A SIMPLE GRAPH NEURAL NETWORK VIA LAYER SNIFFER

*Dingyi Zeng[1], Li Zhou[2], Wanlong Liu, Hong Qu, Wenyu Chen**

School of Computer Science and Engineering, University of Electronic Science and Technology of China

ABSTRACT

Due to the success of Graph Neural Networks(GNNs) in graph-structure data, many efforts have been devoted to enhancing the propagation ability and alleviating the over-smoothing problem of GNNs. However, from the perspective of closeness extent of node representations, most existing GNNs pay less attention to the attributes of node representation space. In light of this, we design a Layer Sniffer module that can combine the effects of the local node-level representation closeness extent and the global layer-level information attention. On this basis, we propose a simple Layer Sniffer Graph Neural Network (LSGNN) with a propagation scheme that can fuse neighborhood information of different receptive fields densely and adaptively. Our extensive experiments on three public node classification datasets demonstrate the superior performance and stability of our proposed model.

***Index Terms*—** Graph Learning, Semi-supervised Learning, Graph Neural Networks, Graph Signal Processing

1. INTRODUCTION

Inspired by the vanilla Convolutional Neural Network (CNN) [1], Graph Convolutional Network(GCN) [2] and subsequent Graph Neural Networks(GNNs) [3] are proposed to apply analogous convolutional operation to graph-structure data. GNNs have achieved state-of-the-art results in various application areas, including but not limited to citation networks [2], social networks [4], applied chemistry [5] and natural language processing [6].

Most models based on graph convolution and its approximation are very shallow and achieve the best performance in two layers, which limits their ability to obtain multi-hop neighborhood information. To expand the receptive field of neighborhood information, stacking multiple layers becomes an option. But in this way, the amount of receptive information increases exponentially and complex spatial structure information is compressed into a narrow vector space, which leads nodes to losing their distinguishability, resulting in the over-smoothing problem [7] eventually. As shown in Fig.1, the node representation vectors with high Conicity value generated by deeper GNNs are converged in a narrow space, which causes worse classification performance.

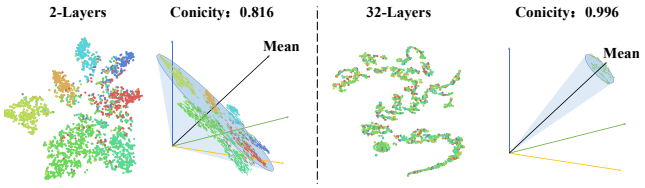

Fig. 1. The t-SNE visualization and the corresponding Conicity value of the node representation vectors derived from the different layer GCN model on the Cora dataset. The right part indicates the case when representation vectors generated by deeper GCN lie in narrow space resulting in a high Conicity value, while the left part shows the case when two-layer GCN's representation vectors are spread out with a lower Conicity value.

Most methods [8–10] focus on the structural design of neural networks with less attention to the attributes of node representation space.

In order to learn distinguishable node representations, we propose a propagation scheme based on Layer Sniffer, which concentrates on each hidden node representation space and can densely aggregate neighborhood information of different receptive fields while enhancing information acquisition ability. We utilize this propagation procedure to construct a simple model, a simple Graph Neural Network via Layer Sniffer (LSGNN) with lower complexity and less learnable parameters. In summary, our main contributions are as follows:

- We design Layer Sniffer with conicity coefficient and layer-attention score, which emphasizes the extent of local node-level representation closeness in a single layer and global layer-level information attention in all layers.
- We propose a Layer Sniffer Graph Neural Network (LSGNN) based on the propagation scheme that can aggregate neighborhood information of different receptive fields densely and adaptively.
- We conduct extensive experiments on three public node classification datasets and the results demonstrate the superiority of our model that the proposed model has stable performance under different depth setups.

[1][2]Equal contributation

*Corresponding author: Wenyu Chen, email: cwy@uestc.edu.cn

2. PRELIMINARIES AND RELATED WORK

Notations. An undirected graph with self-loop edges is defined as $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, in which n nodes $\mathrm{v}_i \in \mathrm{V}$, m edges $(\mathrm{v}_i, \mathrm{v}_j) \in \mathrm{E}$. $\mathbf{A}$ denotes the adjacency matrix of G, $\mathbf{A}_{ij} = 1$ indicating an existing edge between node i and node j. And $\mathbf{D}$ is the diagonal degree matrix of G, where the element $\mathbf{D}_{ii}$ represents the degree of node i. $\mathbf{X} \in \mathbf{R}^{n \times d}$ denotes the node feature matrix, in which $\mathbf{X}_i$ is associated with a d-dimensional feature vector of node i.

