Graphs, Entities, and Step Mixture

Kyuyong Shin [1] **Wonyoung Shin** [2] **Jung-Woo Ha** [1] **Sunyoung Kwon** [1]

Abstract

Existing approaches for graph neural networks commonly suffer from the oversmoothing issue, regardless of how neighborhoods are aggregated. Most methods also focus on transductive scenarios for fixed graphs, leading to poor generalization to unseen graphs. To address these issues, we propose a new graph neural network that considers both edge-based neighborhood relationships and node-based entity features, i.e. **G**raph **E**ntities with **S**tep **M**ixture via *random walk* (GESM). GESM employs a mixture of various steps through random walk to alleviate the oversmoothing problem, attention to dynamically reflect interrelations depending on node information, and structure-based regularization to enhance embedding representation. With intensive experiments, we show that the proposed GESM achieves state-of-the-art or comparable performances on eight benchmark graph datasets.

1. Introduction

The majority of studies on graph neural networks (GNNs) (Scarselli et al., 2008) have predominantly depended on edges to aggregate features of neighboring nodes. These edge-based methods are premised on the concept of relational inductive bias within graphs (Battaglia et al., 2018), which implies that two connected nodes have similar properties and are more likely to share the same label (Kipf & Welling, 2017). To improve the neighborhood aggregation scheme, some studies have incorporated node information; they fully utilize node information and reduce the effects of structural (edge) information, so that weights used for neighborhood aggregation differ according to the feature of nodes (Veličković et al., 2018).

Regardless of neighborhood aggregation schemes, most methods, however, suffer from a common problem where neighborhood information is considered to a limited degree (Klicpera et al., 2019). Consequently, information becomes localized and access to global information is restricted (Xu et al., 2018), leading to poor performance on datasets in which only a small portion is labeled (Li et al., 2018).

In order to address the aforementioned issues, we propose a novel method, **G**raph **E**ntities with **S**tep **M**ixture via *random walk* (GESM), which considers information from all nodes in the graph and can be generalized to new graphs by incorporating *random walk* and *attention*. *Random walk* enables our model to be applicable to previously unseen graph structures, and a mixture of random walks alleviates the oversmoothing problem, allowing global information to be included during training. Hence, our method can be effective, particularly for nodes in the periphery or a sparsely labeled dataset. The *attention* mechanism also advances our model by considering node information for aggregation. This enhances the generalizability of models to diverse graph structures.

Despite the attention mechanism, it is likely that some homogeneous neighbor nodes are not clustered closely in the embedding space. We employ a triplet loss-based regularization term (Gordo et al., 2017), which enforces homogeneous neighbor nodes to be closer.

To validate our approach, we conducted extensive experiments on eight standard benchmark datasets and demonstrate that our proposed model is consistently competitive and is applicable to both transductive and inductive learning tasks. We also provide an in-depth analysis and confirm our model's superiority on the oversmoothing issue.

2. Graph Entity and Step Mixture (GESM)

First, we define the notations used in this paper for convenience. Nodes are represented as a feature matrix $X \in \mathbb{R}^{n\times f}$, where n and f respectively denote the number of nodes and the input dimension per node. The adjacency matrix of graph $\mathcal{G}$ is represented as $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and a learnable weight matrix is denoted by W. The addition of self-loops to the adjacency matrix is $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_n$, and the column normalized matrix of $\widetilde{\boldsymbol{A}}$ is $\hat{\widetilde{\boldsymbol{A}}} = \widetilde{\boldsymbol{A}}\boldsymbol{D}^{-1}$ with $\hat{\widetilde{\boldsymbol{A}}}^0 = \boldsymbol{I}_n$.

[1]Clova AI Research, NAVER Corp. [2]Naver Shopping, NAVER Corp.. Correspondence to: Sunyoung Kwon <sunny.kwon@navercorp.com>.

ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020), Vienna, Austria, 2020. Copyright 2020 by the author(s).

Table 1. Experimental results on the public benchmark datasets. Evaluation metrics on transductive and inductive learning datasets are classification accuracy (%) and F1-score, respectively.

	Transductive			Inductive
Method	Cora public (5.1%)	Citeseer public (3.6%)	Pubmed public (0.3%)	PPI
GCN (Kipf & Welling, 2017)	81.5	70.3	79.0	-
GraphSAGE (Hamilton et al., 2017)	-	-	-	0.612
GAT (Veličković et al., 2018)	83.0	72.5	79.0	0.973
Union (Li et al., 2018)	80.5	65.7	78.3	-
*APPNP (Klicpera et al., 2019)	83.2	71.7	79.7	-
SGC (Wu et al., 2019)	81.0	71.9	78.9	-
MixHop (Abu-El-Haija et al., 2019)	81.9	71.4	**80.8**	-
AdaLNet (Liao et al., 2019)	80.4	68.7	78.1	-
HGCN (Chami et al., 2019)	79.9	-	**80.3**	-
GESM (w/o att, reg)	82.8	71.7	**80.3**	0.753
GESM (w/o reg)	**84.4**	**72.6**	80.1	**0.976**
GESM	**84.5**	**72.7**	**80.4**	**0.974**

