GLOBAL VIEW FOR GCN: WHY GO DEEP WHEN YOU CAN BE SHALLOW?

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing graph convolutional network (GCN) methods attempt to expand the receptive field of its convolution by either stacking up more convolutional layers or accumulating multi-hop adjacency matrices. Either approach increases computation complexity while providing a limited view of the network topology. We propose to extend k-hop adjacency matrices into one generalized exponential matrix to provide GCNs with a global overview of the network topology. This technique allows the GCNs to learn global topology without going deep and with much fewer parameters than most state-of-the-art GCNs, challenging the common assumption that deep GCNs are empirically better for learning global features. We show a significant improvement in performance in semi-supervised learning when this technique is used for common GCNs while maintaining much shallower network architectures (≤ 4 layers) than the existing ones.

1 INTRODUCTION

Graph neural network (GNN) was introduced by Gori et al. (2005) and Scarselli et al. (2009) to generalize the existing neural network approaches to process data with graph representations. It is widely used in fields such as drug discovery (Jiang et al. (2020)), protein prediction (Jumper et al. (2021)), e-commerce, social networks, and molecular chemistry (Wu et al. (2021)) where data are naturally expressed in forms of graphs. While traditional neural networks are only able to perform predictions based on data inputs, GNNs benefit from using versatile graph data structures to provide a more structural and robust prediction.

Graph convolutional networks (GCNs) (Bruna et al. (2014), Kipf & Welling (2017)) extend convolutional neural networks to GNNs by enabling local-level convolution over each graph node. In particular, the main approach consists of two steps: aggregation and update. First, each node aggregates the feature vectors of the neighboring nodes, including that of the node itself, to accumulate local structural information. Second, each aggregated node feature vector is updated by fully connected layers to improve the node feature representation.

GCN uses the adjacency matrix for learning over local neighborhoods, in particular 1-hop neighborhoods. Since long-path dependency is ignored at local levels, GCN is limited to learning only the local structures while missing the global characteristics of the entire graph. As a result, a deeper GCN (Li et al. (2019)) is often sought after; one can expand the receptive field of GCN with the concatenation of each graph convolutional layer. However, this causes over-smoothing (Li et al. (2018)) where each neighborhood has a similar and indistinguishable feature vector, resulting in a sharp drop in prediction accuracy and graph representation skills (Zhao & Akoglu (2019)). Hence, this creates a dilemma: while a deeper GCN can achieve a wider receptive field, it can also negatively affect test performance.

A series of works from Li et al. (2021); Chen et al. (2020); Rong et al. (2019); Hasanzadeh et al. (2020) introduces various techniques, including initial residual learning, normalization, and dropout to mitigate the impact of over-smoothing while employing deep GCNs. Yet, the issue of deep GCNs is not completely resolved.

In this work, we propose GlobalGCN to fundamentally overcome the dilemma and significantly reduce the computational cost. GlobalGCN generates a topological representation of the entire graph structure via one global attention matrix. It uses matrix exponential to summarize the global depen-

dence between each node, thereby providing each node with global information about its neighborhood nodes. As a result, we can avoid over-smoothing feature vectors by restricting our GCNs over shallow networks (as low as 4 layers).

In summary, we make four contributions. First, we introduce the concept of global attention matrix (GAM) to enable convolution with the largest possible receptive field. Second, we provide mathematical intuitions behind the GAM with respect to its impacts on GNNs. Third, we are able to use the GAM to have a better interpretation of how a graph is structured and how well a graph can be learned. Lastly, we empirically validate our theoretical analysis and show that global topological information helps GNNs to gain higher accuracy with fewer parameters and shallower networks in semi-supervised learning settings.

2 GLOBAL-STRUCTURE-AWARE CONVOLUTION

In this section, we provide both practical and theoretical motivation for our proposed model. Even though the adjacency matrix is able to detect the structure of a graph, it is bounded to its local view and cannot directly incorporate the global characteristic of the graph. Consequently, we define the **global attention matrix** (GAM) to describe the network topology.

Definition 2.1. Consider an undirected graph $G = (V, E)$ where V is a set of vertices, or nodes, and E is a set of edges between vertices. An adjacency matrix A is given by $A_{ij} = 1$ if there exists an edge $e \in E$ connecting the ith node $V_i \in V$ and the j th node $V_j \in V$, and 0 otherwise. We define

$$\exp(A) = \Sigma_{i \geq 0} \frac{A^i}{i!} \tag{1}$$

as the **global attention matrix** (GAM) that describes the global topology of the network.

