

# Which way?

## Direction-Aware Attributed Graph Embedding<sup>\*</sup>

Zekarias T. Kefato<sup>1</sup>, Nasrullah Sheikh<sup>2</sup>, and Alberto Montresor<sup>2</sup>

<sup>1</sup> Royal Institute of Technology, Stockholm, Sweden

zekarias@kth.se

<sup>2</sup> University of Trento

{nasrullah.sheikh, alberto.montresor}@unitn.it

**Abstract.** Graph embedding algorithms are used to efficiently represent (encode) a graph in a low-dimensional continuous vector space that preserves the most important properties of the graph. One aspect that is often overlooked is whether the graph is directed or not. Most studies ignore the directionality, so as to learn high-quality representations optimized for node classification. On the other hand, studies that capture directionality are usually effective on link prediction but do not perform well on other tasks.

This preliminary study presents a novel text-enriched, direction-aware algorithm called DIAGRAM, based on a carefully designed multi-objective model to learn embeddings that preserve the direction of edges, textual features and graph context of nodes. As a result, our algorithm does not have to trade one property for another and jointly learns high-quality representations for multiple network analysis tasks. We empirically show that DIAGRAM significantly outperforms six state-of-the-art baselines, both direction-aware and oblivious ones, on link prediction and network reconstruction experiments using two popular datasets. It also achieves a comparable performance on node classification experiments against these baselines using the same datasets.

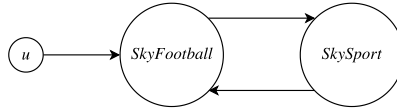
## 1 Introduction

Recently, we have seen a tremendous progress in graph embedding algorithms that are capable to learn high-quality vector space embeddings of graphs. Most of them are based on different kinds of neural networks, such as the standard multi-layer perceptron (MLP) or convolutional neural networks (CNN). Regardless of the architecture, most of them are optimized to learn embeddings for effective node classification.

Even though they have been shown to be useful for link prediction, they are only capable to predict if two nodes are connected; they are not able to identify the direction of the edge. For example, in social networks like Twitter, it is very likely for “organic users” (such as  $u$  in Fig. 1) to follow popular users like *SkyFootball*, based on their interest on the provided content, while the follow-back is very unlikely. Consequently, transitivity from  $u$  to *SkySport* is usually asymmetric as empirically demonstrated by [1]. For this reason, a good embedding algorithm over directed graphs should not be oblivious to directionality.

---

<sup>\*</sup> The source code is provided in the following github repo: <https://github.com/zekarias-tilahun/diagram>



**Fig. 1.** An example follower graph between a random football enthusiast Twitter user and two popular football and sport news channels Twitter accounts.

Another pressing issue in graph embedding research is the need for preserving similarity (homophily) between nodes. For example, the similarity between  $u$  and *SkyFootball* is usually dependent on the undirected neighborhood and the content produced/consumed by them [2, 3]. Thus, it is more important to examine the content and the entire neighborhood of nodes to measure their similarity.

Existing studies usually trade-off between preserving directionality versus similarity [1, 3–12, 2, 13, 14]. Hence, they either learn high-quality embeddings for classification at the expense of asymmetric link prediction, or vice versa.

In this study, we argue that we do not have to choose between these aspects and we propose a novel algorithm called **DIAGRAM (Direction-Aware Attributed Graph Embedding)** that seeks to learn embeddings that balance the two aspects: **DIAGRAM** preserves directionality and hence asymmetric transitivity, while at the same time striving to preserve similarity. We achieve this by designing a multi-objective loss function that seeks to optimize with respect to (1) direction of edges (2) content of nodes (Binary or TF-IDF features) and (3) undirected graph context.

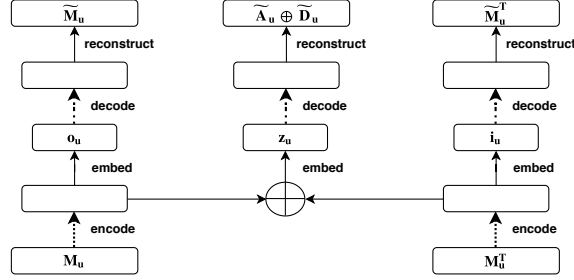
First we propose a variant of **DIAGRAM** called  $\text{DIAGRAM}_{node}$ ; after investigating its limits, we propose an extension called  $\text{DIAGRAM}_{edge}$  to address them.

The paper is structured as follows. In Section 2 we give a description of the proposed model and report the empirical evaluation in Section 3. In Section 4 we discuss related work. Section 5 concludes the paper.

