the projection vector that computes retainment scores for embeddings generating from feature transformation layers of different depths. \( s_0, s_1, s_2, \) and \( s_L \) represent the retainment scores of \( x_0, x_1, x_2, \) and \( x_L, \) respectively.

\[
S = \sigma(Xs) \in \mathbb{R}^{n \times (L+1) \times 1} \quad (8)
\]

\[
\tilde{S} = \text{reshape}(S) \in \mathbb{R}^{n \times 1 \times (L+1)} \quad (9)
\]

\[
X_{\text{out}} = \text{softmax}(\text{squeeze}(\tilde{S}X)) \in \mathbb{R}^{n \times c} \quad (10)
\]

Where \( W^{(l)} \) is the learning weight matrix shared within layers, \( X^{(l)} \) is the transformed feature matrix of the \( l \)-th layer of the MLP with residual connections. \( X \) stack with \( L + 1 \) feature matrices, which contains only the feature information of each node does not contain the graph structure information. Next, an adaptive adjustment mechanism is used to balance the results of the feature transformation output of each layer. Where \( s \in \mathbb{R}^{c \times 1} \) is a trainable projection vector, \( c \) is the number of node classes and \( L \) is a hyperparameter that indicates the number of feature transformation layers \( D_{FT}. \)

\( \sigma(\cdot) \) is an activation function, and a sigmoid function is applied. \( \tilde{S} \) are the retainment scores of \( X^{(0)}, X^{(1)}, \ldots, X^{(L)} \). Stack, reshape, and squeeze are utilized to rearrange the data dimension so that the dimensions can be matched during computation.

The model can extract more sophisticated feature information from the graph datasets and reduce the feature vector to a low dimension to enhance computational efficiency by using FT layers. However, it is difficult to determine the appropriate \( D_{FT} \) for feature transformation. Smaller graph data contain limited information, and effective information can be extracted using feature transformations with fewer layers; too many feature transformation layers can cause problems, such as gradient disappearance and gradient explosion, resulting...
in model degradation. Large amounts of graph data contain more feature information and complex data distributions, which require the use of more feature transformation layers for feature extraction. Fewer layers may not be sufficient to use this information fully and effectively. We use residual connections to avoid model degradation when the number of layers of a feature transformation is too large and introduce an adaptive adjustment mechanism to balance information from different depths.

Neighbor Aggregation: Fig. 3 shows that the number of neighboring aggregation layers required changes depending on the sparsity of the graph structure, with sparser graphs requiring more neighbor aggregation for message transmission. We set the number of neighbor aggregation layers \( D_{NA} \) of the model as hyperparameters. Too many neighbor aggregations, on the other hand, might produce over-smoothing problems owing to the loss of initial information, resulting in performance degradation. We use the initial residual connection to ensure that part of the nodes’ initial feature information was preserved after multiple layers of neighbor aggregation. Furthermore, we employ an adaptive adjustment mechanism that automatically balances information from local and global. We refer to our neighbor aggregation layer in the following format:

\[
\begin{align*}
H^{(l)} &= \hat{A}^l X_{out} + \alpha X_{out}, l = 1, 2, ..., k \in \mathbb{R}^{n \times c} \\
H &= \text{stack}(X_{out}, H^{(1)}, \cdots, H^{(k)}) \in \mathbb{R}^{n \times (k+1) \times c} \\
S &= \sigma(\text{Hs}) \in \mathbb{R}^{n \times (k+1) \times 1} \\
\hat{S} &= \text{reshape}(S) \in \mathbb{R}^{n \times 1 \times (k+1)} \\
H_{out} &= \text{softmax}(\text{squeeze}(\hat{S}H)) \in \mathbb{R}^{n \times c}
\end{align*}
\]