The intuition behind this definition is as follows. For a given graph G, its adjacency matrix A, and a positive integer k, $(A^k)_{ij}$ describes the number of k-hop paths from node V_i to node V_j in G. A large value of $(A^k)_{ij}$ means more k-hop similarities between node V_i and node V_j. This similarity value can be thereby regarded as an *importance weight* between node V_i and node V_j (at k-hop level). This intuition is summarized in the following lemma.

Lemma 2.1. *If there exists a n-hop path between node V_i and node V_j in an undirected graph $G = (V, E)$, $\exp(A)_{ij} \neq 0$.*

The factorial division term in equation (1) performs two tasks. First, it factorially decays the importance weight as the number of hop of paths increases. Therefore, the similarity value for closer nodes in terms of shortest-path distance is naturally favored. Second, due to the factorial division term, the GAM is mathematically stable in the sense that its term converges to 0 rapidly enough so that the total infinite sum is guaranteed to exist.

2.1 CONVERGENCE OF GAM

We use the matrix theory to understand some properties of the GAM, especially its decaying characteristic. We aim to have a very long-distance relationship be reduced as fast as possible because graph dependencies between two majorly distant nodes should be minimum, even if they are connected by certain underlying paths.

Lemma 2.2. *For a normalized adjacency matrix $\tilde{A} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ where $\hat{A} = A + I$ and $\hat{D}$ is the degree matrix of $\hat{A}$,*

$$\lim_{k \to \infty} \frac{\|\tilde{A}^k\|_F}{k!} = 0 \tag{2}$$

with convergence rate $O(\frac{C^k}{k!})$ where C is a positive constant and $\|\cdot\|_F$ the Frobenius norm.

Proof. Based on the definition of the normalized adjacency matrix, $\tilde{A}_{ij} = \frac{\hat{A}_{ij}}{\sqrt{d_i}\sqrt{d_j}}$, where d_i and d_j are given by corresponding diagonal entries of $\hat{D}$. Since $\tilde{A}$ is symmetric, the decomposition of $\tilde{A}$ is given by

$$\tilde{A} = Q\Lambda Q^T \tag{3}$$

where Q is an orthonormal matrix and Λ is a diagonal matrix with the eigenvalues of $\tilde{A}$ on the diagonal entries. Then for a nonnegative integer k,

$$\tilde{A}^k = Q\Lambda^k Q^T. \tag{4}$$

Therefore, the Frobenius norm of $\tilde{A}^k/k!$ is bounded above by

$$\frac{\|\tilde{A}^k\|_F}{k!} = \frac{\|Q\Lambda^k Q^T\|_F}{k!} = \frac{\sqrt{\mathrm{Tr}(\Lambda^{2k})}}{k!} \leq \sqrt{N} \cdot \frac{\sigma_{\max}(|\Lambda|^k)}{k!} \leq \frac{C_1(\sqrt{N}) \cdot C_2^k}{k!} \tag{5}$$

for positive constants C_1, C_2, where $\sigma_{\max}(\cdot)$, $\mathrm{Tr}(\cdot)$ indicate the maximum eigenvalue and the trace of a given matrix respectively. N denotes the number of nodes of the graph associated with the adjacency matrix A and $|\Lambda| = diag(|\lambda_1|, ..., |\lambda_N|)$. □

Remark. C_1 depends on $\sqrt{N}$. As the number of nodes of a graph increases, more additive power terms (*i.e.* $\tilde{A}^k/k!$) contribute to the GAM due to the weighting effect of C_1. Therefore, longer path dependencies are captured.

Both lemmas suggest that the GAM is able to learn two important features. As the norm of kth power of the adjacency matrix divided by factorial terms is rapidly decaying, the GAM captures not only the connectedness of two nodes regardless of the length of the path but also the similarity between two nodes where the connection by long-distance path is heavily penalized by decay terms.

