Attributed Random Walk as Matrix Factorization

Lei Chen*
Courant Institute of Mathematical Sciences
New York University, USA
lc3909@nyu.edu

Shunwang Gong
Department of Computing
Imperial College London, UK
shunwang.gong16@imperial.ac.uk

Joan Bruna
Courant Institute of Mathematical Sciences
Center for Data Science
New York University, USA
bruna@cims.nyu.edu

Michael M. Bronstein
Department of Computing
Imperial College London, UK
m.bronstein@imperial.ac.uk

Abstract

Mainstream random walks on graphs mostly focus on the topology while ignoring node attributes. In this paper, we develop a matrix form of the attributed random walk with pointwise mutual information in an unsupervised fashion. We show through experiments that the generated embeddings of flexible dimensions are robust to label missing on the transductive node classification task.

1 Introduction

Many essential tasks in network analysis involve predictions over nodes and edges with information of connections and features. Node embedding methods[13, 16, 4] along with graph neural networks [1, 9, 3] have been developed to combine such information. However, mainstream random walks, such as DeepWalk [13], Line [16], or node2vec [4], do not take feature information into account.

Recently, Huang *et al.* [7] propose a sampling-based random walk method enhanced with node attributes. In their approach, sequences with features are sampled with the transition probability of attributed random walk and fed into a downstream RNN. Their end-to-end attributed random walk without RNN outperforms baselines on social network datasets, including embedding methods, GCN [9] and GraphSAGE [5].

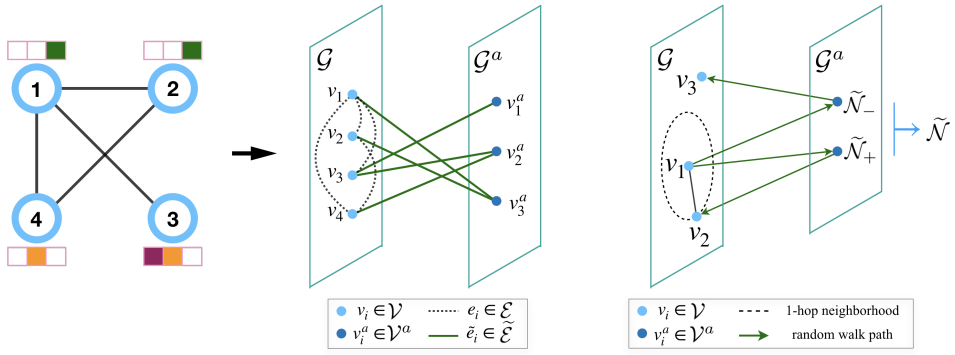
In this paper, we provide further insights into the attributed random walk model by connecting it with the matrix factorization version of DeepWalk, NetMF [14]. Then we propose a matrix factorization form of the attributed random walk using Skip-Gram with Negative Sampling, leading to increased performance.

Skip-gram with Negative Sampling (SGNS). In the context of word embedding, SGNS is to maximize embedding similarities between observed word-context pairs and to minimize similarities between randomly sampled pairs [12]. Levy and Goldberg [10] prove that SGNS is implicitly factorizing $\log\left(\frac{\#(w,c)|\mathcal{D}|}{\#(w)\#(c)}\right) - \log b$, where the "*corpus*" $\mathcal{D}$ is a multiset that counts the multiplicity of word-context pairs; $\#(w,c), \#(w)$ and $\#(c)$ denote the number of times word-context pair (w,c), word w and context c appear in the corpus; b is the number of negative samples. A detailed explanation of "*word*" and "*context*" in a graph, along with introduction to DeepWalk and NetMF, is in Appendix A.

Attributed Random Walk (AttrRW). Huang *et al.* [7] propose an attributed random walk model building an attribute graph $\mathcal{G}^a$ in addition to the original graph $\mathcal{G}$. They define each node in $\mathcal{G}^a$

*Source code: https://github.com/leichen2018/AttrRW_Matrix_Factorization

Graph Representation Learning Workshop NeurIPS 2019, Vancouver, Canada.