where \( X_{out} \) is obtained using MLP with residual connections and adaptive adjustment mechanisms in the feature transformation section. \( \alpha = \ln \left( \frac{\lambda}{s} \right) \) is a parameter controlling the initial residuals, \( \lambda \) is a hyperparameter that controls the amount of initial information is contained after each layer of neighbor aggregation. After the feature transformation, we utilize the symmetrical normalization propagation mechanism \( \hat{A} = D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}} \), where \( \hat{A} = A + I \). \( k \) is a hyperparameter that indicates the number of layers of neighbor aggregations \( D_{NA} \), can be set to a different number of layers, depending on the sparsity of the graph structure data. \( H^{(l)} \) is the node representation matrix obtained by neighbor aggregation and initial residuals, \( H \) stacked with \( k + 1 \) node representation matrices, which contains feature information in \( X_{out} \) and the structure information of the graph in \( \hat{A} \). \( s \) is a trainable projection vector used to calculate the retention scores \( S \). \( \hat{S} \) are the retaintion scores of \( X_{out}, H^{(1)}, \cdots, H^{(k)} \). As the \( D_{NA} \) increases, additional initial information is lost. We gradually increased \( \alpha \) as \( D_{NA} \) increased, implying a gradual increase in the proportion of initial information contained in each layer. Obviously, the \( X_{out} \) via the feature transformation layer contains only the feature information of the individual nodes and not the structural information of the graph. Using the information propagation mechanism and residual connectivity techniques for the feature transformed \( X_{out} \), the output \( H^{(l)} \) represents the node representations obtained from the \( l \)-hop away. As \( l \) increases and the receptive field grows, \( H^{(l)} \) will contain more global information. However, it is difficult to determine a suitable \( D_{NA} \) for balancing the information from local and global. A small \( D_{NA} \) may fail to capture sufficient and essential neighborhood information, whereas a large \( D_{NA} \) may bring too much global information and loss the special local information. Furthermore, each node in the graph has a different number of edges; that is, the nodes require different receptive fields to aggregate information about their neighbors. In addition, each node in the graph has a varied number of edges, requiring various receptive fields to collect information about its neighbors. Consequently, we introduce an adaptive adjustment mechanism and use the initial residuals to avoid the loss of initial information after numerous aggregations of information. To compute the retention scores, we add trainable projection vectors shared by all nodes. These scores are used to create representations that included information from various neighborhoods. These retention scores indicate how much information from relevant representations obtained from various propagation numbers should be retain in the final representation of each node. The AATD-GCN can balance the information from local and global neighborhoods for each node using this adaptive adjustment mechanism.

AATD-GCN using aggregation-transformation decoupled GCN structures can increase the \( D_{FT} \) and \( D_{NA} \) without affecting each other. The use of residual connectivity in the feature transformation section prevents model degradation as the model becomes deeper, and the use of an adaptive adjustment mechanism automatically balances the extracted feature information from different depths. The use of initial residuals in the neighbor aggregation section reduces over-smoothing by retaining some of the initial information after multiple aggregations and automatically balances the information from local and global using an adaptive adjustment mechanism. Setting \( D_{FT} \) and \( D_{NA} \), residual connections, and adaptive adjustment mechanisms enable the AATD-GCN to extract complex feature information from deep feature transformations and aggregate more valid information from large receptive fields, generating a suitable node representation for each special node. Furthermore, after decoupling the GCN, larger receptive fields can be obtained without introducing trainable parameters, other than the projection vector. In addition, using a feature transformation layer to transform the input feature representation into a low-dimensional vector can improve the computational efficiency.

V. EXPERIMENTS
In this section, we evaluate the performance of AATD-GCN against state-of-the-art graph neural network models on a wide variety of real-world graph datasets. In addition, we compared the robustness of the aggregation-transformation decoupled GCN with its counterpart.
**A. DATASET AND SETUP**

We employ three standard citation network datasets for semi-supervised node classification: Cora [26], CiteSeer [26], and PubMed [26]. Nodes correspond to documents and edges to citations in these citation datasets. Each node feature corresponds to the document’s bag-of-words representation and belongs to one of the academic themes. Amazon Computers and Amazon Photo are two segments of the Amazon co-purchase graph [27], in which nodes represent goods, edges show that two goods are commonly purchased together, node characteristics are bag-of-words encoded product reviews, and class labels are determined by the product category. Co-authorship graphs such as Co-author CS and Co-author Physics are co-authorship graphs [28]. Nodes represent authors, and edges connect them if they co-authored a publication. Node features contain paper keywords for each author’s articles and class labels indicate each author’s most active field of study. The statistics for these datasets are listed in Table 1.