2.2 GLOBALGCN

Based on our previous theoretical analysis, we propose a novel GCN architecture using the GAM called **GlobalGCN**, where we replace the adjacency matrix in GCN with the GAM.

$$H^{(l+1)} = \sigma(\text{Dropout}(\exp(A)) \cdot H^{(l)} \cdot \text{Dropout}(W^{(l)})) \tag{6}$$

where σ is an activation function, $W^{(\cdot)}$ a weight matrix and $H^{(\cdot)}$ a feature matrix. For the experiments, we set σ as ReLU activation function (Agarap (2018)).

Dropout is used over both the GAM and weight matrices for particular purposes. Dropout in the GAM is necessary for two reasons. First, the entry values of the GAM are dominated by a small proportion of large values while the vast majority center around zero. This leads GlobalGCN to have a strong tendency towards learning by dominant edge connections unless dropout is performed over the GAM. Dropout makes it possible to learn less important edges, and thus, the underlying graph structures can be considered. Second, by using dropout, GlobalGCN is trained over different subgraphs at each iteration. The final prediction can be interpreted as an ensemble of subgraph predictions, making the neural network more robust.

In weight matrices, dropout is used for regularization; it prevents neural networks from overly relying on certain neurons.

3 RELATED WORK

GCN (Kipf & Welling (2017)) performs convolution over graphs by first aggregating node features of neighborhoods and then updating the node feature itself with the weight matrix.

$$H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W^{(l)}) \tag{7}$$

where $\tilde{A} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$. This approach suffers from over-smoothing as the number of layers becomes larger and thereby cannot incorporate the global topology of a graph.

GAT (Veličković et al. (2018)) updates each feature vector of nodes with fully connected layers and finds the self-attention matrix over each individual node. Afterward, it makes the weighted sum of the updated feature vectors over the neighborhood of each node. This method does not fully rely on the given structural information. Attention introduces extra parameters and creates computation/learning overheads. The spatial and computational complexity increases quadratically

with respect to the number of nodes. Furthermore, there is no guarantee that long-path dependency can be fully captured by the attention matrix.

PPNP (Klicpera et al. (2018)) uses personalized PageRank to generalize the graph structure and uses a multilayer perceptron (MLP) over feature vectors via

$$H = \alpha(I_n - (1-\alpha)\tilde{A})^{-1} f_\theta(X) \tag{8}$$

where X is the input feature matrix and $f_\theta(\cdot)$ is an MLP. The personalized PageRank term can be interpreted as

$$(I_n - (1-\alpha)\tilde{A})^{-1} = \Sigma_{i\geq 0}((1-\alpha)\tilde{A})^i \tag{9}$$

which is a geometric sum of normalized weight matrices. This approach attempts to incorporate graph structural information; however, its weight may not decay fast enough to penalize connections of nodes far from each other.

Clutser-GCN (Chiang et al. (2019)) performs clustering over nodes and formulates random subgraph structures. Afterwards, it restricts neighborhoods over the subgraph structure and forces each node to only learn from neighborhoods inside each cluster. The clustering method is not guaranteed to provide a correct community detection, and if it is incorrectly clustered, the learned sub-graph can be misleading and cannot grasp the topology of the entire graph.

N-GCN (Abu-El-Haija et al. (2020)) trains GCN over multiple n-hop adjacency matrices. It concatenates the feature representation from each branch and uses MLP to predict the long features. Although this method focuses on widening the perspective over n-hop, it is restricted to maximum n-hop neighborhoods and requires much more parameters to train multiple branches using 1 to n-hop adjacency matrices.

DeepGCN (Li et al. (2019)) introduces residual learning (skip connection to avoid gradient vanishing) and k-nearest neighbors clustering to find the local community such that the effect of over-smoothing is minimized. The approach is, however, limited to smaller local communities and is not able to grasp the entire graph structure. In addition, deep structure introduces more parameters.

JKNet (Xu et al. (2018)) combines all intermediate representations as $[H^{(1)}, ..., H^{(K)}]$ to learn the new representations over different hop neighborhoods. The authors prove that k-layer GCN is essentially performing random walks, and stacking them relieves the over-smoothing by having multiple random walks. However, this approach is limited to k-hop neighborhoods at most; it cannot keep track of wider neighborhood behaviors.

GraphSAGE (Hamilton et al. (2017)) performs long short-term memory (LSTM) aggregation over local neighborhoods, and updates each node feature vector based on aggregated feature vectors from LSTM. The method is still limited to the 1-hop neighborhood, and thus, is unable to view the global structure of the graph.