(a) input graph and the attribute graph

(b) partition of non-zero feature set $\tilde{\mathcal{N}}$

Figure 1: *Left*: Input graph of 4 nodes with 3-dimensional features. Colored squares denote non-zero features, then we generate connections to the corresponding feature vertices on the attribute graph. *Right*: The non-zero feature set $\tilde{\mathcal{N}}$ of each node is partitioned into *local* $\tilde{\mathcal{N}}_+$ and *non-local* $\tilde{\mathcal{N}}_-$, details of which are in Section 3.

corresponding to a feature channel, then establish undirected connections $\{(u, v^a)|u \in \mathcal{G}, v^a \in \mathcal{G}^a\}$ if and only if u has a positive value in the feature channel of v^a. After sampling fixed-length paths starting from each node according to the random walk transition probability matrix, they average features of all observed nodes on the paths to generate the embedding of the starting node.

2 Matrix Factorization Based Attributed Random Walk

In this section, we formulate the attributed random walk with definition of transition probability matrix, and then present the closed-form target matrix with pointwise mutual information.

2.1 Attributed Random Walk

We consider an undirected graph $\mathcal{G} = (\mathcal{V} = \{1, \ldots, N\}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ with N nodes, edges $(i, j) \in \mathcal{E}$, an $N \times N$ adjacency matrix $\mathbf{A}$ (with $\mathbf{A}_{i,j} > 0$ if $(i, j) \in \mathcal{E}$ and zero otherwise), a degree matrix $\mathbf{D} = \text{diag}\left(\sum_{\ell=1}^{N} \mathbf{A}_{i,\ell}\right)$, and a *nonnegative* $N \times F$ node attribute matrix $\mathbf{X}$, where F is the dimension of node features. Typically, $\mathbf{A}$ and $\mathbf{X}$ are sparse matrices. [7] propose modeling node attributes as an *attribute graph* $\mathcal{G}^a = (\mathcal{V}^a = \{1, \ldots, F\}, \mathcal{E}^a = \varnothing)$; the two graphs $\mathcal{G}$ and $\mathcal{G}^a$ are connected with *inter-graph edge* set $\tilde{\mathcal{E}}$ containing an undirected edge (i, j) iff $\mathbf{X}_{i,j} > 0, j \in \mathcal{V}^a$. Hence, the whole graph is defined as $\tilde{\mathcal{G}} = (\mathcal{G}, \mathcal{G}^a, \tilde{\mathcal{E}}, \tilde{\mathbf{A}})$ with a new weighted matrix $\tilde{\mathbf{A}}$ defined in Equation 2.

We define the attributed random walk in the following way, as illustrated in Figure 1: 1) from a vertex in $\mathcal{G}$, the next step of the walk has a probability of α to stay in $\mathcal{G}$, $1 - \alpha$ to switch into $\mathcal{G}^a$; 2) from a vertex in $\mathcal{G}^a$, the next step goes backs to $\mathcal{G}$ with probability 1; 3) the conditional probability of walk from $i \in \mathcal{V}$ to $j \in \mathcal{V}^a$ is proportional to $\mathbf{X}_{i,j}$, which means $\text{Prob}(j|j \in \mathcal{V}^a, i) \propto \mathbf{X}_{i,j}$. We denote the row-wise normalized feature matrix as $\overline{\mathbf{X}} = (\overline{\mathbf{X}}_{i,j}) = (\mathbf{X}_{i,j} / \sum_l \mathbf{X}_{i,l})$. Hence, the transition probabilities are $\text{Prob}(j \in \mathcal{V}|i \in \mathcal{V}) = \alpha \cdot \mathbf{A}_{i,j}/\mathbf{D}_{i,i}$, $\text{Prob}(j \in \mathcal{V}^a|i \in \mathcal{V}) = (1 - \alpha)(\mathbf{D}_{i,i}\overline{\mathbf{X}}_{i,j})/(\sum_k \mathbf{D}_{i,i}\overline{\mathbf{X}}_{i,k}) = (1 - \alpha) \cdot \overline{\mathbf{X}}_{i,j}$, $\text{Prob}(j \in \mathcal{V}|i \in \mathcal{V}^a) = \mathbf{D}_{j,j}\overline{\mathbf{X}}_{j,i}/(\sum_k \mathbf{D}_{k,k}\overline{\mathbf{X}}_{k,i})$. The matrix form is:

