ARE GRAPH ATTENTION NETWORKS ATTENTIVE ENOUGH? RETHINKING GRAPH ATTENTION BY CAPTURING HOMOPHILY AND HETEROPHILY

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention Mechanism has been successfully applied in Graph Neural Networks (GNNs). However, as messages propagate along the edges, the node embeddings for edge-connected nodes will become closer even though we can not ensure these nodes have similar features and labels, especially in heterophily graphs. The current attention mechanisms cannot adaptively extract information from the neighbors because they can not fully use the graph information in self-attention calculation. We introduce new a graph attention mechanism (GATv3) straightly involving the graphic information in the self-attention calculation, which can be aware of the homophily or heterophily of the graphs. We conduct an extensive evaluation in node classification tasks and show that using graphic information and features simultaneously can extract more diverse attention scores. Our code is available at https://github.com/anonymousSubscriber/G-GAT

1 INTRODUCTION

Graph Neural Networks (GNNs) have become increasingly popular since many real-world relational data can be represented as graphs, such as social networks (Lee et al., 2009), molecules (Gilmer et al., 2017) and financial data (Yang et al., 2021; Zhang et al., 2022). As long as there are abstractable connections between entities, we can model these entities and their relationships with nodes and edges. In general GNNs, each node updates its representation and generates a more widely applicable embedding for the downstream tasks by analyzing the message sent from their neighbors. Graphic information plays a major role in solving the problem of overfitting, compared with directly mapping the nodes' features with the linear transformation.

To improve the representation ability of GNNs, several GNN variants have been proposed but vary in the way each node aggregates and combines the representations of its neighbors (e.g., chebyNet (Defferrard et al., 2016), GAT (Velickovic et al., 2017), JK-net (Xu et al., 2018)), or for the nodes to be selected as "neighbors" (e.g., GraphSAGE (Hamilton et al., 2017), GeomGCN (Pei et al., 2020)). Specifically, in GATs, each node selectively extracts information from its neighbors by the attention mechanism, which generalizes the standard neighbor aggregation methods (e.g., averaging or max-pooling). In the traditional GAT model, node features are projected to the low-dimension representations by a linear transformation. Analyzing node computes a weighted average of its neighbors through a linear transformation from the combination of their low-dimension representations. The work also generalizes the Transformers (Dwivedi & Bresson, 2020) that applies the self-attention mechanism to graph neural networks, and GATv2 (Brody et al., 2021) which modify the order of internal operations and achieve a dynamic attention mechanism.

Graph architecture mainly depends on the purpose and pattern of graphic construction. Some graphs are "Homophily Graphs" in which edge-connected nodes have more substantial feature similarities and have the same labels. Others are "Heterophily Graph" with a large number of heterophily edges. We will selectively mask or replace the edges in the real-world graph and prove that the heterophily edges will hinder the model from extracting information from the graphs. The current GATs calculate the attention based on the node's features. However, in the message propagation, the adjacent nodes on the graph will generate similar embedding, which makes the attention scores lack differentiation, and the attention mechanism cannot adaptively select information from different neighbors.

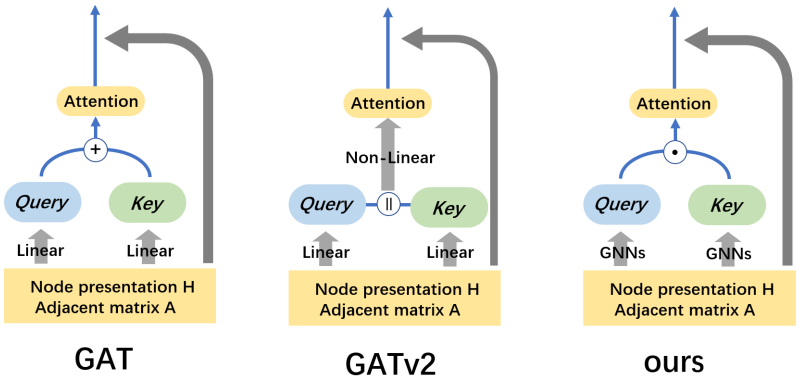

Figure 1: Overview of 3 GATs, $\oplus$ means addition operation, $\odot$ means dot product

,

Furthermore, if there is a higher weight to nodes of different labels, the embedding of these two nodes will become closer with message transmission, making it more difficult to distinguish such nodes in downstream tasks. This problem will become more serious in heterophily graphs as there are more heterophily edges.