DropEdge (Rong et al. (2019)) implements random dropout over adjacency matrix such that

$$H^{(l+1)} = \sigma(\text{Dropout}(\tilde{A})H^{(l)}W^{(l)}). \tag{10}$$

This method becomes problematic when dropout removes critical edges where the most connection occurs. Consider, for example, a bipartite graph structure where two communities are connected by one singular edge. If that edge is dropped, the connected community is separated and is transformed into a graph with a completely different structure.

RevGNN-Deep (Li et al. (2021)) introduces a deep GCN structure by performing residual learning and normalization over feature vectors. This method requires much more parameters and a long time for training. It is restricted to the 1-hop adjacency matrix, and cannot capture the long-path dependency between two nodes.

The aforementioned papers focus on either going deep while reducing the over-smoothing effect or stacking feature vectors to increase the receptive field of the convolution. Yet, none of them focuses on ensembling multiple adjacency matrices to form a global-level adjacency matrix.

4 EXPERIMENTS

We validate GlobalGCN in semi-supervised document classification in citation networks and conduct multiple experiments over these datasets, as explained below.

Table 1: Dataset statistics.

Dataset	Classes	Nodes	Edges	Features
Cora	7	2,708	5,429	1,433
Citeseer	6	3,327	4,732	3,703
Pubmed	3	19,717	44,338	500

Table 2: Statistical information of GAM of dataset.

	Cora	Citeseer	Pubmed
Nonzero minimum	3.21×10^{-28}	5.61×10^{-45}	1.27×10^{-27}
Maximum	1.54	2.72	1.52
Nonzero(%)	84.23	40.65	100.00
Mean	9.26×10^{-4}	7.72×10^{-4}	1.12×10^{-4}
Nonzero Mean	1.10×10^{-3}	1.80×10^{-3}	1.12×10^{-4}
Median	2.04×10^{-9}	0.0	5.19×10^{-10}
Nonzero Median	8.35×10^{-9}	3.66×10^{-13}	5.19×10^{-10}
Std.	2.51×10^{-2}	2.61×10^{-2}	8.19×10^{-3}
Nonzero Std.	2.74×10^{-2}	4.10×10^{-2}	8.19×10^{-3}

4.1 DATASET

We use three citation networks for the semi-supervised learning experiments of GlobalGCN. The statistics of each dataset are summarized in Table 1. The citation network datasets (Cora, Citeseer and Pubmed) (Sen et al. (2008)) contain sparse feature vectors for each document and a list of citation links between documents. We consider all citation links as undirected edges and each document as a node, creating an undirected and unweighted graph $G = (V, E)$ with A as the adjacency matrix of the graph. Each document has a class label, and we use public splits of 20 labels per class and all feature vectors for training.

4.2 EXPERIMENTAL SETUP

We perform Bayesian optimization (Nogueira (2014–)) over GlobalGCN to maximize the validation accuracy by optimizing hyperparameters, e.g. the learning rate, the number of layers, the dimension of hidden layers, and the L_2-regularization weight.

We iterate the Bayesian optimization 1000 times, and for each iteration, we train GlobalGCN maximum 400 epochs using Adam. Under the same setting, we train GlobalGCN maximum 200 epochs for Pubmed, as the network is much denser and slower for computation. We use StepLR with a step size of 500 and a gamma of 0.3 for learning rate decay. We keep the best model for each iteration and stop training once the best validation loss is ($\geq 10\%$) larger than the validation loss. We fix the dropout rate for both weight dropout and GAM dropout as 0.6 and initialize all weights with Xavier initialization (Glorot & Bengio (2010)).

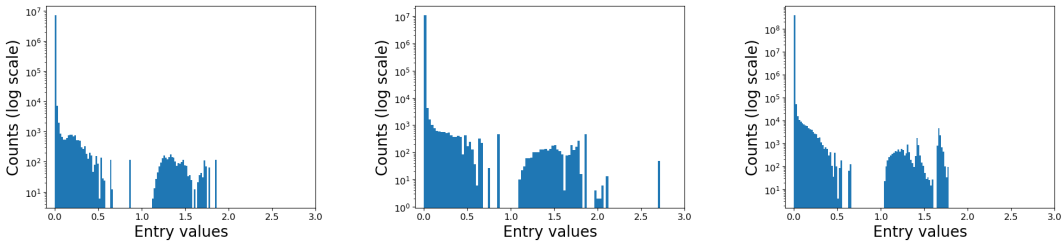
We use negative log-likelihood loss with L_2 regularization multiplied by the lambda weight.