**Table 1. Dataset Statistics**

| Dataset   | Classes | Nodes | Edges | Features | Edge Density |
|-----------|---------|-------|-------|----------|--------------|
| Cora      | 7       | 2708  | 5429  | 1433     | 0.074%       |
| CiteSeer  | 6       | 3312  | 4732  | 3703     | 0.043%       |
| PubMed    | 5       | 19717 | 44338 |          |              |
| C-CS      | 15      | 18333 | 81894 | 6805     | 0.024%       |
| CS-Physics| 5       | 34493 | 247962| 8415     | 0.021%       |
| A-CS       | 10      | 13381 | 245778| 767      | 0.0014%      |
| A-Photo    | 8       | 7487  | 119043| 745      | 0.2124%      |

Edge density is computed using $\frac{m}{n^2}$. We use the Adam SGD optimizer [29] to train the AATD-GCN with a learning rate of 0.01, run the methods for 1000 epochs, and stop early with 100 epochs. We tune the hyperparameters for models: For the PubMed data, where the edge density is relatively small, we use 20 times neighbor aggregation, for the Cora and CiteSeer datasets, where the edge density is slightly larger, we use 10 times neighbor aggregation, and for the cs dataset, where the edge density is much larger, we use only 5 times neighbor aggregation. The learning rate is set to 0.1, weight is set to 0.005, and the dropout rate is obtained from a search of range 0.1 to 0.5 with step 0.1. The initial residual ratio is obtained from a search of range 200 to 1000 with step 200, the number of feature transformation layers, $D_{FT}$, is set to 2 for Cora, CiteSeer, and PubMed. The other model hyperparameters follow the settings described in the original study. We run all methods 10 times and report the mean values and variances.

The following baselines are considered: Multilayer Perceptron (MLP), Graph Convolutional Network (GCN) [2], Graph Attention Network (GAT) [20], GraphSAGE [30], APPNP [6], SGC [7], JKNet [18], AP-GCN [8], SIGN [31], GBP [32], and DAGNN [5]. We use the same dataset splits and training techniques to provide a thorough and fair comparison between the various models for each dataset.

**B. OVERVIEW OF RESULTS**

We summarize the classification accuracy of the ten models on citation networks, giving their mean values and variances in Table 2. To ensure that the results were true and valid, we use 20 labeled nodes per class as the training set, 500 nodes as the validation set, and 1000 nodes as the test set for our models, and run 100 times for the fixed training/validation/test split from [5], which is commonly used to evaluate performance by the community.

**Table 2. Summary of classification accuracy (%) results**

| Method    | Cora     | CiteSeer | PubMed  |
|-----------|----------|----------|---------|
| MLP       | 61.6±0.6 | 61.0±0.0 | 74.2±0.5|
| GCN       | 81.8±0.5 | 70.8±0.5 | 79.3±0.5|
| APPNP     | 83.3±0.5 | 71.8±0.5 | 80.1±0.5|
| JKNet     | 81.8±0.5 | 70.8±0.5 | 79.3±0.5|
| SGC       | 81.0±0.2 | 71.3±0.5 | 79.5±0.5|
| AP-GCN    | 83.3±0.5 | 71.3±0.5 | 79.5±0.5|
| SIGN      | 82.1±0.5 | 72.4±0.8 | 79.5±0.5|
| GBP       | 83.9±0.7 | 75.2±0.5 | 80.6±0.4|
| DAGNN     | 84.4±0.5 | 73.3±0.0 | 80.5±0.0|
| AATD-GCN  | 84.8±0.4 | 73.7±0.3 | 80.9±0.3|