Vanilla GCN. The vanilla GCN [2] is defined as:

$$\mathbf{H}^{l+1} = \sigma\left(\mathbf{P}\mathbf{H}^{l}\mathbf{W}^{l}\right), \tag{1}$$

where σ denotes the ReLU activation function, $\mathbf{P}$ is the symmetric normalization of the adjacency matrix, expressed as $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$.

JKNet. JKNet [9] is the first deep GCN framework that aggregates information from multiple layers. The output of JKNet is expressed as $\mathrm{f}\left(\mathbf{H}^0, \mathbf{H}^1, \ldots, \mathbf{H}^{L-1}\right)$. This model applys a variety of mechanisms to integrate information from all layers, including **Concatenation** , **Max-pooling** , and **LSTM-attention** .

DropEdge. After in-depth research on multiple models, DropEdge [10] suggests that randomly removing some edges from the graph retards the convergence rate of over-smoothing. Let $\mathbf{P}^{Drop}$ denote the renormalized graph convolution matrix with some edge removed at random, the vanilla GCN equipped with DropEdge is defined as:

$$\mathbf{H}^{l+1} = \sigma\left(\mathbf{P}^{Drop}\mathbf{H}^{l}\mathbf{W}^{l}\right), \tag{2}$$

Other Related Work. Besides, many efforts focus on attention-based GCN models [8,11,12] , which learn the edge weights at each layer based on node features. Zhou [13] proposed a weighted GCN by constructing a logical adjacency matrix, which can fuse multi-hop information in a single layer. And Mix-hop [14] repeatedly mixes feature representations of neighbors at various distances, which proves the importance of neighbor information at different distances.

3. THE PROPOSED METHOD

Fig.2 illustrates a propagator layer block that is the critical component of our proposed LSGNN. The whole propagator network comprises L layers, each of which is implemented as $P(\cdot)$. This is a composite function of operations including **Layer Sniffer** and **Information Aggregator** , which can initiatively measure the uniqueness of node representation in each layer and obtain the new layer representation by fusing multi-layer feature information. We denote the output of the l^{th} layer as $\mathbf{H}^l = \{\mathrm{h}_0^l, \mathrm{h}_1^l, \ldots, \mathrm{h}_{n-1}^l\} \in \mathrm{R}^{n \times d}$. Specifically, the $(l+1)^{th}$ layer receives graph structure and the hidden node representations of all preceding layers as input:

$$\mathbf{H}^{l+1} = P(\mathbf{A}, \mathbf{H}^0, \mathbf{H}^1, \ldots, \mathbf{H}^l), \tag{3}$$

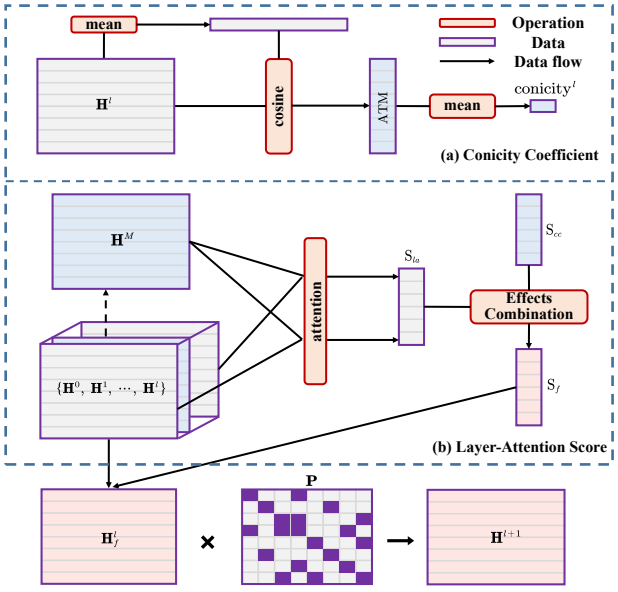

Fig. 2. The critical component of LSGNN. The upper part is **Layer Sniffer** , which is inside of dotted box, while the bottom part is **Information Aggregator** .

$$\mathbf{H}^0 = \mathbf{W}_h\mathbf{X}, \tag{4}$$

where $\mathbf{W}_{\mathrm{h}} \in \mathrm{R}^{\mathrm{d} \times \mathrm{d_h}}$ is a trainable parameter, which transfers the original node feature to a low-dimensional representation space. Further decomposition operations of $P(\cdot)$ will be described in detail below.