4.3 BASELINES

We use GCN (Kipf & Welling (2017)), GAT (Veličković et al. (2018)), APPNP (Klicpera et al. (2018)), JKNet (Xu et al. (2018)), N-GCN (Abu-El-Haija et al. (2020)), HGCN (Hu et al. (2019)), and GraphAir (Hu et al. (2020)) as our baseline models. The results are reported in Abu-El-Haija et al. (2020) or Chen et al. (2020), or their respective papers. Our experiment setup is different from theirs as they are tailored to maximize their own accuracy performance. We compare our model with their best possible performance to fairly assess the performance differences.

Table 3: Summary of classification accuracy (%) results on Cora, Citeseer, and Pubmed. The number inside the bracket indicates the number of layers.

Method	Cora	Citeseer	Pubmed
GCN	81.5(2)	71.1(2)	79.0(2)
GAT	83.1(2)	70.8(2)	78.5(2)
APPNP	83.3(2)	71.8(2)	80.1(2)
JKNet	81.1(4)	69.8(16)	78.1(32)
N-GCN	83.0(2)	72.2(2)	79.5(2)
HGCN	84.5(9)	72.8(9)	79.8(9)
GraphAir	84.7(16)	72.9(16)	80.0(16)
GlobalGCN	**85.1**(2)	**73.0**(2)	**80.1**(4)



(a) Cora dataset (b) Citeseer dataset (c) Pubmed dataset

Figure 1: Histogram (with 100 bins) of entry values of the GAMs of three datasets (Cora, Pubmed, and Citeseer) in log-scale.

5 Results

In this section, we evaluate the performance of GlobalGCN against the state-of-the-art (SOTA) GNN models on three semi-supervised learning tasks. We also analyze the properties of the GAM of each dataset for better interpretation of the prediction result.

5.1 Comparison with SOTA

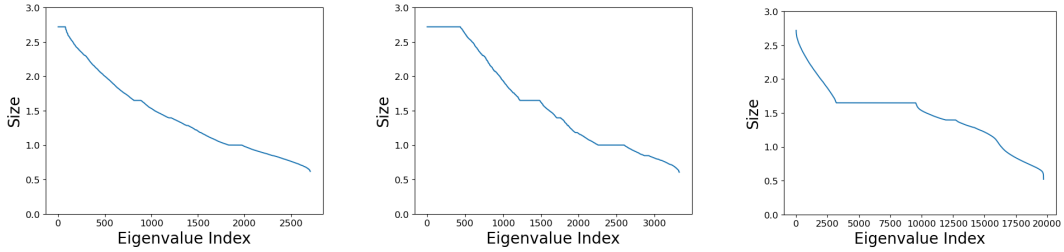
Table 3 includes experiment results of our GlobalGCN model and the other baseline models. We can see that the prediction accuracy of the GlobalGCN model clearly surpasses the others over all three datasets with much fewer layers (2-4 layers in GlobalGCN). We are able to achieve 85.1% over Cora dataset, 73.0% over Citeseer and 80.1% over Pubmed, outperforming all other models with margins while maintaining lower hidden layer dimensions (43 for Cora, 109 for Citeseer, and 92 for Pubmed).

5.2 Evaluation of Properties of GAM

In order to grasp a better understanding of the property of the GAM, we analyze the GAMs of 3 different datasets, Cora, Pubmed and Citeseer, and characterize their behavior.

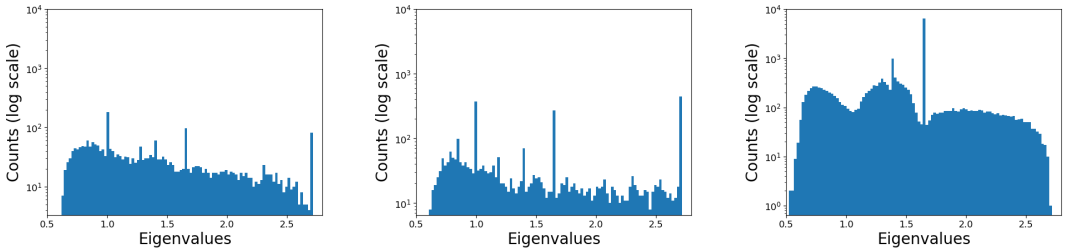
5.2.1 Distribution of Entries of GAM

Figure 1 illustrates the distribution of entry values of the GAM of each dataset. The similarity among the three datasets is that a majority of entries are located near zeros while the counts decrease exponentially as the entry value increases. This behavior corresponds to the common perception that most graph structures follow the power distribution (Aiello et al. (2001)) where the number of high-value entries exponentially decays.