Table 2: Summary of classification accuracy (%) results on citation network datasets: Cora, CiteSeer, and PubMed. As shown in Table 2, our proposed AATD-GCN exhibits a significant improvement over most representative baseline models and achieves a state-of-the-art performance. Specifically, compare with the initial GCN, our proposed AATD-GCN improved by 3.0%, 2.9%, and 1.6% in Cora, CiteSeer, and PubMed, respectively. For the other variants of the GCN model, we obtain large enhancements. This is because we not only use aggregation- transformation decoupled GCN structures that enable for multiple feature transformations and neighbor aggregation but also add residual connectivity, initial residuals, and adaptive adjustment mechanisms.

Table 3 presents the results for the co-authorship and co-purchase datasets. We use a training set of 20 labeled nodes per class, validation set of 30 nodes per class, and test set of the remaining nodes. The baseline results are obtained from [30]. To ensure a fair comparison with the baselines, we run 100 runs for random training/validation/test splits in the AATD-GCN. Table 3 shows that our proposed AATD-GCN model outperforms all the listed baseline models and achieves better performance than the GCN models by significant margins of 1.8%, 1.4%, 2.2%, and 0.4% on the Coauthor CS, Coauthor Physics, Amazon Computers, and Amazon Photo, respectively. It is worth noting that our proposed AATD-GCN model not only exhibits the best performance, but also the least bias.

**C. MODEL ROBUSTNESS**

To investigate the robustness of our AATD-GCN in the face of different depth, we perform experiments to validate the performance of our proposed AATD-GCN model for a progressively increasing number of feature transformation layers $D_{FT}$ and neighbor aggregation layers $D_{NA}$ in Cora,
CiteSeer, and PubMed. First, we fix \( D_{FT} \) and vary \( D_{NA} \) from 1 to 200. Then, we fix \( D_{NA} \) and vary \( D_{FT} \) from 1 to 8. The results of the two groups of experiments are present in Fig. 8. We can observe that AATD-GCN can maintain stable performance when \( D_{NA} \) increases, benefiting from the adaptive adjustment mechanism and initial residuals used at the NA layer. A large receiver field can be obtained, balancing information from local and global sources, and avoid the loss of original information when \( D_{NA} \) increases. However, the AATD-GCN also maintains stable performance during \( D_{FT} \) growth, which is attributed to the adaptive adjustment mechanism and residual connectivity used in the FT part. This enables AATD-GCN to extract more complex feature information, balance information from different depths, and avoid model degradation.

To further validate the robustness of our AATD-GCN in the face of different model depth, we chose SGC and S\(^2\)GC as baseline methods for comparison.

First, we fix the \( D_{FT} \) to 3 and increase the \( D_{NA} \) from 2 to 20 on the ogbn-arxiv dataset. As shown in Fig. 9, the SGC test accuracy decrease rapidly when the \( c \) increase. S\(^2\)GC and AATD-GCN can still maintain high accuracy when the number of neighbor aggregations increases, but our proposed AATD-GCN performs better. The robustness of the AATD-GCN to the \( D_{NA} \) is owing to the use of an initial residual connection technique and an adaptive adjustment mechanism that retains some of the initial information after multiple neighbor aggregations and adaptively balances the information from local and global.

Second, we fix the \( D_{NA} \) to 10 and increase the \( D_{FT} \) from 1 to 10. Fig. 9 shows that SGC also shows heavy performance degradation as the number of feature transformations increases, with S\(^2\)GC showing less degradation and a more moderate trend than SGC, whereas AATD-GCN shows less degradation and more robustness as the \( D_{FT} \) increases. This robustness is owing to the addition of residual connectivity and adaptive adjustment mechanisms to the feature transformation layer, which effectively avoids the degradation of the model when the \( D_{FT} \) increases. Overall, our proposed AATD-GCN considers the effects of FT and NA on the deep GCNs. Therefore, it is more robust than other baseline models in the face of larger \( D_{FT} \) and \( D_{NA} \).

