PERSONALIZED PAGERANK MEETS GRAPH ATTENTION NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

There has been a rising interest in graph neural networks (GNNs) for representation learning over the past few years. GNNs provide a general and efficient framework to learn from graph-structured data. However, GNNs typically only use the information of a very limited neighborhood for each node. A larger neighborhood would be desirable to provide the model with more information. However, increasing the size of the neighborhood is not trivial since neighborhood aggregation over many layers leads to over-smoothing. In this work, we incorporate the limit distribution of Personalized PageRank (PPR) into graph attention networks (GATs) to address this issue. Intuitively, message aggregation based on Personalized PageRank corresponds to infinitely many neighborhood aggregation layers. We show that our models outperform a variety ofbaseline models across all datasets used for our experiments. Our implementation is publicly available online.

1 INTRODUCTION

There has been a rising interest in graph neural networks (GNNs) (Gori et al., 2005; Scarselli et al., 2008) for representation learning over the past few years (Duvenaud et al., 2015; Atwood & Towsley, 2016; Bronstein et al., 2017; Monti et al., 2016). GNNs provide a general and efficient framework to learn from graph-structured data. Thus, GNNs are easily applicable in problems where the data can be represented as a set of nodes and the prediction depends on the relationships (edges) between the nodes. Such problems include molecules, social networks, knowledge graphs, and recommendation systems.

A GNN can be viewed as a message-passing network (Gilmer et al., 2017), where each node iteratively updates its state by interacting with its neighbors. GNN variants (Wu et al., 2019; Xu et al., 2018a; Li et al., 2016) mostly differ in how each node aggregates the representations of its neighbors and combines them with its own representation. Several works have been proposed to improve the basic neighborhood aggregation scheme by using attention mechanisms (Kearnes et al., 2016; Hamilton et al., 2018; Veličković et al., 2018), random walks (Abu-El-Haija et al., 2020; Ying et al., 2018; Li et al., 2018), edge features (Kearnes et al., 2016; Gilmer et al., 2017; Schlichtkrull et al., 2018) and making it more scalable on large graphs (Chen et al., 2018; Ying et al., 2018). However, all of these methods only use the information of a very limited neighborhood for each node. A larger neighborhood would be desirable to provide the model with more information.

Increasing the size of the neighborhood is not trivial since neighborhood aggregation in this scheme is essentially a type of Laplacian smoothing and too many layers lead to over-smoothing (Li et al., 2018). Xu et al. (2018b) highlighted the same problem by establishing a relationship between the message passing algorithm of Graph Convolutional Network (GCN) by Kipf & Welling (2017) and a random walk. Using this relationship we see that GCN converges to this random walk's limit distribution as the number of layers increases. The limit distribution is a property of the graph as a whole, rather than the property of the starting node. As such GCN's performance necessarily deteriorates for a high number of layers (which means the aggregation over the larger size of the neighborhood). Kipf & Welling (2017) also reported that the performance of GCNs decreases for the number of layers beyond 2.

In this work, we incorporate the limit distribution of Personalized PageRank (PPR) (Page et al., 1999) into GNNs to address these limits of the GNN family. Intuitively, message aggregation based on Personalized PageRank corresponds to infinitely many neighborhood aggregation layers where the node influence decays exponentially with each layer. Furthermore, we use an approximate Personalized PageRank matrix, which approximates the PPR matrix with a sparse matrix for scalability.

Now, to fully leverage the expressive power of GNNs, we incorporate this PPR matrix into Graph Attention Networks (GATs) (Veličković et al., 2018). We call our model PPRGAT. GAT pioneered the use of attention-based neighborhood aggregation, in one of the most popular GNN variants - Graph Attention Network (GAT). In GAT, every node updates its representation by attending to its neighbors using its own representation as to the query. This generalizes the standard averaging or max-pooling of neighbors (Kipf & Welling, 2017; Hamilton et al., 2018), by allowing every node to compute a weighted average of its neighbors according to the neighbor's importance. The work of Hamilton et al. (2018) also generalizes the Transformer's (Vaswani et al., 2017) self-attention mechanism, from sequences to graphs. GAT is one of the most popular GNN architectures (Bronstein et al., 2021) and is considered as the state-of-the-art neural architecture for learning with graphs (Wang et al., 2019).