$$\widetilde{\mathbf{P}} = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}} \in \mathbb{R}^{(N+F)\times(N+F)}, \tag{1}$$

$$\widetilde{\mathbf{A}} = \begin{bmatrix} \alpha\mathbf{A} & (1-\alpha)\mathbf{D}\overline{\mathbf{X}} \\ (1-\alpha)\overline{\mathbf{X}}^\top\mathbf{D} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(N+F)\times(N+F)}, \widetilde{\mathbf{D}} = \text{diag}\left(\sum_{j=1}^{N+F} \widetilde{\mathbf{A}}_{i,j}\right) \tag{2}$$

Note that: 1) $\widetilde{\mathbf{A}}$ is symmetric, with a different lower left block from that in [7]; 2) the sum of each row of $\widetilde{\mathbf{P}}$ is 1; 3) in the first N rows of $\widetilde{\mathbf{P}}$, the sum of the left N entries in each row is α while the sum of the right F entries is $1 - \alpha$. Hence, $\widetilde{\mathbf{P}}$ is a transition probablity matrix of attribute random walk with probability of $(1 - \alpha)$ jumping from $\mathcal{G}$ to $\mathcal{G}^a$; 4) in this paper, α is set as 0.5, which is experimentally optimal suggested in [7].

2.2 Matrix Factorization of Pointwise Mutual Information

With transition probablity matrix $\widetilde{\mathbf{P}}$, now we add SGNS into attributed random walk. Consider an *observed* vertex-context pair (v, c) with $v, c \in \mathcal{V} \cup \mathcal{V}^a$. SGNS's objective for a single (v, c) observation is then

$$\max \quad \log \sigma(\mathbf{H}_v \cdot \mathbf{H'}_c) + k \cdot \mathbb{E}_{c_N \sim P_\mathcal{D}}[\log \sigma(-\mathbf{H}_v \cdot \mathbf{H'}_{c_N})], \tag{3}$$

where $\sigma(x \cdot y) = 1/(1+e^{-x \cdot y})$, $P_\mathcal{D}(c) = \#(c)/|\mathcal{D}|$ as randomly sampling k negative contexts from collection $\mathcal{D}$ of empirical observed vertex-context pairs. $\mathbf{H}$, $\mathbf{H'}$ are node and context representations we are seeking.

Proposition 1. *The objective in Equation 3 is equivalent to factorizing*

$$\mathbf{H}_v \cdot \mathbf{H'}_c = \log \left(\frac{\#(v, c) \cdot |\mathcal{D}|}{\#(v) \cdot \#(c)} \right) - \log k. \tag{4}$$

The proof is the same as that in (Section 3.1, [10]). The first $\log(\cdot)$ term in RHS is known as Pointwise Mutual Information of (v, c).

Proposition 2. *Denote the length of each random walk by L, the context window size by T, use vol$(\cdot)$ to denote summing of all entries in the matrix, and set $d_i = \widetilde{\mathbf{D}}_{i,i}$. When $L \to \infty$, it converges in probability as*

$$\frac{\#(v, c) \cdot |\mathcal{D}|}{\#(v) \cdot \#(c)} \xrightarrow{p} \frac{vol(\widetilde{\mathbf{A}})}{2T} \left(\frac{1}{d_c} \sum_{r=1}^{T} (\widetilde{\mathbf{P}}^r)_{v,c} + \frac{1}{d_v} \sum_{r=1}^{T} (\widetilde{\mathbf{P}}^r)_{c,v} \right). \tag{5}$$