**D. ANALYSIS OF MODEL SUPERIORITY**

Our proposed AATD-GCN has two depths \( D_{FT} \) and \( D_{NA} \). For the two different parts, we use techniques such as adaptive adjustment mechanisms, residual connectivity, and initial residuals for learning node representations.

First, in the feature transformation part, AATD-GCN differs from DAGNN strategy of fixing the number of feature transformation layers by adjusting number of feature transformation layers and using residual connectivity and adaptive adjustment mechanisms to obtain better feature embeddings. To demonstrate the effect of adjustability \( D_{FT} \) in AATD-GCN, we change the neighbor aggregation part of AATD-GCN to the same structure as DAGNN and use the same number of layers. Subsequently change the number of feature transformation layers \( D_{FT} \) to compare the experimental results. The experimental results are shown in Fig. 10. As the number of feature transformation layers increase, DAGNN quickly shows a decrease in performance, whereas AATD-GCN maintained more stable results and tend to have better results than DAGNN. This is because the adjustability \( D_{FT} \) with residual connectivity and adaptive adjustment mechanism can learn feature information from different model depths while retaining the initial feature information. This enables AATD-GCN to learn more effective embeddings.
from the input feature vectors, which in turn improves the final prediction results.

Second, in the neighbor aggregation part, AATD-GCN adopts an adaptive adjustment mechanism, initial residuals, and designs the parameter $\alpha$ to control the percentage of initial residuals to obtain a large adaptive receiving field to obtain the neighbor information. $\alpha$ increases as $D_{NA}$ increases to avoid losing initial information. APPNP, which also expands the receiving field, proposes an improved version of Personalized Propagation of Neural Prediction (PPNP) based on Personalized Page Rank compared to the traditional GCN. The feature transformation part and the neighbor aggregation part are decoupled to provide a larger receiving field for each node whereas only one parameter, $\alpha$, is introduced. AP-GCN is also a model that decouples feature transformation and neighbor aggregation, and proposes a method to select the number of neighbor aggregations per node adaptively by weighing the computation time and accuracy. AATD-GCN, compared to APPNP and AP-GCN, uses an adaptive adjustment mechanism and initial residuals while increasing the receiving field, which can achieve dynamic adjustment of local and global neighbor information, and preserve the initial information of nodes when $D_{NA}$ becomes large.

VI. CONCLUSION

When the GCN model is trained to a deeper level, the performance decreases. In this study, we investigated the impact of FT and NA operations on the model degradation, and identified methods to avoid model degradation in deep GCNs. Furthermore, the experimental results show that different graph datasets have different requirements of the model depths $D_{FT}$ and $D_{NA}$. We proposed an Adaptive Aggregation-Transformation Decoupled Graph Convolutional Network (AATD-GCN), a flexible and robust network that can simultaneously obtain large receiver fields and extract complex features information. Extensive experiments on real-world graph datasets show the superiority of AATD-GCN in terms of accuracy and robustness.