3.1. Layer Sniffer

Layer Sniffer mainly consists of two parts: conicity coefficient and layer-attention score, which can emphasize the extent of local node-level representation closeness and global layer-level information attention simultaneously.

Conicity Coefficient. Inspired by the potential factor of the over-smoothing problem that the learned node representations are too similar in GNNs, leading nodes indistinguishable, we introduce $Conicity$ [15] to quantitatively measure the similarity between node representations in $\mathbf{H}^l$.

Firstly, we compute the alignment to mean (ATM) of a node feature vector h_i^l, which quantifies the distance between each node vector and the whole node representation space center. The specific definition is as follows:

$$\mathrm{ATM}(\mathrm{h}_i^l, \mathbf{H}^l) = \mathrm{cosine}(\mathrm{h}_i^l, \frac{1}{n}\sum_{j=0}^{n-1}\mathrm{h}_j^l), \tag{5}$$

where the function $\mathrm{cosine}(\mathrm{x}, \mathrm{y}) = \frac{\mathrm{x}^{\mathrm{T}} \cdot \mathrm{y}}{\|\mathrm{x}\| \cdot \|\mathrm{y}\|} \in [-1, 1]$. And $\mathrm{Conicity}^l$ of node representation space $\mathbf{H}^l$ is defined as the

Table 1. Summary of classification accuracy(%) results on Cora, Citeseer, and Pubmed. The number in parentheses corresponds to the number of layers of the model.

Method	Cora	Citeseer	Pubmed
GCN	81.5	71.1	79.0
GCN(Drop)	82.8	72.3	79.6
GAT	83.1	70.8	78.5
GLCN	82.1	72.2	77.8
GFCN	83.7	**73.7**	79.1
Scattering GCN	84.2	71.7	79.4
Incep	81.7 (32)	70.2 (16)	77.9 (8)
Incep(Drop)	83.5 (64)	72.7 (4)	79.5 (4)
JKNet	81.1 (4)	69.8 (16)	78.1 (32)
JKNet(Drop)	83.3 (4)	72.7 (16)	79.5 (32)
LSGNN	**85** $\pm$0.6(8)	72.8 $\pm$0.8(8)	**79.7** $\pm$0.4(8)

mean ATM of all node vectors in $\mathbf{H}^l$:

$$\text{Conicity}^l = \frac{1}{n}\sum_{j=0}^{n-1}\text{ATM}(\text{h}_j^l, \mathbf{H}^l). \tag{6}$$

The high Conicity^l value of node representation vectors $\mathbf{H}^l$ indicates that the node vectors in $\mathbf{H}^l$ are closely aligned with their mean (i.e. the representation space center), which leads to nodes indistinguishable. Naturally, we desire to obtain the node representation vectors with low Conicity, in which each node vector gets an apparent discrepancy, contributing to alleviating the over-smoothing problem. So We apply conicity coefficient, a variant of Conicity, to quantify the extent of local node-level representation closeness extent in each layer, which is defined as follows:

$$\text{S}_{CC}^l = 1 - \text{Conicity}^l. \tag{7}$$

Layer-Attention Score. The conicity coefficient focuses on the local similarity of node-level representation vectors in the current layer. Furthermore, to appraise the global attention of layer-level representation in all layers, we apply a layer-attention mechanism, which is defined as:

$$CAT(\mathbf{H}^l) = [\text{h}_0^l : \text{h}_1^l : \cdots : \text{h}_{n-1}^l] \in R^{n*d}, \tag{8}$$

$$\text{S}_{LA}^l = \text{CAT}(\mathbf{H}^M) \cdot \text{CAT}(\mathbf{H}^l)^T, \tag{9}$$

where $\mathbf{H}^M$, as the query, is the hidden node representation space with the max S_{CC} in $\{\mathbf{H}^0, \mathbf{H}^1, \cdots, \mathbf{H}^{l-1}\}$.

Effects Combination. From local to global, the conicity coefficient S_{CC}^k and the layer-attention score S_{LA}^k of all preceding layers ($k \in \{0, 1, \cdots, l\}$) are effectively combined to more comprehensively evaluate the effect of preceding layers, which is defined in detail as follows:

$$S_C^k = S_{CC}^k \cdot S_{LA}^k, \tag{10}$$

Table 2. Summary of classification accuracy(%) results with various depths.