The proof is the same as that in (Theorem 2.3, [14]). As a consequence, since $\widetilde{\mathbf{A}}$ is symmetric and $\widetilde{\mathbf{P}} = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}$, we derive the following matrix form of Equation 5's RHS

$$\begin{aligned} &\frac{\text{vol}(\widetilde{\mathbf{A}})}{2T} \left(\sum_{r=1}^{T} \widetilde{\mathbf{P}}^r \widetilde{\mathbf{D}}^{-1} + \sum_{r=1}^{T} \widetilde{\mathbf{D}}^{-1} (\widetilde{\mathbf{P}}^r)^\mathsf{T} \right) = \text{vol}(\widetilde{\mathbf{A}}) \left(\frac{1}{T} \sum_{r=1}^{T} \widetilde{\mathbf{P}}^r \right) \widetilde{\mathbf{D}}^{-1} \\ &= \text{vol}(\widetilde{\mathbf{A}}) \widetilde{\mathbf{D}}^{-1/2} \left(\frac{1}{T} \sum_{r=1}^{T} (\widetilde{\mathbf{D}}^{-1/2} \widetilde{\mathbf{A}} \widetilde{\mathbf{D}}^{-1/2})^r \right) \widetilde{\mathbf{D}}^{-1/2} \triangleq \mathbf{M}. \end{aligned} \tag{6}$$

Therefore, we are explicitly factorizing $\mathbf{H}\mathbf{H'}^\mathsf{T} = \log \mathbf{M} - \log k$ with the definition of $\mathbf{M}$ in Equation 6. Moreover, while it is impractical to directly use the dense matrix [10], we would utilize a variant version called Shifted PPMI as $\mathbf{S} = \max(\log \mathbf{M} - \log k, 0)$. With different definitions of $\mathbf{H}$ and $\mathbf{H'}$, we can generate different representations for downstream tasks. For example, if $\mathbf{H'}$ is an identity matrix, then $\mathbf{H} = \mathbf{S}$. On the other hand, to generate a representation of low dimension d with $d \ll |\mathcal{V} + \mathcal{V}^a| = N + F$, we can let $\mathbf{H} = \mathbf{H'} \in \mathbb{R}^{(N+F) \times d}$ then $\mathbf{H}\mathbf{H'}^\mathsf{T}$ is a rank-d approximation of $\mathbf{S}$. The loss of approximation relies on the spectrum of $\mathbf{S}$ [2].

3 Evaluation and Discussion

We evaluate the matrix-based algorithm in two aspects. First, we compare its performance with some baselines on a task of transductive node classification. Then, we conduct an ablation study to explain why attributed random walk outperforms NetMF[2].

Experiment setup and baselines are shown in Appendix A. All model settings are provided in the Appendix B.

Our Models. To better evaluate various settings of all nodes as $\mathbf{H}_\mathcal{V}, \mathbf{H'}_\mathcal{V}$ and show flexibility as an advantage of matrix factorization methods, we test several forms of representation $\mathbf{H}$ in Table 2 and 3, all denoted as "Ours": 1) $\mathbf{H}_\mathcal{V} = \mathbf{S}_{:N,:N} \in \mathbb{R}^{N \times N}$, as augmented graph weight matrix; 2) $\mathbf{H}_\mathcal{V} = \mathbf{S}_{:N,:N}\mathbf{X} \in \mathbb{R}^{N \times F}$, as augmented node features; 3) $\mathbf{H}_\mathcal{V} = \mathbf{H'}_\mathcal{V} = (\mathbf{U}_d \mathbf{\Sigma}_d^{1/2}) \in \mathbb{R}^{N \times d}$ with $\mathbf{S}_{:N,:N} \approx \mathbf{U}_d \mathbf{\Sigma}_d \mathbf{U}_d^\mathsf{T}$, $d \in \{16, 128, 200\}$, as low-dimensional node embeddings. Models 1 and 2 are followed by a 2-layer MLP since their outputs are still in high dimension, while model 3 is followed by a 1-layer perceptron that can be viewed as logistic regression.

[2] NetMF [14] is a matrix factorization version of DeepWalk [13]. We pick NetMF as a baseline to evaluate how much the attribute graph contributes.