To incorporate the PPR matrix into GAT, we consider two versions. First, we concatenate the PPR matrix information to the node features x_i, x_j when we compute the attention score between node i and node j if there is an edge between node i and node j. In this version, we use the original adjacency matrix and the neighbors are identified by this adjacency matrix as in the standard GATs (Veličković et al., 2018). Second, we replace the original adjacency matrix with the sparse approximate PPR matrix. In this version, only the top k indices (nodes) of i's row of the approximate PPR matrix are considered the neighbors of node i.

2 Relate work

Our work builds upon a number of recent advancements in deep learning methods and Personalized PageRank for graph-structured data. In this section, we first introduce our notation and then review prior works related to the neighbor aggregation methods on graphs. Let $G = (V, E)$ denote a graph with a set of nodes $V = \{v_1, \cdots, v_N\}$, connected by a set of edges $E \subseteq V \times V$. Node features are organized in a compact matrix $X \in R^{N\times D}$ with each row representing the feature vector of one node, where N is the number of nodes and D is the dimension of the features. Let $A \in R^{N\times N}$ denote the adjacent matrix that describes graph structure of $G : A_{ij} = 1$ if there is an edge e_{ij} from node i to node j, and 0 otherwise. By adding a self-loop to each node, we have $\tilde{A} = A + I_N$ to denote the adjacency matrix of the augmented graph, where $I_N \in R^{N\times N}$ is an identity matrix.

For a semi-supervised node classification task, given a set of labeled nodes $\{(v_i, y_i), i = 1, \cdots, n\}$, where y_i is the label of node i and $n < N$, we learn a function $f(X, A, W)$, parameterized by W, that takes node features X and graph structure A as inputs and yields a node embedding matrix $H \in R^{N\times D'}$ for all nodes in V, where D' is the dimension of the final representation. Subsequently, H is fed to a classifier to predict the class label of each unlabeled node. To learn the model parameter W, we typically minimize an empirical risk over all labeled nodes:

$$\mathcal{R} = \frac{1}{n}\sum \mathcal{L}(f_i(X, A, W), y_i), \tag{1}$$

where $f_i(X, A, W)$ denotes the output of $f(X, A, W)$ for node i and $\mathcal{L}(\cdot)$ is a loss function, such as the cross-entropy loss that measures the error between model predictions and class labels. Although there exist many different GNN algorithms that can solve Equation 1, the main difference among them is how the encoder function $f(X, A, W)$ is defined.

2.1 Neighbor Aggregation Methods

Most of the graph learning algorithms are based on a neighbor aggregation mechanism. The basic idea is to learn a parameter-sharing aggregator, which takes feature vector x_i of node i and its neighbors' feature vectors $x_j, j \in N_i$ as inputs and outputs a new feature vector for the node i. Essentially, the aggregator function aggregates lower-level features of a node and its neighbors and generates high-level feature representations. The popular Graph Convolution Networks (GCNs)

(Kipf & Welling, 2017) fall into the category of neighbor aggregation. For a 2-layer GCN, its encoder function can be expressed as:

$$f(X, A, W) = \text{softmax}\left(\hat{A}\sigma(\hat{A}XW^{(0)})W^{(1)}\right), \tag{2}$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, $D_{ii} = \sum_j \tilde{A}_{ij}$, and $W^{(\cdot)}$s are the learnable parameters of GCNs. Apparently, GCNs define the aggregation coefficients as the symmetrically normalized adjacency matrix $\tilde{A}$, and these coefficients are shared across all GCN layers. More specifically, the aggregator of GCNs can be expressed as

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i} \tilde{A}_{ij} h_j^{(l)} W^{(l)}\right), \tag{3}$$

where $h_j^{(l)}$ is the hidden representation of node j at layer l , $h^{(0)} = X$, and $\mathcal{N}_i$ denotes the set of all the neighbors of node i, including itself.