(a) Cora dataset (b) Citeseer dataset (c) Pubmed dataset

Figure 2: Eigenvalues of the GAMs of three datasets (Cora, Pubmed, and Citeseer) in descending order.



(a) Cora dataset (b) Citeseer dataset (c) Pubmed dataset

Figure 3: Histogram (with 100 bins) of eigenvalues of the GAMs of three datasets (Cora, Pubmed, and Citeseer) in log-scale.

Comparing Citeseer with other datasets, Figure 1b shows that the GAM of Citeseer has a nonnegligible amount of big entries around 2.7 separate from the intermediate entry values, while the GAM of Cora or Pubmed has large entry values concentrated at around 1.5.

This can explain why GlobalGCN, or any other model, does not perform well in the Citeseer dataset in comparison with other datasets. There are a few entries of the GAM with distinctively large values which heavily affect the training of GCN on Citeseer. Table 2 shows that the entry values of the GAM of Citeseer has the largest maximum, nonzero mean, and standard deviation among the three datasets, thus confirming that Citeseer has minority edges dominating the learning.

5.2.2 Distribution of Eigenvalues of GAM

We perform the singular value decomposition (SVD) over the GAM of each dataset. Figure 2 shows the eigenvalues of the GAM generated by each dataset in descending order. While both Figure 2a and 2b show that there are multiple eigenvectors for the maximum eigenvalue, the GAM of Citeseer in 2b presents a clear salient plateau region of maximum eigenvalues.

This is another possible explanation for why GlobalGCN and all other models perform relatively poorly in Citeseer. The number of eigenvectors for the maximum eigenvalue of the GAM can be related to the number of potential underlying clusters in the graph (Chung (1997)). The GAM of Citeseer has many eigenvectors for the maximum eigenvalue, implying that there is a very rich underlying structure behind the graph. This makes learning more complicated, since GCN or GlobalGCN may focus on learning that substructure instead of capturing the desired global characteristics of the graph.

Figure 3 is a histogram of eigenvalues of the GAM of each dataset. The GAM of Pubmed in Figure 3c has very dense eigenvalue concentrations overall since the graph is fully-connected with much more nodes than the others. Figure 3b shows that for the GAM of Citeseer, eigenvectors with large eigenvalues dominate the distribution of eigenvalues, and this makes the training of GlobalGCN or GNN mainly over clusters associated with those eigenvectors and restricts the field of learning space.

Table 2 supports the explanation since the GAM of Citeseer has the least number of nonzero entries and has a median equal to 0, indicating that most connections are not there. This implies that we have many nodes inside their own clusters without any contact with nodes belonging to other clusters. This complicates the graph structure as learning is not over the entire graph but only over each small cluster.

6 DISCUSSION

6.1 CORE INSIGHTS BEHIND GAM

The GAM can be considered as a weighted sum of fast-decaying filters for the input feature matrix. It captures the similarity level between two nodes in the graph without introducing any extra parameters and is able to capture the global characteristics of any graph. Based on this explanation, we can conclude that GlobalGCN essentially performs both supervised learning and unsupervised learning simultaneously. It clusters the graph based on the adjacency matrix via the GAM and performs supervised learning based on the relationship learned from clustering.

6.2 LIMITATIONS AND FUTURE WORK

We describe several limitations of GlobalGCN and potential future directions for improvements.

6.2.1 LARGE-SCALE TRAINING

The GAM is a dense matrix for our implementation. The size of the matrix increases quadratically with respect to the number of nodes. In contrast, adjacency matrices over large-scale data are usually sparse. Therefore, it is required to utilize the sparsity in order to overcome the memory bottleneck of the GAM computation.