To overcome the limitation, we introduce a new formula for GAT by generating query and key with GNNs model (e.g., GCN or its variant) to import graphic information in attention calculation. In addition, we replace the addition operation with dot product in general GAT to implement dynamic attention. We conducted many searches on the schemes used to calculate queries and keys and found that queries and keys generated by GNNs are more distinguishable than ordinary linear layers, and the model will perform better in heterophily graphs.

In summary, our main contributions are: Firstly, we mask or replace the edges in real-world data and confirm that the heterophily edges will partly affect the performance of the GNN models. Secondly, We introduce a new attention architecture that calculates query and key with other GNN models in the attention layer and evaluates the model in node classification real-world data. Finally, we thoroughly searched the GNN model used in generating queries and keys and compared the model performance with the traditional linear transformation.

2 RELATED WORK

GNNs have emerged as an indispensable tool to learn graph-centric data. A graph neural network layer updates every node representation by aggregating its neighbors' representations. This process can also be considered as a voting process where each node in a certain subgraph works together and generates embedding vectors for the whole subgraph. With the development of GNN, researchers try to improve GNN from different perspectives.

2.1 IMPROVEMENT IN WHO IS "NEIGHBORS"

In general GNNs, each node adopts the edge-connected nodes as neighbors and exchanges information with them. A GNN architecture piles the convolution layer to expand the reception field. However, as the graph and downstream task become more complex, the traditional method will not work in many of the datasets. Researchers also make the improvement in which nodes would be selected as neighbors and exchange information with the central node.

Sampling As the graphs grow larger and become more complicated, it would be hard or impossible to get the complete graph in the training process. Hamilton define these tasks as inductive downstream tasks and sample the ordinary neighbors (edge connected node) to overcome the weakness of traditional GNN models (GraphSAGE (Hamilton et al., 2017)) and this work was generalized by other kinds of sampling strategies (e.g., FastGNN (Chen et al., 2018), ASGCN (Li et al., 2019)).

More hops In order to better analyze the information of the whole graph, GNNs began to grow deeper, and more hops neighbors[1] begin to exchange messages with the central node. However, the over-smoothing problem becomes more serious as the reception field increases. To overcome the limitation, Lius introduce to weighted average the information of different hops with learnable parameters (Liu et al., 2020) so the model can self-adaptively select the information in different graphs and adjust the importance of different hop neighbors. The skip connection mechanism which combines the output of different layers of GNNs (e.g., JK-net (Xu et al., 2018)) is widely used recently. And different strategies are proposed to coordinate information from different hops (e.g JACOBICONV (Wang & Zhang, 2022)). Now, Jk-net and its variant have become one of the most popular GNN structures and achieve SOTA performance in many of the datasets.

2.2 Improvement in How to aggregate

The design of the AGGREGATE is what mostly distinguishes one type of GNN from the other. Earlier GNN frameworks utilized a fixed propagation scheme along all edges, and the work was generalized by a degree-related convolution kernel which tends to receive more information from nodes with lower degrees (Kipf & Welling, 2016). Moreover, a polynomial convolution kernel is proposed to provide a more flexible convolution (e.g., ChebyGCN (Defferrard et al., 2016)).

On the other hand, as the graphs grow larger, it takes more time to sequentially calculate the convolution kernel of multiple layers. Some GNNs discard the nonlinear transformation in the convolution layer and use the convolution results that are calculated in advance as input to speed up the training and reduce the memory allocation (e.g., SGC (Wu et al., 2019)).

In this paper, Our main work focuses on "How to aggregating". And we combine the attention layer with the skip connection mechanism to improve the ability of the model.

3 PRELIMINARIES

3.1 Node classification

Node classification is a graph-based semi-supervised learning problem. It has been one of the most popular downstream tasks to measure the GNNs' ability. Each node has a label as the attribute in the node classification task. The model establishes an end-to-end transformation to predict the nodes' labels.