Dataset	Method	layers 2	4	8	16	32	64
Cora	GCN	81.1	80.4	64.9	64.9	60.3	28.7
	GCN(Drop)	**82.8**	82.0	75.8	75.7	62.5	49.5
	JKNet	-	80.2	80.7	80.2	81.1	71.5
	JKNet(Drop)	-	83.3	82.6	83.0	82.5	83.2
	Incep	-	77.6	76.5	81.7	81.7	80.0
	Incep(Drop)	-	82.9	82.5	83.1	83.1	83.5
	LSGNN	-	**84.3**	**85**	**84.9**	**84.7**	**84.4**
Citeseer	GCN	70.8	67.6	30.2	18.3	25.0	20.0
	GCN(Drop)	**72.3**	70.6	61.4	57.2	41.6	34.4
	JKNet	-	68.7	67.7	69.8	68.2	63.4
	JKNet(Drop)	-	72.6	71.8	**72.6**	70.8	**72.2**
	Incep	-	69.3	68.4	70.2	68.0	67.5
	Incep(Drop)	-	**72.7**	71.4	72.5	**72.6**	71.0
	LSGNN	-	72.3	**72.8**	72.5	72.2	72
Pubmed	GCN	79.0	76.5	61.2	40.9	22.4	35.3
	GCN(Drop)	**79.6**	79.4	78.1	78.5	77.0	61.5
	JKNet	-	78.0	78.1	72.6	72.4	74.5
	JKNet(Drop)	-	78.7	78.7	79.1	79.2	78.9
	Incep	-	77.7	77.9	74.9	OOM	OOM
	Incep(Drop)	-	**79.5**	78.6	79.0	OOM	OOM
	LSGNN	-	78.9	**79.7**	**79.4**	**79.3**	**79.1**

$$\widetilde{S}_C^k = \frac{S_C^k}{\max(S_C^0, S_C^1 ... S_C^l)}, \tag{11}$$

$$S_f^k = \frac{\exp(\widetilde{S}_C^k)}{\sum_{k=0}^{l}\exp(\widetilde{S}_C^k)}. \tag{12}$$

3.2. Information Aggregator

The information aggregator is constitutive of global aggregation and neighborhood propagation, which can enhance the information acquisition ability of our proposed model.

Global Aggregation. Different from most GNNs that directly adopt the upper-layer output for representation transformation and information propagation, we firstly aggregate the node representations of all the preceding hidden layers adaptively based on layer sniff, which improves the information flow between layers:

$$\mathbf{H}_f^l = \sum_{\text{k=0}}^{l} S_f^k \mathbf{H}^k. \tag{13}$$

Neighborhood Propagation. The final step that performs neighborhood information propagation can contribute to expanding the receptive field of model and obtaining high-hop

information in the graph:

$$\mathbf{H}^{l+1} = \mathbf{P}\mathbf{H}_f^l. \tag{14}$$

After repeatedly iterating the propagator layer block, the node representations with rich information can be obtained.

4. EXPERIMENTS

4.1. Experimental Setup

Dataset. We run our experiments on three standard citation network datasets Cora, Citeseer, and Pubmed [16] for semi-supervised node classification. These datasets are split in the usual standard way [17], specifically 20 nodes per class for training, 500 nodes for validation and 1,000 nodes for testing.

Baselines. For baselines, we include GCN [2], GLCN [18], GAT [8], JKNet [9], Scattering GCN [19], GFCN [20], IncepGCN [10] and DropEdge [10]. And we equip DropEdge on three backbones : GCN [2], JKNet [9] and IncepGCN [10] for further comparison.

Settings. We use ADAM optimizer [21] to train our LSGNN with a learning rate of 0.01 for 1500 epochs and apply early stopping with a patience of 100 epochs. Also we perform a grid search to tune the hyper-parameters for all models with different depths.

4.2. Results and Analysis

Results. For fair comparison, we reuse the best metrics already reported in Fey & Lenssen [22] for GCN, GAT, Jiang [18] for GLCN and the best metrics reported in Rong [10] for GCN(Drop), JKNet, JKNet(Drop), Incep and Incep(Drop). Table 1 reports the mean classification accuracy with the standard deviation on the test nodes of our model and other preceding shallow and deep models after 100 runs. The results show that LSGNN outperforms all compared state-of-the-art methods across all three datasets with the help of multi-level information brought by the Layer Sniffer and the efficient propagation scheme of the LSGNN.