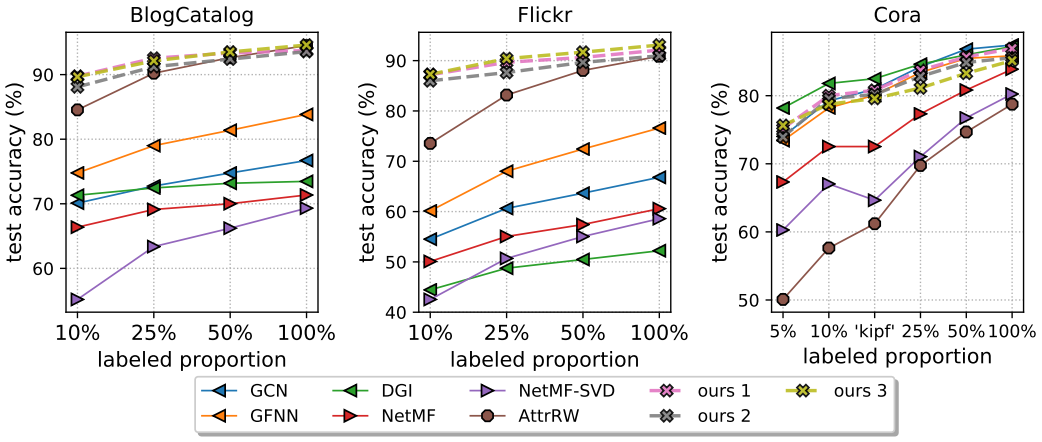


Figure 2: Node Classification Results (test accuracy, %) on BlogCatalog [6], Flickr [6] and Cora [15] with mean accuracies of 10 running. Precise numercial results are provided in Table 2 and 3. Details of baselines and our models are shown in Appendix A and B.

Node Classification Results. The classification results are shown in Figure 2, Table 2 3, with the mean and standard deviation of running with 10 seeds. For social network datasets BlogCatalog and Flickr, our methods consistently outperform all baselines; for citation network dataset Cora, our methods show a competitive performance, especially largely outperforming sample-based attributed random walk [7]. Moreover, our methods are much more robust to label missing in training data. Among our methods, generally the augmented weight matrix version (Ours 1) and the low-dimensional version (Ours 3) are better. Briefly, our attributed random walk with matrix factorization shows the state-of-the-art performance and enjoys the flexibility of output.

Why does attributed random walk work? One motivation of attributed random walk is to mitigate the tendency of converging to nodes with high centralities [7]. We design a toy experiment to verify it as shown in Figure 1 and Table 1. The full version connects $i \in \mathcal{V}, j \in \mathcal{V}^a$ once $\mathbf{X}_{i,j} > 0$ which may still go back to 1-hop neighbours in 2 steps. Now we divide features of each node into "*local*" set $\widetilde{\mathcal{N}}_+$ and "*non-local*" set $\widetilde{\mathcal{N}}_-$. Define $\widetilde{\mathcal{N}}(i)$ as the nonzero feature set for vertex i, which means $\widetilde{\mathcal{N}}(i) = \{j \in \mathcal{V}^a | \mathbf{X}_{i,j} > 0\}$. Define two disjoint subsets of $\widetilde{\mathcal{N}}$ as $\widetilde{\mathcal{N}}_+(i) = \widetilde{\mathcal{N}}(i) \cap [\bigcup_{k \in \mathcal{V}:(i,k)\in\mathcal{E}} \widetilde{\mathcal{N}}(k)]$ and $\widetilde{\mathcal{N}}_-(i) = \widetilde{\mathcal{N}}(i) \setminus \widetilde{\mathcal{N}}_+(i)$. So $\widetilde{\mathcal{N}}_+$ is an attribute node set through which the walk would have a higher probability (probably not 100%) back to 1-hop neighbourhood in 2 steps than $\widetilde{\mathcal{N}}$ while $\widetilde{\mathcal{N}}_-$ would definitely not. We separately substitute $\widetilde{\mathcal{N}}$ with $\widetilde{\mathcal{N}}_-$ or $\widetilde{\mathcal{N}}_+$ in model "Ours 1" and, moreover, take NetMF [14] as the baseline since our model without $\widetilde{\mathcal{N}}$ would perfectly degenerate to it. The results in Table 1 reveal that connections to non-local features showing a comparable performance with the original one make more contribution than local features. Since the attribute matrix $\mathbf{X}$ is sparse and non-negative, connections to non-local features help the walk step into another non-neighbouring vertex with a probability approximating attribute similarities between vertices. Therefore, attributed random walk naturally combines structural connections with node similarity.