Since a fixed adjacency matrix $\hat{A}$ is used for feature aggregation, GCNs can only be used for the transductive learning tasks, and if the graph structure changes, the whole GCN model needs to be retrained or fine-tuned. To support inductive learning, GraphSage (Hamilton et al., 2018) proposes to learn parameterized aggregators (e.g., mean, max-pooling or LSTM aggregator) that can be used for feature aggregation on unseen nodes or graphs. To support large-scale graph learning tasks, GraphSage uniformly samples a fixed number of neighbors per node and performs computation on a sampled subgraph at each iteration. Although it can reduce computational cost and memory usage significantly, its accuracies suffer from random sampling and partial neighbor aggregation.

2.2 Graph Attention Networks

Recently, attention networks have achieved state-of-the-art results in many computer vision and natural language processing tasks, such as image captioning (Xu et al., 2015b) and machine translation (Bahdanau et al., 2014). By using learnable weights on each input, the attention mechanism can decide how much attention to give to each input in order to gather the most useful information. Extending the attention mechanism to graph-structured data, Graph Attention Networks (GATs) (Veličković et al., 2018) utilize an attention-based aggregator to generate attention coefficients over all neighbors of a node for feature aggregation. In particular, the aggregator function of GATs is similar to that of GCNs:

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i} a_{ij}^{(l)} h_j^{(l)} W^{(l)}\right), \tag{4}$$

except that (1) $a_{ij}^{(l)}$ is the attention coefficient of edge e_{ij} at layer l, assigned by an attention function rather than by a predefined $\tilde{A}$, and (2) different layers utilize different attention functions, while GCNs share a predefined $\tilde{A}$ across all layers. To increase the capacity of attention mechanism, GATs further exploit multi-head attentions for feature aggregation: each head works independently to aggregate information, and all the outputs of multi-heads are then concatenated to form a new feature representation for the next layer. In principle, the learned attention coefficient can be viewed as an importance score of an edge.

In GAT (Veličković et al., 2018), a scoring function $e : R^d \times R^d \to R$ computes a score for every edge (j, i), which indicates the importance of the features of the neighbor j to the node i:

$$e(h_i, h_j) = \text{LeakyReLU}(a^T \cdot [Wh_i||Wh_j]) \tag{5}$$

where $a \in R^{2d'}, W \in R^{d'\times d}$ are learned, and $||$ denotes vector concatenation. These attention scores are normalized across all neighbors $j \in \mathcal{N}_i$ using softmax, and the attention function is defined as:

$$\alpha_{ij} = \text{softmax}_j(e(h_i, h_j)) = \frac{\exp(e(h_i, h_j))}{\sum_{j' \in \mathcal{N}_i} \exp(e(h_i, h_{j'})}) \tag{6}$$

Then, GAT computes a weighted average of the transformed features of the neighbor nodes (followed by a non-linearity σ) as the new representation of node i, using the normalized attention coefficients:

$$h_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot W h_j\right) \tag{7}$$

Later, Brody et al. (2021) pointed out that the ranking of attention coefficients in Equation (6 is global for all nodes in the graph, and is unconditioned on the query node. This attention coefficient can be considered static attention. Brody et al. (2021) proposed GATv2, which modifies Equation (5) to

$$e(h_i, h_j) = a^T \text{LeakyReLU}(W \cdot [h_i || h_j]) \tag{8}$$

and showed that it can yield dynamic attention using this simple modification.