The node classification task encompasses training the GNN to generate an embedding for each node with its original feature and graphic information. Moreover, predicting the nodes' labels with a classifier (e.g., linear classifier). The gradients of loss are back-propagated through the GNN layers. Once trained, the model can be used for the prediction of labels of nodes in the test set.

3.2 Graph and Graph Neural Networks

A directed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ contains nodes $\mathbb{V} = \{1, ..., n\}$ and $\mathbb{E} \subset \mathbb{V} \times \mathbb{V}$, where $(i, j) \in \mathbb{E}$ denotes an edge from node i to j, each node has an initial representation $h_i^0 \in \mathbb{R}^{d_0}$.

In Graph Neural Networks, a layer's input is a set of node representations$\{h_i \in \mathbb{R}^d | i \in \mathcal{V}\}$ and the set of edges. A layer's output is a set of node representations $\{h_i^{'} \in \mathbb{R}^{d^{'}} | i \in \mathcal{V}\}$, where the same parametric function is applied to every node. A GNN layer can be represented as:

$$h_i^{'} = f_\theta(h_i, AGGREGATE(h_j | j \in \mathcal{N}_i)) \tag{1}$$

where the N means the neighbors of node i, $AGGREGATE(h_j | j \in \mathcal{N}_i)$ is the function used to synthesize the information of neighbors (e.g., max-pooling, sum, average).

3.3 Homophily and Heterophily of the graph

We use the β to measure the homophily or heterophily of the graphs proposed in (Pei et al., 2020).

[1] x-hop neighbor means there is an x-length route between the nodes

$$\beta = \frac{Number\ of\ v's\ neighbors\ who\ have\ the\ same\ label}{Number\ of\ v's\ neighbors} \tag{2}$$

The graph that has a higher β implies that the edges in that graph tend to connect nodes with different labels and vice versa. In the node classification task, adopting too much information for the nodes with different labels results in noise being introduced into the model.

3.4 GAT

To make the self-adaptive selection among the neighbor message, GAT introduces an attention framework by computing a learned weighted average for the representation of node neighbors. A scoring function $e : \mathbb{R}^{d_0} \times \mathbb{R}^{d_0} \to \mathbb{R}^1$ computes a score for every edge$(i, j) \in \mathbb{E}$, which indicates the importance of the features of the neighbor i to the node j:

$$e(h_i, h_j) = LeakyReLU(a^T \cdot [Wh_i||Wh_j]) \tag{3}$$

where $a \in \mathbb{R}^{2d}$, $\mathcal{W} \in \mathbb{R}^{d_0} \times \mathbb{R}^{d_0}$ are learned, and $||$ denotes vector concatenation. These attention scores are normalized across all neighbors $j \in N_i$ using softmax function by:

$$\alpha_{(i,j)} = softmax(e(h_i, h_j)) = \frac{exp(e(h_i, h_j))}{\sum_{k \in \mathcal{N}_i} exp(h_i, h_k)} \tag{4}$$

Then, GAT computes a weighted average for the transformed features of the neighbors (followed by a nonlinearity σ as the new representation of i, using the normalized attention coefficients:

$$h_i^{=} \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot Wh_i) \tag{5}$$

(3) mathematically equals to

$$e(h_i, h_j) = LeakyReLU(a_1^T \cdot [Wh_i] + a_2^T \cdot [Wh_j]) \tag{6}$$

where $a_1, a_2 \in \mathbb{R}^{d_0}$, and $a_1||a_2 = a$. With the "Associative Property of Matrix Multiplication", $a_1^T W$ and $a_2^T W$ can be regarded as two independent linear transformations

$$e(h_i, h_j) = LeakyReLU(W_1 h_i + W_2 h_j) \tag{7}$$

This indicates that the calculated weights are equal to the sum of one-dimensional projections for nodes connected by edges. This problem also has been defined as statistic attention denoting the ranking (the argsort) of attention coefficients is identical for all nodes in the graph and is unconditioned on the query nodes (Brody et al., 2021).

In this paper, we focus on the detail that since messages are continually propagating along the edges of the network, the one-dimensional projections of the key nodes are very close, which implies the $W_2 h_j (j \in \mathbb{N}_i)$ have a substantial similarity because they are frequently affected by the same gradient. Suppose we consider the process of each node making predictions as a voting activity. The central node i and the neighbors N_i have the right to express their opinions about the label of node i, and then they will be told the **same** correct answer. The graphically adjacent nodes will have more possibility to directly or indirectly participate in the duplicate voting ("indirectly" means they participate in the same vote with the same nodes). That will lead to graphically adjacent nodes will have the same opinion because they are informed with the same answers even though they may not share the same labels.