6.2.2 SPECTRAL PRECONDITIONING

According to the experiment results, eigenvalues of the adjacency matrix plays a key role in controlling the converging speed of adjacency matrix power. We can precondition eigenvalues by using the relationship $cAx = c\lambda x$ such that we can modify the size of eigenvalues of the adjacency matrix to enable a wider coverage of neighborhoods. Refer to the Appendix for detailed explanations.

7 CONCLUSION

We have proposed a novel GCN architecture called GlobalGCN which uses the global attention matrix (GAM) from matrix exponential to learn the global topology/structure of the graph. It is able to learn over the graph at the maximum receptive field while taking the similarity between each node into consideration. GlobalGCN shows significant improvement in prediction accuracy over semi-supervised learning tasks and is easy to implement. It outperforms SOTA with notable margins while maintaining shallower network architectures (as few as 4 layers) with fewer parameters than most existing GCN architectures.

REFERENCES

Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *uncertainty in artificial intelligence*, pp. 841–851. PMLR, 2020.

Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

William Aiello, Fan Chung, and Linyuan Lu. A random graph model for power law graphs. *Experimental Mathematics*, 10(1):53–66, 2001. doi: 10.1080/10586458.2001.10504428.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs". In *International Conference on Learning Representations*, 2014.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 257–266, 2019.

F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734 vol. 2, 2005.

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

Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian graph neural networks with adaptive connection sampling. In *International conference on machine learning*, pp. 4094–4104. PMLR, 2020.

Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan. Hierarchical graph convolutional networks for semi-supervised node classification, 2019.

Fenyu Hu, Yanqiao Zhu, Shu Wu, Weiran Huang, Liang Wang, and Tieniu Tan. Graphair: Graph representation learning with neighborhood aggregation and interaction. *Pattern Recognition*, pp. 107745, 2020.

Hao Jiang, Peng Cao, MingYi Xu, Jinzhu Yang, and Osmar Zaiane. Hi-gcn: A hierarchical graph convolution network for graph embedding learning of brain network and brain disorders prediction. *Computers in Biology and Medicine*, 127:104096, 2020. ISSN 0010-4825.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276, 2019.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pp. 6437–6449. PMLR, 2021.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29:93, Sep. 2008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

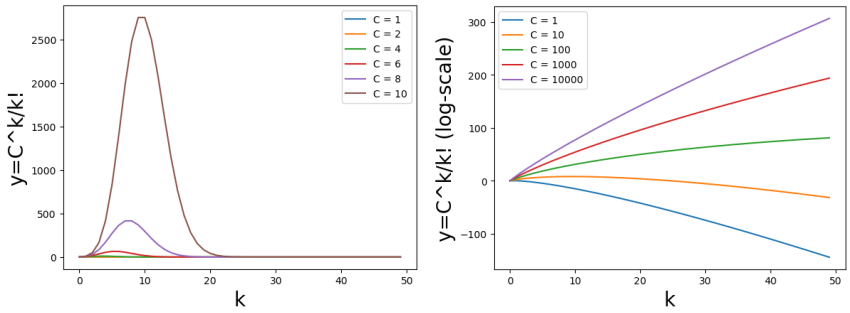
Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pp. 5453–5462. PMLR, 2018.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2019.

A APPENDIX

A.1 DISTRIBUTION OF BOUNDS OVER MATRIX NORM

In Lemma 2.2, we provide the upper bound of the Frobenius norm of each term appearing in the infinite sum which defines the GAM (see equation (1)). Here, we test the behavior of the upper bound $\frac{C^k}{k!}$ for a positive constant C and a nonnegative integer k. Figure 4a shows the shape of the bound over different values for constant C. This gives an overview of the impact of C over the asymptotic behavior of the bound. As C increases, the bandpass regions become larger, indicating that more neighborhoods are viewed, corresponding to our interpretation of Lemma 2.2. Figure 4b gives a further overview of the behavior of the bound over large C in the log scale. This graph indicates that with an increase of constant terms, the covered neighborhoods would exponentially increase. As a result, we could perform matrix preconditioning over eigenvalues of the normalized adjacency matrix to flexibly adjust the receptive field.



(a) Distribution of $\frac{C^k}{k!}$ over small C constants

(b) Distribution of $\frac{C^k}{k!}$ over large C constants in log-scale

Figure 4: Distribution of $\frac{C^k}{k!}$ over multiple constant values.