In our work, we implement our PPRGAT for these two versions: GAT (Veličković et al., 2018) and GATv2 (Brody et al., 2021). We will call the second version (PPRGAT applied to GATv2) as PPRGATv2 if the distinguishment is needed. We show that PPRGAT outperforms both baselines.

2.3 Personalized PageRank and GNNs

Despite their success of GNNs, over-smoothing is a common issue faced by GNNs (Li et al. (2018)), which means that the representations of the graph nodes of different classes would become indistinguishable when stacking multiple layers, which hurts the model performance. Veličković et al. (2018) showed that the model performance of GCNs decreases if the number of layers goes beyond 2. Due to these reasons, GNNs typically only use the information of a very limited neighborhood for each node. A larger neighborhood would be desirable to provide the model with more information.

In order to address this issue, there have been efforts to incorporate Personalized PageRank into GNNs recently. Personalized PageRank (Page et al., 1999) is a standard tool for finding vertices in a graph that are most relevant to a query. To personalize PageRank, one adjusts node weights or edge weights that determine teleport probabilities and transition probabilities in a random surfer model. The idea is to use the Personalized PageRank limit distribution, instead of the original edges, as the weights to aggregate the node features. Specifically, Klicpera et al. (2019) proposes "Predict then Propagate" methods: It first predicts the node representations using a simple MLP, and then aggregates these node representations using Personalized PageRank limit distributions. Bojchevski et al. (2020) uses a similar approach, except that it pre-compute the limit distributions and uses only top k important nodes for scalability.

3 Proposed Method: PPRGAT

In this section, we define our problem and introduce our new model, PPRGAT.

Our work is inspired by GAT described in Section 2.2 and Personalized PageRank described in Section 2.3. We incorporate Personalized PageRank limit distribution into the attention coefficients in GAT layers. In this way, GNNs can be aware of the whole graph structure and learn more efficient attention weights.

3.1 Approximate PPR Distribution as Sparse Matrix

Personalized PageRank matrix is defined by

$$\Pi^{\text{ppr}} = \alpha \left(I_n - (1-\alpha) D^{-1} A\right)^{-1} \tag{9}$$

Each row $\pi(i) := \Pi^{\text{ppr}}_{i,:}$ is equal to the Personalized PageRank vector of node i. We are interested in efficient and scalable algorithms for computing (an approximation) of Personalized PageRank.

Random walk sampling (Fogaras et al., 2005) is one such approximation technique. While simple to implement, in order to guarantee at most ϵ absolute error with probability of $1 - \frac{1}{n}$, we need $O\left(\frac{\log n}{\epsilon^2}\right)$ random walks.

For this work, we adopt the approach by Andersen et al. (2006). It offers a good balance of scalability, approximation guarantees, and ease of distributed implementation. They show that $\pi(i)$ can be weakly approximated with a low number of non-zero entries using a scalable algorithm that applies a series of operations that can be executed in a distributed manner.

When the graph is strongly connected $\pi(i)$ is non-zero for all nodes. Nevertheless, we can obtain a good approximation by truncating small elements to zero since most of the probability mass in the Personalized PageRank vectors $\pi(i)$ is localized on a small number of nodes (Andersen et al., 2006; Nassar et al., 2015). Thus, we can approximate $\pi(i)$ with a sparse vector and in turn approximate Π^{ppr} with a sparse matrix.

Once we obtain an approximation Π^{ϵ} of Π^{ppr} we can either use it directly to propagate information, or we can renormalize it via $D^{\frac{1}{2}}\Pi^{\epsilon}D^{-\frac{1}{2}}$ to obtain an approximation of the matrix Π^{sym}.

We additionally truncate Π^{ϵ} to contain only the top k largest entries for each row $\pi(i)$. We call it $\Pi^{\epsilon,k}$. Note that this computation can be parallelized for each node i and hence computationally efficient. Furthermore, we only need to pre-compute $\Pi^{\epsilon,k}$ once before training and use it during training.