Table 1: Node classification results (test accuracy, %) with various sets of connections between nodes and features.

	BlogCatalog	Flickr
Ours 1 w./ $\widetilde{\mathcal{N}}$ (full)	**93.73±0.69**	**92.10±0.70**
Ours 1 w./ $\widetilde{\mathcal{N}}_-$ (non-local)	**91.58±0.76**	**89.50±0.76**
Ours 1 w./ $\widetilde{\mathcal{N}}_+$ (local)	89.53±0.66	67.35±1.16
NetMF [14] (none)	71.36±1.00	60.58±0.81

4 Conclusions

In this work we further develop unsupervised attributed random walks to obtain efficient node embeddings. We formally derive the matrix of pointwise mutual information for attributed random walk. Experimentally our matrix form of attributed random walk enjoys robustness to label missing together with flexible output dimensions. Mechanism of attributed random walk is verified as mitigating converging to nodes with high centralities, which could be extended for future work.

References

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[2] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[3] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[4] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[6] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739. ACM, 2017.

[7] Xiao Huang, Qingquan Song, Yuening Li, and Xia Hu. Graph recurrent networks with attributed random walks. In *SIGKDD Conference on Knowledge Discovery*, 2019.

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

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

[10] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[11] Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[13] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[14] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 459–467. ACM, 2018.

[15] 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.

[16] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[17] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[18] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

Table 2: Node Classification Results (test accuracy, %) on BlogCatalog [6] and Flickr [6]. All results are reported as the mean and standard deviation of running results with 10 random seeds. *Details of our models are shown as "**Our Models**" in Section 3.

dataset	BlogCatalog				Flickr			
$\|\mathcal{V}\|$	5196				7575			
$\|\mathcal{E}\|$	171,743				239,738			
$\|\mathcal{V}^a\|$	8,189				12,047			
Labeled%	10%	25%	50%	100%	10%	25%	50%	100%
GCN [9]	70.10±1.52	72.75±0.89	74.77±1.08	76.70±0.97	54.52±4.65	60.63±1.15	63.66±0.98	66.81±1.44
GFNN [11]	74.78±1.23	79.00±1.54	81.38±1.50	83.84±0.94	60.13±3.02	68.02±2.15	72.43±0.88	76.60±0.80
DGI [17]	71.32±0.79	72.45±0.88	73.18±0.60	73.47±0.64	44.44±0.90	48.75±0.98	50.48±0.77	52.22±0.98
NetMF [14]	66.38±1.40	69.12±1.92	70.01±1.15	71.36±1.00	50.00±1.46	55.05±1.13	57.43±1.35	60.58±0.81
NetMF-SVD [14]	55.19±1.43	63.37±0.89	66.22±1.72	69.32±0.86	42.54±1.27	50.69±1.04	55.05±1.04	58.59±1.07
AttrRW [7]	84.54±1.52	90.22±1.14	**92.63±0.94**	**94.50±0.75**	73.55±1.05	83.16±0.86	88.05±0.82	90.87±0.84
Ours 1*	**89.76±0.85**	**92.54±0.62**	**93.34±0.58**	**93.73±0.69**	**87.19±1.29**	**89.66±1.21**	**90.64±0.75**	**92.10±0.70**
Ours 2*	**88.10±2.28**	**91.22±0.79**	92.40±0.73	93.56±0.85	**85.99±1.29**	**87.63±1.58**	**89.65±0.62**	**90.96±0.91**
Ours 3*	**89.66±1.08**	**92.13±0.50**	**93.51±0.48**	**94.58±0.45**	**87.28±0.78**	**90.40±0.64**	**91.68±0.50**	**93.10±0.51**