4 GATV3

To avoid the embeddings of edge-connected but label-different nodes becoming over similar, it is necessary to select the information of different neighbors adaptively. If we only use linear transformation to generate queries and keys from the original features, the quality of the attention scores will

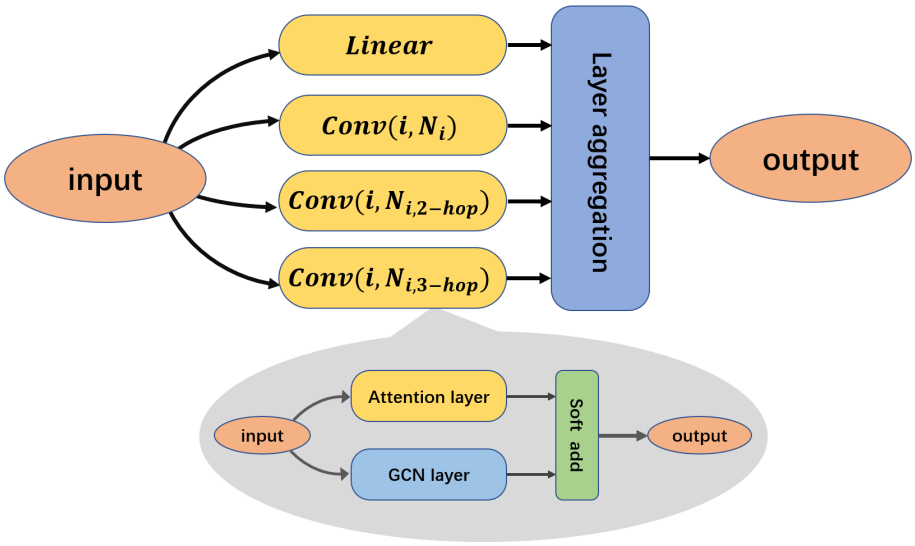

Figure 2: Skip connection in GATs

,

be limited. So we propose a new attention mechanism that uses GNN models to generate queries and keys to improve the quality of attention.

$$e(h_i, h_j) = GNN_Q(Wh_i, A) \cdot GNN_k(Wh_j, A)^T \tag{8}$$

$$\alpha = softmax(e(h_i, h_j)) \tag{9}$$

where $GNN_Q, GNN_K \in \{\langle R^d, G\rangle \rightarrow \langle R^{d'}, G\rangle\}$ are learned GNNs such as GCN , and $\mathcal{W} \in \mathbb{R}^{d_0} \times \mathbb{R}^{d_0}$

Compared with linear transformation, GNN models allow a more comprehensive range of nodes to participate in the calculation of the attention scores. Moreover, the graphic information in Q and K can help the attention layer learn more graphic information which can help the nodes distinguish from others. As the neighbors are involved in generating queries and keys through GNN models, the nodes with shared neighbors will be more likely to have a higher attention score because they adopt information from the same node. Simultaneously, the nodes with shared neighbors are more likely to have a closer graphic connection.

5 SKIP CONNECTION IN GATS

The 'skip connections framework has been widely applied in GNN architecture to overcome the over-smoothness in the deep layer GNNs and has been considered to be one of the most popular constructions for multi-layer GNN models (Xu et al., 2018). By combining the output of each GNN layer, the model can self-adaptive adopt information from the different hop of neighbors.