3.2 GAT LAYER WITH PPR MATRIX

Recall the importance of the features of the neighbor j to the node i in GAT is Equation (5). To incorporate the PPR matrix information into the GAT layer, we modify this to

$$e(h_i, h_j) = \text{LeakyReLU}(a^T \cdot [Wh_i || Wh_j || \Pi_{ij}^{\epsilon,k}]) \tag{10}$$

Using this simple modification, we can naturally incorporate the global graph structure into the local GAT layer.

We also apply the same approach to GATv2 (Brody et al., 2021). For this, Equation (8) is modified to

$$e(h_i, h_j) = a^T \text{LeakyReLU}(W \cdot [h_i || h_j || \Pi_{ij}^{\epsilon,k}]) \tag{11}$$

We call this model as PPRGATv2.

Now, when we normalize the attention scores across all neighbors $j \in \mathcal{N}_i$ as in Equation (6) and aggregate the features as in Equation (7), we consider two approaches for the definition of neighbors $\mathcal{N}_i$. First, the default version is to aggregate over the top-k nodes of $\pi(i)$. Second, we aggregate over the original neighbors defined by the original adjacency matrix. We call this variant PPRGAT-local, PPRGATv2-local, respectively.

4 EXPERIMENTS

We compare our proposed model against a wide variety of GAT based models and PPR based models. Specifically, GAT based baselines include GAT (Veličković et al., 2018) and GATv2 (Brody et al., 2021), and PPR based baselines include (A)PPNP Klicpera et al. (2019) and PPRGo (Bojchevski et al., 2020). For our proposed model, we evaluate the four variants described in Section 3: PPRGAT, PPRGAT-loc , PPRGATv2, and PPRGATv2-loc.

4.1 DATASETS

Transductive learning We test the models on the three standard citation network benchmark datasets: Cora, Citeseer, and Pubmed (Sen et al., 2008). For these datasets, we follow the transductive experimental setup of Yang et al. (2016). In all of these datasets, nodes correspond to documents, and edges correspond to citations. Node features correspond to elements of a bag-of-words representation of a document. Each node has a class label. While large graphs do not necessarily have a larger diameter (Leskovec et al., 2005), note that these graphs indeed have average shortest

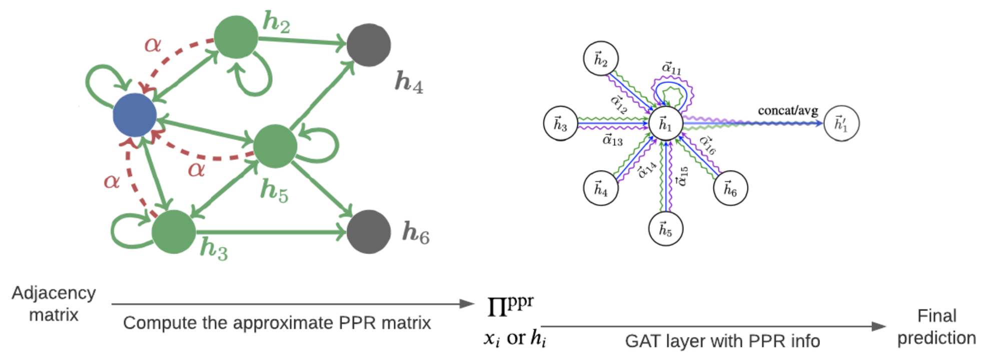

Figure 1: Illustration of PPRGAT. First, we precompute $\Pi^{\epsilon,k}$ from the adjacency matrix. Second, $\Pi^{\epsilon,k}_{ij}, x_i, x_j$ are used together to generate the attention score from node j to nose i in the GAT layer. The model is trained end-to-end.