Table 3: Node Classification Results (test accuracy, %) on Cora [15]. All results are reported as the mean and standard deviation of running results with 10 random seeds. *Details of our models are shown as "**Our Models**" in Section 3. **11.7% is from Thomas Kipf's split of Cora data [9].

dataset	Cora					
$\|\mathcal{V}\|$	2,708					
$\|\mathcal{E}\|$	5,429					
$\|\mathcal{V}^a\|$	1,433					
Labeled%	5%	10%	11.7%**	25%	50%	100%
GCN [9]	74.02±2.96	79.25±1.05	**80.95±0.64**	**84.26±1.37**	**86.83±0.96**	**87.44±1.10**
GFNN [11]	73.40±3.19	78.21±1.57	80.08±1.23	83.42±1.29	85.48±0.94	85.86±1.07
DGI [17]	**78.19±2.47**	**81.8±1.51**	**82.5±0.46**	**84.63±1.16**	**86.00±1.18**	**87.23±1.00**
NetMF [14]	67.33±2.64	72.53±1.43	72.53±0.55	77.33±1.79	80.83±1.25	83.89±1.25
NetMF-SVD [14]	60.28±3.73	67.04±1.77	64.65±0.73	71.05±1.85	76.74±2.39	80.25±1.26
AttrRW [7]	50.08±2.51	57.63±1.72	61.2±0.75	69.75±1.47	74.67±1.35	78.75±1.11
Ours 1*	**75.19±1.92**	**80.01±1.65**	**80.80±0.57**	**83.73±1.54**	**85.7±0.97**	**86.92±0.99**
Ours 2*	73.93±3.89	**79.56±1.11**	80.18±1.42	82.83±0.67	84.88±1.31	85.58±1.03
Ours 3*	**75.64±2.15**	78.75±1.60	79.59±0.28	81.13±1.44	83.29±1.62	85.15±1.11

A Further Explanations

Word and context. The target of random walks on graphs is to obtain embeddings of all nodes for downstream tasks. A typical approach to generate the embedding of a node u is sampling fixed-length paths starting from u according to the random walk transition probability matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ and recording all observed pairs (u, v) in a multiset-like "*corpus*" $\mathcal{D}$, where v denotes each node that appears in these paths and the length of paths is the window size. Hence, similar to a sentence in natural languages, u is a *word* and v is a *context*, with which we would like to generate the embedding of u through processing the mutual information between u and all $v's$.

DeepWalk and NetMF. DeepWalk [13] first samples a random vertex and a following path on the graph, then utilizes SGNS to maximize mutual information among vertices that appear within a limited-size window. NetMF [14] establish a connection between the target matrix of SGNS and normalized graph Laplacian, obtaining an explicit closed-form matrix that DeepWalk aims to implicitly approximate and factorize: $\frac{\#(w,c)|\mathcal{D}|}{\#(w)\#(c)} \xrightarrow{p} \text{vol}(\mathbf{A})\left(\frac{1}{T}\sum_{r=1}^{T}\mathbf{P}^r\right)\mathbf{D}^{-1}$, where $\text{vol}(\mathbf{A})$ is the volume of a graph adjacency matrix $\mathbf{A}$; T is the window size; $\mathbf{D}$ is the diagonal matrix of node degrees; $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ is the transition probability matrix of random walk on the graph.

Experiment Setup. We take three network benchmarks BlogCatalog [6], Flickr [6] and Cora [15], whose details are included in Table 2 and 3. For BlogCatalog and Flickr, we randomly split each dataset into 80%/20% as training/test set. To evaluate the models' capability of inferring with limited labels, we vary the proportion of labeled data among the training data from 10% to 100%. We randomly choose 10% of the labeled training data are isolated as the validation set. For Cora, we randomly select out 500/1000 nodes as validation/test set, numerically keeping same with Kipf's data split [9], with the rest as the training set and vary the proportion of labeled data from 5% to 100%. In addition, we include the same label setting as Kipf's experiments. For all datasets, the validation set is used for early stopping with a patience of 10 epochs with respect to validation loss.

Baselines. We compare our methods against sample-based Attributed Random Walk [7], NetMF [14], GFNN [11], GCN [9], DGI [17]. Except for GCN, all other methods are followed by a 2-layer MLP or a single1 layer percepton, depending on the output dimension, to evaluate the embedding quality.