Depth Study. Table 2 summarises the test results for different models with various depths which demonstrates our model has the state-of-the-art performance in most depths, while not having significant performance degradation at deeper depths like other models. By dynamically acquiring the information of different receptive fields, our model can obtain the multi-hop information but not encounter the over-smoothing problem caused by excessive information. For further comparison, when LSGNN is stacking layers, the amount of parameters will not rise and always remain the same as the two-layer GCN. So our model can obtain the multi-hop information without additional computational burden.

Over-smoothing Analysis. As shown in Fig.3, the representation vectors of the deeper layers are more clustered,

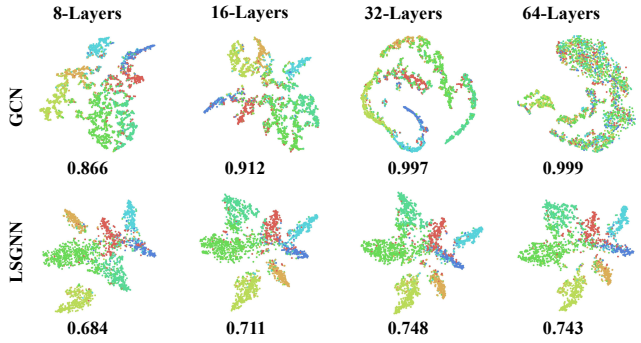

Fig. 3. The t-SNE visualization and the corresponding Conicity value of the node representation vectors derived from the GCN model and the LSGNN model with different layers on the Cora dataset.

the Conicity value also increases, and the performance of the GCN model continues to decline. Compared with GCN, the representation vectors generated by our model are less clustered and also have a lower Conicity value, corresponding to our model having a better performance. Through Fig.3, we intuitively demonstrate the difference between over-smoothed and non-over-smoothed representation vectors, that is, as the over-smoothing problem becomes more serious, the representation vectors become more clustered and difficult to distinguish, and prove that a higher Conicity value corresponds to a more serious over-smoothed representation vector.

5. CONCLUSION

In this paper, from the perspective of closeness extent of node representations, we propose the Layer Sniffer module that can combine the effects of local node-level and global layer-level information. Through this we further propose a simple Layer Sniffer Graph Neural Network (LSGNN) with a propagation scheme to densely and adaptively fuse different receptive fields neighborhood information. Our extensive experiments on three public node classification datasets demonstrate the superior performance and stability of our proposed model. We further study the performance of different models at different depths and demonstrated the superior depth stability of our proposed model. Through visualization experiments, we intuitively demonstrate the difference between over-smoothed and non-over-smoothed representation vectors, and prove the direct correlation between high Conicity and the over-smoothing problems which provides a technical basis for the quantitative analysis of the over-smoothing problem. In future work, we want to further explore the boundaries of Graph Neural Network's information acquisition capabilities.

6. REFERENCES

[1] Yann LeCun, Yoshua Bengio, et al., "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, pp. 1995, 1995.

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

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *arXiv preprint arXiv:1606.09375*, 2016.

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

[5] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S Zemel, "Lanczosnet: Multi-scale deep graph convolutional networks," *arXiv preprint arXiv:1901.01484*, 2019.

[6] Yuhao Zhang, Peng Qi, and Christopher D Manning, "Graph convolution over pruned dependency trees improves relation extraction," *arXiv preprint arXiv:1809.10185*, 2018.

[7] Qimai Li, Zhichao Han, and Xiao-Ming Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, vol. 32.

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[9] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.

[10] 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.

[11] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li, "Attention-based graph neural network for semi-supervised learning," *arXiv preprint arXiv:1803.03735*, 2018.

[12] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," *arXiv preprint arXiv:1803.07294*, 2018.

[13] Li Zhou, Tingyu Wang, Hong Qu, Li Huang, and Yuguo Liu, "A weighted gcn with logical adjacency matrix for relation extraction," in *European Conference on Artificial Intelligence*, 2020.

[14] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *international conference on machine learning*. PMLR, 2019, pp. 21–29.

[15] Aditya Sharma, Partha Talukdar, et al., "Towards understanding the geometry of knowledge graph embeddings," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 122–131.

[16] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

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

[18] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11313–11320.

[19] Yimeng Min, Frederik Wenkel, and Guy Wolf, "Scattering gcn: Overcoming oversmoothness in graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14498–14508, 2020.

[20] Feng Ji, Jielong Yang, Qiang Zhang, and Wee Peng Tay, "Gfcn: A new graph convolutional network based on parallel flows," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3332–3336.

[21] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] Matthias Fey and Jan Eric Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.