Table 1: Summary of the graph datasets used in the experiments

	Cora	**Citeseer**	**Pubmed**	**PPI**
Task	Transductive	Transductive	Transductive	Inductive
Nodes	2708 (1 graph)	3327 (1 graph)	19717 (1 graph)	56944 (24 graphs)
Edges	5429	4732	44338	818716
Features/Node	1433	3703	500	50
Classes	7	6	3	121 (multilabel)
Training Examples	140	120	60	44906 (20 graphs)
Validation Examples	500	500	500	6514 (2 graphs)
Test Examples	1000	1000	1000	5524 (2 graphs)

path lengths between 5 and 10 and therefore a regular two-layer GCN cannot cover the entire graph. This characteristic make these citation datasets useful to evaluate the effectiveness of introducing PPR information into the shallow layers of GNNs.

Inductive learning We test the models on protein-protein interaction (PPI) dataset that consists of graphs corresponding to different human tissues (Zitnik & Leskovec, 2017). In this task, the testing graphs remain completely unobserved during training.

Table 1 summarizes the graph datasets used in the experiments.

4.2 Experimental Setup

For fair comparisons, for all of our 4 variants of PPRGAT, we use the same hyperparameters as the GAT based baselines and PPR based baselines if possible. Specifically, GAT based setup is for our 4 variants of PPRGAT and the two baselines: GAT and GATv2. PPR based setup is applicable to our 4 variants of PPRGAT and the two baselines: (A)PPNP and PPRGo. More details are described below.

4.2.1 Transductive learning

Common setup All the models are initialized using Glorot initialization (Glorot & Bengio, 2010). We use an early stopping strategy on validation loss, with a patience of 100 epochs.

GAT related setup We apply a two-layer GAT model. For the first layer, we use 8 features and 8 attention heads. And it's followed by a ReLU (Xu et al., 2015a). For the second layer, the number of output features is the same as the number of classes. Also for the second layer, we use 1 head except for PubMed. For PubMed, we use 8 output attention heads, following the observation from

Table 2: Classification accuracies (in %) of different node classification algorithms on the citation datasets. Results are the averages of 10 runs.

Model	**Cora**	**Citeseer**	**Pubmed**
GAT	83	70.8	79
GATv2	82.9	71.6	78.7
APPNP	83.3	71.8	80.1
PPRGo	74.2	65.6	70.7
PPRGAT	83.9	**72.5**	80.4
PPRGAT-loc	**84**	72.2	80.1
PPRGATv2	83.8	72.4	**80.5**
PPRGATv2-loc	83.9	72.1	80.2

Monti et al. (2016). Then we apply softmax nonlinear activation for the final output. Furthermore, dropout (Srivastava et al., 2014) with p = 0.6 is applied to both layers' inputs and the normalized attention coefficients

PPR related setup We use teleport parameter $\alpha = 0.25$ for all datasets. For the approximation Π^{ϵ} of Π^{ppr}, we use $\epsilon = 10^{-4}$. Furthermore, for the truncated $\Pi^{\epsilon,k}$ of Π^{ϵ} introduced in Section 3.1, with $k = 32$. In other words, we keep only the top 32 elements of each PPR distribution $\pi(i)$

4.2.2 Inductive learning

Common setup All the models are initialized using Glorot initialization (Glorot & Bengio, 2010). We use an early stopping strategy on micro-F1 (inductive) score on the validation sets, with a patience of 100 epochs.

GAT related setup For the inductive learning task (PPI dataset), we apply a three-layer GAT model. Both of the first two layers consist of K = 4 attention heads computing 256 features (for a total of 1024 features), followed by an ELU nonlinearity. The final layer is used for the multi-label classification. For this final layer, we use 6 attention heads computing and 121 features each, that are averaged and followed by a sigmoid activation. As observed by Veličković et al. (2018), the training sets for the PPI dataset are sufficiently large, and we found no need to apply dropout for regularization.