* Best experimental results from our own implementation

2.1. Step Mixture to Avoid Oversmoothing

Most graph neural networks suffer from oversmoothing as the propagation step gets deeper. Although JK-Net (Xu et al., 2018) handles oversmoothing by utilizing GCN blocks with mulitple propagation, it cannot completely resolve the oversmoothing issue as shown in Figure 2. We explicitly separate the node embedding and propagation process by employing a mixture of multiple random walk steps. This step mixture approach allows our model to alleviate the oversmoothing issue along with localized aggregation.

Our method consists of three stages. First, input X passes through a fully connected layer with a nonlinear activation, which outputs embedding node features $Z = \sigma(XW)$. Second, Z is multiplied by a normalized adjacency matrix $\hat{\hat{\boldsymbol{A}}}$ for each random walk step that is to be considered. As shown in the first and second stage, node embedding and propagation processes are separated. Finally, the concatenated result of each step f_{cat} is fed into the prediction layer. The entire propagation process can be formulated as:

$$f_{cat} = \Big\|_{k=0}^{s} \hat{\hat{\boldsymbol{A}}}^k Z, \tag{1}$$

where $||$ is the concatenation operation, s is the maximum number of steps considered for aggregation, and $\hat{\hat{\boldsymbol{A}}}^k$ is the normalized adjacency matrix $\hat{\hat{\boldsymbol{A}}}$ multiplied k times. As can be seen from Equation 1, weights are shared across nodes.

In our method, the adjacency matrix $\hat{\hat{\boldsymbol{A}}}$ is an asymmetric matrix, which is generated by *random walks* and flexible to arbitrary graphs. On the other hand, prior methods such as JK-Net (Xu et al., 2018) and MixHop (Abu-El-Haija et al., 2019), use a symmetric Laplacian adjacency matrix, which limits graph structures to given fixed graphs.

2.2. Neighborhood Interaction-based Attention

For more sophisticated design of node embedding Z, we adopt bilinear pooling-based neighborhood interaction as an attention mechanism so that node information is adaptively emphasized for aggregation. We simply replace Z in Equation 1 with attention features denoted by H_{multi}.

$$\text{output} = \text{softmax}((\Big\|_{k=0}^{s} \hat{\hat{\boldsymbol{A}}}^k H_{\text{multi}})W)). \tag{2}$$

As described by Equation 3, we employ multi-head attention, where H_{multi} and α_i denote the concatenation of m attention layers and the i-th attention coefficient. We only compute α for nodes $j \in \mathcal{N}_i$, the neighborhood nodes of node i, to maintain the structural representation of the graph. The attention coefficients α is calculated by the sum of outer products between encoding vectors of node i and its neighbor node j:

$$H_{\text{multi}} = \Big\|_{i=1}^{m} \sigma(\alpha_i Z_i)), \tag{3}$$

$$\alpha_i = \text{softmax}_j(\sum e_i \otimes e_j), \tag{4}$$

where e is a hadamard-product of node embedding Z and weight matrix W, *i.e.*, $e = Z \odot W$.

By incorporating attention to our base model, we can avoid or ignore noisy parts of the graph, providing a guide for random walk (Lee et al., 2018). In particular, datasets with the same structure but different node information can benefit from our method because these datasets can only be distinguished by node information. Therefore, focusing on node features for aggregation can provide more reliable results in inductive learning.

Table 2. Node classification results on datasets with low label rates.

	Cora		Citeseer		Pubmed
Method	1%	3%	0.5%	1%	0.1%
GCN (Kipf & Welling, 2017)	62.3	76.5	43.6	55.3	65.9
Union (Li et al., 2018)	**69.9**	78.5	46.3	59.1	70.7
*JK-GCN (Xu et al., 2018)	65.1	76.8	37.1	55.3	71.1
*APPNP (Klicpera et al., 2019)	67.6	**80.8**	40.5	59.9	70.7
*SGC (Wu et al., 2019)	64.2	77.2	41.0	58.1	71.7
AdaLNet (Liao et al., 2019)	67.5	77.7	**53.8**	**63.3**	**72.8**
GESM (w/o att, reg)	68.2	**81.6**	45.6	62.6	**73.0**
GESM (w/o reg)	**70.5**	**81.2**	**53.2**	**62.7**	**73.8**
GESM	**70.9**	80.8	**51.8**	**63.0**	**72.8**

* Best experimental results through our own implementation

Table 3. Average test set accuracy and standard deviation over 100 random train/validation/test splits with 20 runs.