However, as the network layer deepens and each node accepts more information in the message propagation process. If we still construct a serial network, in deeper attention layers, we have to use repeatedly convoluted information to calculate the attention score which hinders the attention scores from becoming attentive. So we perform a parallel structure. The processes in which each node receives messages from different hop neighbors are set in parallel channels. Hence the attention layer can generate queries and keys through shallow networks. For example, if we propose a singer layer GCN to generate the Q and K, no matter which hop of neighbors we collect information from, the query and key are only generated by one layer GCN (convoluted once):

$$Q, K = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}XW \tag{10}$$

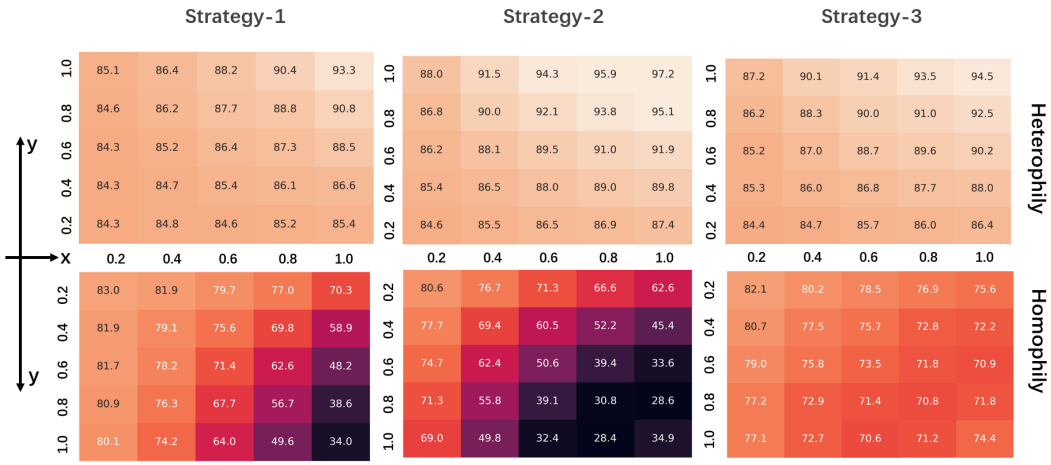

Figure 3: Mask or replace x nodes' y edges, and get the prediction accuracy with GCN model

,

where the D is degree matrix and $A \in \{0,1\}^{n\times n}$is adjacent matrix, $W \in \{\mathbb{R}^{d\times d^{'}}\}$. and $X \in R^{d}$ is original features.

6 HETEROPHILY EDGES LEAD TO NOISE?

In this section, we will prove that the heterophily edges will have a harmful effect on the model performance. We introduce three strategies to change the proportion of heterophily edges in the real-world graph and evaluate GNN models' performance (accuracy on node classification task).

strategy-1: For $x\%$ nodes, we mask $y\%$ of their heterophily edges or homophily edges. x, y are hyperparameters and $0 \leq x, y \leq 100$.

strategy-2: For $x\%$ nodes, we replace $y\%$ of their heterophily edges with non-existing homophily edges. Or replace $y\%$ of their homophily edges with non-existing heterophily edges. x, y are hyperparameters and $0 \leq x, y \leq 100$. Compared with **strategies-1**, this strategy will keep the graph edges density and exclude the influence of the change of graph structure on the experimental results.

strategy-3: For $x\%$ nodes, we replace $y\%$ of their heterophily edges with 2-hop homophily edges. Or replace $y\%$ of their homophily edges with 2-hop heterophily edges. x, y are hyperparameters and $0 \leq x, y \leq 100$. 2-hop edges represent the edges connecting a 2hop neighbor and the center node. Compared with replacing the existing edges with random non-existing edges, 2hop neighbors have a stronger graphic relationship with the central node. This strategy can further exclude the influence of the change of graphic structure on the experimental results.

Table 1: Datasets statistics

Dataset .	Cora	Cite.	Pubm.	Cham	Squi.	Actor	Corn.	Texa.	Wisc.
# Nodes	2708	3327	19717	2277	5201	7600	183	183	251
# Edges	5429	4732	44338	36101	217073	33544	295	309	499
# Features	1433	3703	500	2325	2089	931	1703	1703	1703
# Classes	7	6	3	4	4	4	5	5	5
# β	0.83	0.71	0.79	0.25	0.22	0.24	0.11	0.06	0.16

DATASET Cora, Citeseer, and Pubmed are citation networks base datasets in general. In these networks, nodes represent papers, and edges denote there are citation relationships between the edge-connected nodes. Node features are the bag-of-words representation of papers. Wisconsin, Cornell,

and Texas represent links between web pages. In these datasets, nodes represent web pages, and edges are hyperlinks between them. Node features are the bag-of-words representation of papers. Chameleon and Squirrel represent the web pages in Wikipedia discussing related topics. Nodes represent web pages, and edges are mutual links between them. Node features correspond to some informative nouns on Wikipedia pages. In these ten datasets, the previous researcher has split nodes of each class into 60%, 20%, and 20% for training, validation, and testing. We measure performance on the final test sets over 10 random splits in all the experiences.