PPR related setup We use teleport parameter $\alpha = 0.25$. For the approximation Π^{ϵ} of Π^{ppr}, we use $\epsilon = 10^{-4}$. Furthermore, for the truncated $\Pi^{\epsilon,k}$ of Π^{ϵ} introduced in Section 3.1, with $k = 32$. In other words, we keep only the top 32 elements of each PPR distribution $\pi(i)$

4.3 Evaluation Results

The results of our comparative evaluation experiments are summarized in Tables 2 and 3. The experiment results successfully demonstrate our models outperform the baselines across all four datasets. This is in line with our expectations as per the discussion 3. It is important to note that our models outperform the baseline on the PPI dataset. This implies that the approximate PPR matrix of the unseen data along side the learned weights can still efficiently predict the outputs

5 Conclusion

We have introduced PPRGAT, novel convolution-style neural networks that operate on graph-structured data. It incorporates the PPR information into the Graph Attention Networks (GATs). In this way, PPRGAT utilizes the full potential of GATs, while incorporating the whole adjacency matrix information via PPR matrix, even with the shallow GAT layers. Approximating the PPR matrix is very efficient following the approach by Andersen et al. (2006), and it can be parallelized. Furthermore, computing the PPR matrix can be considered as a preprocess step before starting the training. This PPR matrix is stored in memory and reused during the training. This implies that PPRGATs are scalable at the same level of GATs.

Table 3: Classification micro-F1 scores (in %) of different node classification algorithms on the PPI dataset. Results are the averages of 10 runs.

Model	**PPI**
GAT	96.5
GATv2	96.3
APPNP	96.7
PPRGo	87.8
PPRGAT	**97.5**
PPRGAT-loc	**97.1**
PPRGATv2	97.4
PPRGATv2-loc	97.1

There are several potential improvements and extensions to PPRGATs. First, statistically analyzing the relation between PPR matrix distribution, original attention score, and the attention score using PPR matrix will provide the intuition behind what makes PPRGATs better than the other baselines. Second, there have been some works on graph sparsification (Calandriello et al., 2018; Chakeri et al., 2016; Rong et al., 2020; Hasanzadeh et al., 2020; Zheng et al., 2020). It will be interesting to see how the truncated PPR matrix will compare with the subgraph after these graph sparsification.

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.

Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 475–486. IEEE, 2006.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, pp. 1993–2001, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2014.

Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2020. ACM.

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

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Daniele Calandriello, Alessandro Lazaric, Ioannis Koutis, and Michal Valko. Improved large-scale graph learning through ridge spectral sparsification. In *International Conference on Machine Learning*, pp. 688–697. PMLR, 2018.

Alireza Chakeri, Hamidreza Farhidzadeh, and Lawrence O Hall. Spectral sparsification in spectral clustering. In *2016 23rd international conference on pattern recognition (icpr)*, pp. 2301–2306. IEEE, 2016.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling, 2018.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 2015.

Dániel Fogaras, Balázs Rácz, Károly Csalogány, and Tamás Sarlós. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

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.

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.

Marco Gori, Gabriele Monfardini, and Franco 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. IEEE, 2005.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

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.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8): 595–608, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2019.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 177–187, 2005.

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.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *In International Conference on Learning Representations*, 2016.

F Monti, D Boscaini, J Masci, E Rodola, J Svoboda, and MM Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Huda Nassar, Kyle Kloster, and David F Gleich. Strong localization in personalized pagerank vectors. In *International Workshop on Algorithms and Models for the Web-Graph*, pp. 190–202. Springer, 2015.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *deep graph convolutional networks on node classification," in International Conference on Learning Representations (ICLR),*, 2020.

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, 2008.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks, 2018.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. Improving graph attention networks with large margin-based constraints. *arXiv preprint arXiv:1910.11945*, 2019.

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.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015a.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057. PMLR, 2015b.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462. PMLR, 10–15 Jul 2018b.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, pp. 974–983, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pp. 11458–11468. PMLR, 2020.

Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):190–198, 2017.

A Appendix

You may include other additional sections here.