	Coauthor CS	Coauthor Physics	Amazon Computers	Amazon Photo
MLP	88.3 ± 0.7	88.9 ± 1.1	44.9 ± 5.8	69.6 ± 3.8
GCN (Kipf & Welling, 2017)	91.1 ± 0.5	92.8 ± 1.0	**82.6** $\pm$ 2.4	**91.2** $\pm$ 1.2
GraphSAGE (Hamilton et al., 2017)	91.3 ± 2.8	93.0 ± 0.8	**82.4** $\pm$ 1.8	**91.4** $\pm$ 1.3
GAT (Veličković et al., 2018)	90.5 ± 0.6	92.5 ± 0.9	78.0 ± 19.0	85.7 ± 20.3
GESM (w/o att, reg)	**91.8** $\pm$ 0.4	**93.3** $\pm$ 0.6	79.2 ± 2.0	89.3 ± 1.9
GESM (w/o reg)	**91.5** $\pm$ 0.5	**93.4** $\pm$ 0.6	80.6 ± 2.1	89.8 ± 1.9
GESM	**91.4** $\pm$ 0.5	**93.5** $\pm$ 0.8	**80.8** $\pm$ 2.0	**90.3** $\pm$ 2.1

2.3. Embedding Regularization

For better representation learning, node embedding Z requires a regularization that helps neighbor nodes to be clustered and irrelevant nodes to become distant by reflecting the graph structure (Figure 4). Our push and pull based triplet regularization R can be formulated as:

$$R = \frac{1}{|S|} \sum_{\substack{p \in S \\ n \in S^c}} \left(\beta \cdot \text{Dis}(Z_c, Z_p) - (1-\beta) \cdot \text{Dis}(Z_c, Z_n) \right), \tag{5}$$

where $S \subset E$ and its cardinality $|S|$ is the number of samples, and β denotes a weight for the distance of positive and negative nodes. A positive node p represents the neighbor node of a center node c, and a negative node n, on the other hand, represents all nodes except the positive and center nodes. For the distance function $\text{Dis}(\cdot,\cdot)$, we used a sigmoid of dot products, *i.e.*, $\text{Dis}(Z_i, Z_j) = 1 - \text{sigmoid}(Z_i^\top Z_j)$.

Finally, our overall loss $\mathcal{L}$ is $\mathcal{L} = J + R$, where J denotes softmax cross-entropy loss, and R denotes the regularizer.

3. Results

3.1. Node classification

Results on benchmark datasets. Table 1 summarizes the comparative experiments for transductive and inductive learning tasks. In general, most methods limit their domain to one type of learning tasks, and it is rare to have satisfactory results on both domains. Our methods, however, are ranked in the top-3 for each task with large predictive power.

For transductive learning tasks, our model GESM (w/o reg) outperforms many existing baseline models. These results indicate the significance of considering both global and local information using the attention mechanism. It can also be observed that GESM yielded more stable results than GESM (w/o reg), suggesting the importance of reconstructing structural node representation in the aggregation process.

For the inductive learning task, our proposed model GESM (w/o reg) and GESM surpass the results of GAT, despite the fact that GAT consists of more attention layers. These results for unseen graphs are in good agreement with results shown by Veličković et al. (2018), in which enhancing the influence of node information improved generalization.

Results on datasets with low label rates. To demonstrate the importance of global information, we experimented on sparse datasets with low label rates. As can be seen in Table 2, our models show remarkable performance even on the dataset with very few labels. These results indicate that our methods can adaptively select node information from local to global neighborhoods.

Experiments for robustness. For an in-depth verification of overfitting, we extended our experiments to four additional node classification datasets: Coauthor CS, Coauthor Physics,

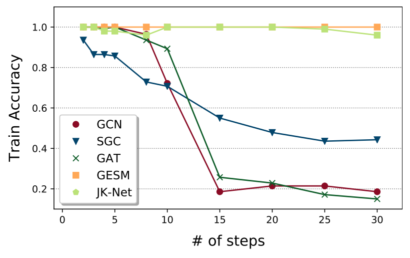


Figure 1. Training accuracy as the number of propagation step increases.

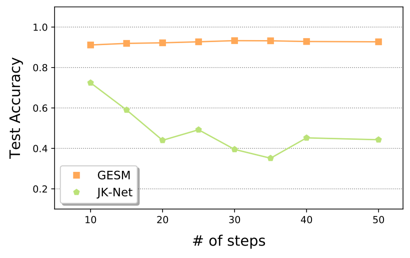


Figure 2. Test accuracy of JK-Net and GESM after the 10th step.

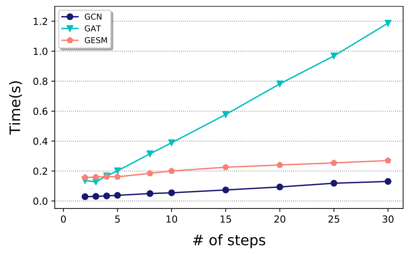


Figure 3. Inference time as the step size increases on Cora.