REFERENCES

[1] Q. Li, Z. Han, and X.-M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in Thirty-Second AAAI conference on artificial intelligence, 2018.
[2] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” arXiv preprint arXiv:1609.02907, 2016.
[3] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, “Simple and deep graph convolutional networks,” in International Conference on Machine Learning. PMLR, 2020, pp. 1725–1735.
[4] W. Zhang, Z. Sheng, Y. Jiang, Y. Xia, J. Gao, Z. Yang, and B. Cui, “Evaluating deep graph neural networks,” arXiv preprint arXiv:2108.09055, 2021.
[5] M. Liu, H. Gao, and S. Ji, “Towards deeper graph neural networks,” in Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 338–348.
[6] J. Klicpera, A. Bojchevski, and S. Günnemann, “Predict then propagate: Graph neural networks meet personalized pagerank,” arXiv preprint arXiv:1810.05997, 2018.
[7] F. Wu, A. Souza, T. Zhang, C. Fity, T. Yu, and K. Weinberger, “Simplifying graph convolutional networks,” in International conference on machine learning. PMLR, 2019, pp. 6861–6871.
[8] I. Spinelli, S. Scardapane, and A. Uncini, “Adaptive propagation graph convolutional network,” IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 10, pp. 4755–4760, 2020.
[9] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and E. G. Dahl, “Neural message passing for quantum chemistry,” in International conference on machine learning. PMLR, 2017, pp. 1263–1272.
[10] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in Icml. 2010.
[11] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, “Measuring and relieving the over-smoothing problem for graph neural networks from the topological view,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, 2020, pp. 3438–3445.
[12] G. Li, C. Xiong, A. Thabet, and B. Ghanem, “Deepergcnn: All you need to train deeper gcns,” arXiv preprint arXiv:2006.07739, 2020.
[13] G. Li, M. Muller, A. Thabet, and B. Ghanem, “Deeppcns: Can gcns go as deep as cnns?” in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 9267–9276.
[14] Y. Rong, W. Huang, T. Xu, and J. Huang, “Dropedge: Towards deep graph convolutional networks on node classification,” arXiv preprint arXiv:1907.10903, 2019.
[15] L. Zhao and L. Songul, “Pairnorm: Tackling oversmoothing in gcns,” arXiv preprint arXiv:1909.12223, 2019.
[16] G. Taudin, “A signal processing approach to fair surface design,” in Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, 1995, pp. 351–358.
[17] K. Zhou, Y. Dong, K. Wang, W. S. Lee, B. Hooi, H. Xu, and J. Feng, “Understanding and resolving performance degradation in deep graph convolutional networks,” in Proceedings of the 39th ACM International Conference on Information & Knowledge Management, 2021, pp. 2728–2737.
[18] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,”
in *International Conference on Machine Learning*, PMLR, 2018, pp. 5453–5462.

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[21] Y. Xie, S. Li, C. Yang, R. C. W. Wong, and J. Han, “When do gnn work: Understanding and improving neighborhood aggregation,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2020.

[22] C. Zhuang and Q. Ma, “Dual graph convolutional networks for graph-based semi-supervised classification,” in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 499–508.

[23] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web,” Stanford InfoLab, Tech. Rep., 1999.

[24] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] M. Telgarsky, “Benefits of depth in neural networks,” in *Conference on learning theory*. PMLR, 2016, pp. 1517–1539.

[26] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, “Collective classification in network data,” *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[27] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, “Image-based recommendations on styles and substitutes,” in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 43–52.

[28] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, “Pitfalls of graph neural network evaluation,” *arXiv preprint arXiv:1811.05868*, 2018.

[29] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[30] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Advances in neural information processing systems*, vol. 30, 2017.

[31] E. Rossi, F. Frasca, B. Chamberlain, D. Eynard, M. Bronstein, and F. Monti, “Sign: Scalable inception graph neural networks,” *arXiv preprint arXiv:2004.11198*, 2020.

[32] M. Chen, Z. Wei, B. Ding, Y. Li, Y. Yuan, X. Du, and J.-R. Wen, “Scalable graph neural networks via bidirectional propagation,” *Advances in neural information processing systems*, vol. 33, pp. 14556–14566, 2020.

---

**DEZHI SUN**
Doctor. He is currently a lecturer of School of Information Engineering, Beijing Institute of Graphic Communication.

His main research interests include social network analysis, data mining, and network security, and so on.

**MAN HU**
was born in 1997 in Anyang, Henan Province, China, and received his Bachelor’s degree from Zhengzhou University of Light Industry in 2020. He is currently studying for a master’s degree at the Beijing Institute of Graphic Communication. His research areas include data mining, graph representation learning, and social networks, which work on solving graph data problems using deep learning methods.

**ZHENYU LI**
is currently pursuing a bachelor’s degree, majoring in computer science, at the School of Information Engineering, Beijing Institute of Graphic Communication, whose primary research fields are social networks, data analysis, and community detection. He has participated in many critical projects of the Beijing Natural Science Foundation, scientific research projects of the Beijing Municipal Education Commission, and vital school-level projects.

---

***