B Specific Model Architectures

We compare our methods with several baselines on the node classification task in Section A. Here are some details of the model settings. Note that all models are under the same setup of seed and dataset split. Some specific details are included for experiments on Cora.

- **GCN**: Graph Convolutional Network by [9]. The GCN model has two graph convolutional layers with 128 hidden units. It is trained through the Adam [8] optimizer with learning rate of 0.01, batch size of 128, dropout rate of 0.5 for 200 epochs with 10 as patience.
- **GFNN**: Graph Filter Neural Network by [11]. We set the power order of normalized augmented adjacency matrix as 2 and hidden units as 128. We have tried classifiers of a 2-layer MLP and a single-layer perceptron, where the latter would degenerate to SGC [18]. Results of the 2-layer MLP are much better so reported. It is trained with the Adam optimizer with learning rate of 0.01, batch size of 128, dropout rate of 0.5 for 200 epochs with 10 as patience.
- **DGI**: Deep Graph Infomax by [17]. The feature extractor is a 2-layer GCN with 200 hidden units. The classifier is a single-layer perceptron that can be viewed as logistic regression. The extractor is trained through the Adam optimizer with learning rate of 0.0001 for 10000 with 100 as patience. The classifier is trained through the Adam optimizer with learning rate of 0.01 for 20000 epochs and the model with the lowest validation loss is taken for evaluation on the test data. The negative sampling method is the same as proposed in the authors' code[3].
- **NetMF**: Network Embedding as Matrix Factorization by [14]. The algorithm we utilize here is the "algorithm 3" for a small window size in the original literature but without the last step of SVD. Therefore, the embedding is in $\mathbb{R}^{N\times N}$. The classifier is a 2-layer perceptron with 200 hidden units. It is trained through the Adam optimizer with learning rate of 0.001, batch size of 128, dropout of 0.5 for 200 epochs with 10 as patience. The window size is 5.
- **NetMF-SVD**: Also from [14]. The algorithm we utilize here is exactly the "algorithm 3" for a small window size in the original literature. The SVD is a truncated version of rank 200. Therefore, the embedding is in $\mathbb{R}^{N\times 200}$. The classifier is a single-layer perceptron. It is trained through the Adam optimizer with learning rate of 0.001, batch size of 128, dropout of 0.5 for 200 epochs with 10 as patience. The window size is 5. On Blogcatalog and Flickr, it has output dimension of $d = 200$. On Cora, it output embeddings in dimension of $d = 128$ for data split of $50\%, 100\%$ while in dimension of $d = 16$ for fewer labels.
- **AttrRW**: Attributed Random Walk by [7]. The implementation is with exactly the authors' code[4] with the RNN and BatchNorm parts eliminated. The model is with 800 hidden units, 100 number of paths for each node, 0.5 as α and 5 as the lengths of paths. It is trained through the Adam optimizer with learning rate of 0.0001, batch size of 32, weight decay of 5e-4, dropout rate of 0.5 for 200 epochs with 10 as patience.
- **Ours**: Definitions of representations $\mathbf{H}$ for all the three models are provided in section 3. The window size is 5 and α is 0.5. The classifier for "ours 1" and "ours 2" is a 2-layer perceptron with 200 hidden units while that for "our 3" is a single-layer perceptron. They are trained through the Adam optimizer with learning rate of 0.001, weight decay of 5e-4, dropout rate of 0.5, batch size of 128 for 200 epochs with 10 as patience. On Blogcatalog and Flickr, "ours 3" has output dimension of $d = 200$. On Cora, "ours 3" output embeddings in dimension of $d = 128$ for data split of $50\%, 100\%$ while in dimension of $d = 16$ for fewer labels.

[3] https://github.com/PetarV-/DGI/tree/0afce4e36b5edbe1e735536d15b748d0381e4083

[4] https://github.com/xhuang31/GraphRNA_KDD19/tree/c9bcaf5d9ca0f8e266a12762cce78b1d240eaa50