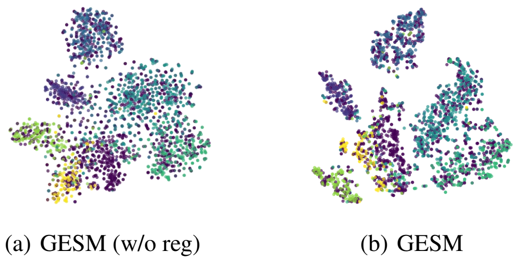
Amazon Computers and Amazon Photo. We followed the experimental setup of Shchur et al. (2018) including the hyperparameter settings. The results in Table 3 prove that our proposed methods do not overfit to a particular dataset. Moreover, in comparison to GAT, the performance of GESM is more accurate and stable.

3.2. Model Analysis

Oversmoothing and Accuracy. As shown in Figure 1, GCN (Kipf & Welling, 2017), SGC (Wu et al., 2019), and GAT (Veličković et al., 2018) suffer from oversmoothing. GCN and GAT show severe degradation in accuracy after the 8th step; The accuracy of SGC does not drop as much as GCN and GAT but nevertheless gradually decreases as the step size increases. The proposed GESM, unlike the others, maintains its performance without any degradation, because no rank loss (Luan et al., 2019) occurs and oversmoothing is overcome by step mixture.

Interestingly, JK-Net (Xu et al., 2018) also maintains its training accuracy regardless of the step size by using GCN blocks with multiple steps according to Figure 1. We further compared the test accuracy of GESM with JK-Net, a similar approach to our model, in regards to step sizes. To investigate the adaptability to larger steps of GESM and JK-Net, we concatenated features after the 10th step. As shown in Figure 2, GESM outperforms JK-Net, even though both methods use concatenation to alleviate the oversmoothing issue. These results are in line with the fact that JK-Net obtains global information similar to GCN or GAT. Consequently, the larger the step, the more difficult it is for JK-Net to maintain performance. GESM, on the other hand, maintains a steady performance, which confirms that the accuracy of GESM does not collapse even for large step sizes.

Inference time. As shown in Figure 3, the computational complexity of all models increases linearly as the step size increases. We can observe that the inference time of GCN (Kipf & Welling, 2017) is slightly faster than that of GESM by a constant margin. GESM is much faster than GAT (Veličković et al., 2018) in terms of inference time, while providing higher and stable results as shown in Table 3. Our methods are both fast and accurate due to the mixture of random walk steps.



(a) GESM (w/o reg) (b) GESM

Figure 4. t-SNE plot of the last hidden layer trained on Cora.

Embedding visualization. Figure 4 visualizes the hidden features with the t-SNE algorithm (Maaten & Hinton, 2008). The figure illustrates the difference between GESM (w/o reg) and GESM. While the nodes are scattered for GESM (w/o reg), they are more closely clustered for GESM. According to the results in Table 1, more closely clustered GESM generally produces better results than loosely clustered GESM (w/o reg).

4. Conclusion

Traditional graph neural networks suffer from the oversmoothing issue for a large number of propagation steps as well as poor generalization on unseen graphs. To tackle these issues, we propose a simple but effective model that weights differently depending on node information in the aggregation process and adaptively considers global and local information by employing the mixture of multiple random walk steps. To further refine the graph representation, we have presented a new regularization term, which enforces similar neighbor nodes to be closely clustered in the node embedding space. The results from extensive experiments on eight benchmark graph datasets show that our GESM successfully achieves state-of-the-art or competitive performance for both transductive and inductive learning tasks.

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Harutyunyan, H., Alipourfard, N., Lerman, K., Steeg, G. V., and Galstyan, A. Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing. *International Conference on Machine Learning (ICML)*, 2019.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 4869–4880, 2019.

Gordo, A., Almazan, J., Revaud, J., and Larlus, D. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124 (2):237–254, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *International Conference on Learning Representations (ICLR)*, 2019.

Lee, J. B., Rossi, R. A., Kim, S., Ahmed, N. K., and Koh, E. Attention models in graphs: A survey. *arXiv preprint arXiv:1807.07984*, 2018.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Liao, R., Zhao, Z., Urtasun, R., and Zemel, R. S. Lanczosnet: Multi-scale deep graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2019.

Luan, S., Zhao, M., Chang, X.-W., and Precup, D. Break the ceiling: Stronger multi-scale deep graph convolutional networks. *Advances in neural information processing systems*, 2019.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

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

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.

Wu, F., Zhang, T., Souza Jr, A. H. d., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. *International Conference Machine Learning (ICML)*, 2019.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. *International Conference on Machine Learning (ICML)*, 2018.