Due to limitations of space, we only represent the result of the GCN model in Cora.

Result In general, reducing the proportion of heterophily edges can increase the model's performance in downstream classification tasks and vice versa. Compared with strategies 2 and 3, although replacing the neighbors with 2-hop neighbors can alleviate the decline of model performance, the negative influence of heterophily edges on the model performance can still be observed. Therefore, we hold it is necessary to select information from edges adaptively.

7 SELF-ADAPTIVELY ADOPT ATTENTION LAYER OR GCN LAYER

To accelerate model convergence and Visualized the role of attention mechanism in different graphs and different hop nodes. In the implementation, we use the GCN layer to further improve the ability of the convolution layer. However, we also provide the performance of the model without the trick to prove the effectiveness of our proposed attention mechanism.

$$Q, K = GCN_q(X_i), GCN_k(X_j) \tag{11}$$

$$\alpha = a \cdot \mathrm{softmax}(QK^{\mathrm{T}}) + b \cdot \mathcal{A}^i \tag{12}$$

$$O = \boldsymbol{\alpha} H_i \tag{13}$$

where the X_i is the central node and X_j are the "neighbors" for convolution. a, b are learnable weights $a + b = 1$, A is normalized adjacent matrix,

	1hop	2hop	3hop
cora	0.15	0.03	0.04
citeseer	0.27	0.08	0.10
chameleon	1.56	0.74	0.88
squirrel	6.30	1.50	1.19

Table 2: The ratio of a/b

The final matrix for message propagation is the sum of the calculated attention and GCN convolution kernel in a learnable ratio. if the a is equal to zero, the convolution operation degenerates into a single layer GCN. We print the value of a, b after model convergence in Table 3. We discover that in the dataset with higher heterophily, the weight for the attention matrix tends to be higher, which shows that the attention mechanism produces a marked effect in heterophily graphs. And with the hop number increases, the weight of the attention matrix fast descends. That implies that as the distance between the nodes participating in convolution and the central node increases, the traditional GCN convolution operation makes greater contributions than the attention layer.

7.1 THE STANDARD DEVIATION OF ATTENTION SCORES

In this section, we will prove that GATv3 can be aware of the homophily and heterophily of the graphs. In Figure 3, We replace the propagation process in our model with different attention convolution layers (GATv3(GCN), GAT, GATv2) and calculate the standard deviation of the attention scores for 1,2,3 hop neighbors. We guarantee that the difference between models is only in the way of generating attention. We evaluate it on four datasets: cora and citeseer which are highly homophily graphs, and chameleon and squirrel, which are highly heterophily graphs

Results We can observe that, in all test data, the attention score calculated by traditional GAT maintains a low level of difference. On the contrary, the attention score calculated by GATv2 has the highest standard deviation. GATv3 has higher standard deviations in heterophily graphs and becomes insensitive in homophily graphs.

Moreover, As the receptive field becomes more expansive, it becomes more difficult for the attention layer to make differentiated judgments, and the attention scores will be less attentive. To overcome the limitation, we introduce the mechanism. At the same time, as the number of hops increases, the

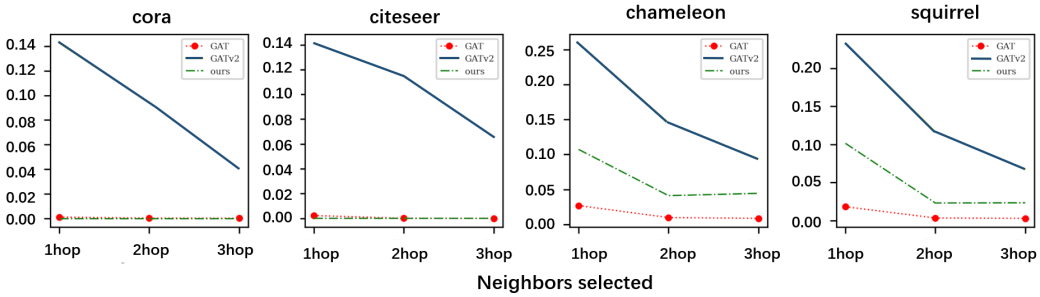

Figure 4: The standard deviation of the attention scores for 1,2,3 hop neighbors

,

relationship between the "neighbor" and the central node becomes more "estrange", making these nodes less critical in the classification task. So average-like receiving the information sent by these sections will not have a significant impact on the model performance.

8 SELECTION OF GNN MODEL

In this section, We conducted an extensive search of the methods used to calculate queries and keys in the attention layer, and compare the GNN model with the traditional linear transformation in attention calculation. We will show the experimental results on homophily graph cora and heterophily graph chameleon, and the results for other graphs will be listed in the supplementary materials.

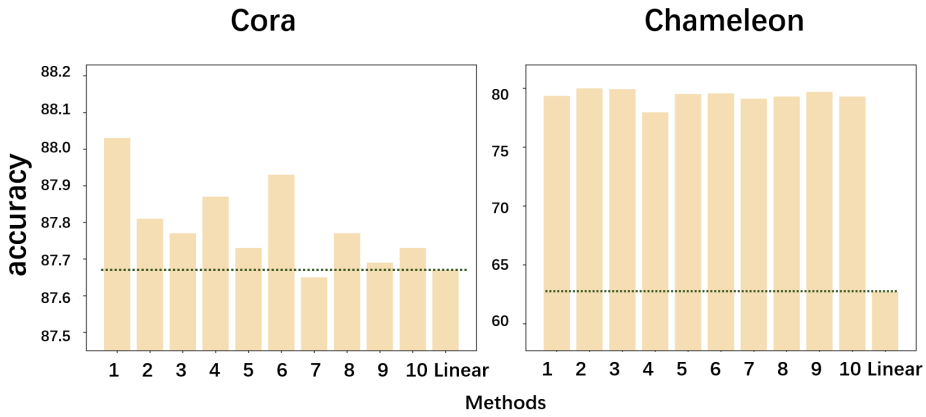

Figure 5: Different GNN models used in attention layer, Methods in the x-axis are: 1: GCN1 (singer layer GCN), 2: GCN2: (double layers of GCN), 3: SGC (Wu et al., 2019), 4: GAT1 (singer layer of GAT), 5: GAT2 (double layers of GAT), 6: GBP (Chen et al., 2020), 7: GCN1+GCN2 (using GCN1 for Q and GCN2 for K), 8: GCN1+GBP, 9: GCN2+Linear, 10: GBP+Linear, Linear: using linear transformation to generate Q and K

,

Results We extensively search the methods used to calculate queries and keys in the attention layer. We notice that GATv3 has significant advantages in heterophily graphs compared to the linear transformation in generating queries and keys. At the same time, in homophily graph, there is no noticeable difference in the node classification accuracy for the models using different methods to calculate the attention. We hold that in homophily graphs, there is substantial similarity for edge-connected nodes' features and labels. Hence, we don't need to distinguish the neighbors of nodes.

9 NODE CLASSIFICATION TASK

In this section, we will evaluate our model in node classification tasks. For the sake of fairness comparison, we also apply the trick like multi-hop and soft selected in GAT and GATv2 model in GAT-trick and GATv2-trick (Due to the limitation of memory, only 1,2,3-hop of neighbors adopted by GATv2).

Table 3: Mean classification accuracy on fully-supervised node classification task. Results for GCN, GAT, GraphSAGE, Cheby+JK, MixHop, and H2GCN-1 are taken from (Zhu et al., 2020). For GEOM-GCN and GCNII results are taken from the respective article. The best performance for each dataset is marked as bold

	cora	citeseer	Pumbed	cham	wisconsin	texas	cornell	squirrel	Actor
GCN	87.28±1.26	76.68±1.64	87.38±0.66	59.82±2.58	59.80±6.99	59.46±5.25	57.03±4.67	36.89±1.34	30.26±0.79
GATv2	86.86± 1.22	76.31± 1.62	88.77± 0.39	44.91± 2.05	64.12± 4.81	67.84± 4.75	61.35± 3.21	29.27± 1.74	34.9, 0.79
GAT	82.68±1.80	75.46±1.72	84.68±0.44	54.69±1.95	55.29±8.71	58.38±4.45	58.92±3.32	30.62±2.11	26.28±1.73
GraphSAGE	86.90±1.04	76.04±1.30	88.45±0.50	58.73±1.68	81.18±5.56	82.43±6.14	75.95±5.01	41.61±0.74	34.23±0.99
Cheby+JK	85.49±1.27	74.98±1.18	89.07±0.30	63.79±2.27	82.55±4.57	78.38±6.37	74.59±7.87	45.03±1.73	35.14±1.37
MixHop	87.61±0.85	76.26±1.33	85.31±0.61	60.50±2.53	75.88±4.90	77.84±7.73	73.51±6.34	43.80±1.48	32.22±2.34
GEOM-GCN	85.27	**77.99**	**90.05**	60.90	64.12	67.57	60.81	38.14	31.63
GCNII	88.01±1.33	77.13±1.38	90.30±0.37	62.48±2.74	81.57±4.98	77.84±5.64	76.49±4.37	N/A	N/A
H2GCN-1	86.92±1.37	77.07±1.64	89.40±0.34	57.11±1.58	86.67±4.69	84.86±6.77	82.16±4.80	36.42±1.89	35.86±1.03
FSGNN	88.01, 1.51	76.96, 1.31	89.71, 0.43	78.88, 0.97	87.25, 2.19	**87.03, 4.65**	87.03, 6.02	73.91, 1.76	**37.76, 1.0**
GAT(tirck)	86.88, 1.38	76.26, 1.68	N/A	72.85, 1.5	86.27, 3.51	84.32, 4.65	85.68, 6.63	59.21, 3.8	37.54, 1.13
GATv2-trick	86.5, 1.1	76.83, 1.16	88.91, 0.34	48.64, 1.52	87.06, 3.84	84.59, 4.02	85.68, 6.74	30.27, 2.28	34.97, 0.73
GATv3	**88.03, 1.07**	77.3, 2.06	89.35, 0.38	**79.98, 1.03**	**88.04, 4.0**	86.22, 5.47	**87.3, 6.74**	**74.85, 1.93**	37.55, 1.2

10 CONCLUTION

We identify that, for the GNN models, adjacent nodes on the graph will generate similar embedding with the process of message propagation. It will be challenging to distinguish these nodes in the downstream tasks. In the heterophily graph, it is also difficult to ensure that the edge-connected nodes have the same labels, which prevents the model from correctly predicting the labels of the nodes. The traditional attention model does not fully apply the graphic information in attention calculation which limits the model adaptively selecting the information from the neighbors. We propose a new attention mechanism that straightly involves graphic information in attention calculation and can be attentive in heterophily graphs.

We first proved the negative influence of heterophily on model performance. Further, we show the differentiation of attention scores for GATv3 in homophily and heterophily graphs proving that GATv3 can adaptively adjust the discrimination of attention scores according to the graphs. Finally, we search for the GNN models used in GATv3 and evaluate our model in opening benchmarks.

We observe that GATv3 can adaptively be aware of the homophily and heterophily graphs. Therefore, we encourage the community to use GATv3 instead or in addition to traditional GAT, especially in heterophily graphs. Further, we believe there will be more and better ways to introduce graphic information to attention calculation or increase the quality of attention scores in the future.

REFERENCES

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. Scalable graph neural networks via bidirectional propagation. *Advances in neural information processing systems*, 33:14556–14566, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Daniel D Lee, P Pham, Y Largman, and A Ng. Advances in neural information processing systems 22. Technical report, Tech. Rep., Tech. Rep, 2009.

Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3595–3603, 2019.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 338–348, 2020.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *stat*, 1050:20, 2017.

Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. *arXiv preprint arXiv:2205.11172*, 2022.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pp. 5453–5462. PMLR, 2018.

Shuo Yang, Zhiqiang Zhang, Jun Zhou, Yang Wang, Wang Sun, Xingyu Zhong, Yanming Fang, Quan Yu, and Yuan Qi. Financial risk analysis for smes with graph-based supply chain mining. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 4661–4667, 2021.

Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. Protgnn: Towards self-explaining graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9127–9135, 2022.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.