## 2 Model

Let  $G = (N, E, \mathbf{M}, \mathbf{D})$  be a directed graph, where  $N$  is a set of  $n$  nodes and  $E$  is a set of  $m$  edges.  $\mathbf{M} \in \mathbb{R}^{n \times n}$  is an adjacency matrix, binary or real, and  $\mathbf{D} \in \mathbb{R}^{n \times d}$  is a feature matrix, for instance constructed from a document corpus  $\mathcal{D} = \{D_1, \dots, D_n\}$ , where  $D_u$  is a document associated to node  $u \in N$ . Each word  $w \in D_u$  is a sample from a vocabulary  $V$  of  $d$  words. Matrix  $\mathbf{D}$  can be a simple binary matrix, where  $\mathbf{D}_{u,i} = 1$  means that the word, the  $i$ -th word of  $V$ , appears in document  $D_u$  of user  $u$ . It can also be a TF-IDF weight that captures the importance of word  $w$  to node  $u$ .

For every node  $u \in N$ , our objective is to identify three embedding vectors,  $\mathbf{e}_u = \{\mathbf{z}_u, \mathbf{o}_u, \mathbf{i}_u\}$ . A recent study [15] has shown the power of learning multiple embedding vectors of each node that capture different social contexts (communities). In our study, however, each embedding vector is designed to meet the three goals that we set out to achieve in the introduction, which are:



**Fig. 2.** The DIAGRAM architecture. Dotted and solid arrows indicate an MLP and SLP, respectively

1. Undirected neighborhood and content similarity-preserving embedding –  $\mathbf{z}_u \in \mathbb{R}^k, k \ll n$  – based on  $\mathbf{M}_u | \mathbf{M}_u^T$  and  $\mathbf{D}$ , where  $|$  is a bitwise or.
2. Outgoing neighborhood-preserving embedding –  $\mathbf{o}_u \in \mathbb{R}^k$  – based on  $\mathbf{M}_u$
3. Incoming neighborhood-preserving embedding –  $\mathbf{i}_u \in \mathbb{R}^k$  – based on  $\mathbf{M}_u^T$

To motivate our work, suppose we want to estimate the proximity between  $u$  and  $v$ . If  $(u, v) \in E$  is a reciprocal edge, then the estimated proximity should be equally high for both  $(u, v)$  and  $(v, u)$ , otherwise it should be high for  $(u, v)$  and very small for  $(v, u)$ .

To this end, we adopt the technique proposed by [1] for estimating the asymmetric proximity between  $u$  and  $v$  as the dot product between  $\mathbf{o}_u \in \mathbb{R}^k$  and  $\mathbf{i}_v \in \mathbb{R}^k$  as:

$$\text{prx}(u, v) \propto \text{dot}(\mathbf{o}_u, \mathbf{i}_v)$$

which are the learned embeddings of node  $u$  and  $v$  that capture their outgoing and incoming neighborhood, respectively.

*Remark 1.* To satisfy the above condition, if there is an unreciprocated directed edge  $(u, v) \in E$  we consider that as a high non-symmetric proximity, and hence  $\mathbf{o}_u$  and  $\mathbf{i}_v$  should be embedded close to each other.

Thus, our goal is to jointly learn three embeddings  $\mathbf{e}_u = \{\mathbf{z}_u, \mathbf{o}_u, \mathbf{i}_u\}$  of each node  $u \in N$  using a single shared autoencoder model as follows.

## 2.1 Node Model

The first variant of DIAGRAM is  $\text{DIAGRAM}_{node}$  (Fig. 2); for each node  $u$ , its inputs are  $\mathbf{M}_u$  and  $\mathbf{M}_u^T$ . It has four components, which are encode, embed, decode, and reconstruct. The encode and decode components are formulated as a multi-layer perceptron (MLP), with  $L_{enc}$ , and  $L_{dec}$  layers, respectively, and the output of each layer  $l$  is specified as:

$$\mathbf{h}^l = \tanh(\mathbf{W}^l \cdot \mathbf{h}^{l-1} + \mathbf{b}^l)$$

where the weight matrix at the  $l^{th}$  layer,  $\mathbf{W}^l$ , is shared in the process of learning,  $\mathbf{e}_u : u \in N$ .

On the other hand, `embed` and `reconstruct` are a single-layer perceptron (SLP) defined as:

$$\begin{aligned} \text{embed} &= \tanh(\mathbf{W}_{\text{emb}} \cdot \mathbf{h}^{\text{Lenc}} + \mathbf{b}_{\text{emb}}) \\ \text{reconstruct} &= \tanh(\mathbf{W}_{\text{rec}} \cdot \mathbf{h}^{\text{Ldec}} + \mathbf{b}_{\text{rec}}) \end{aligned}$$

where  $\mathbf{W}_{\text{emb}}$  and  $\mathbf{W}_{\text{rec}}$  are the weight matrices for the embedding and reconstruction layers, respectively, which are also shared while learning the parameters  $\mathbf{e}_u : u \in N$ .  $\mathbf{h}^{\text{Lenc}}$  and  $\mathbf{h}^{\text{Ldec}}$  are the activations of the last layers of the encoder and decoder, respectively.  $\mathbf{b}^l$ ,  $\mathbf{b}_{\text{emb}}$  and  $\mathbf{b}_{\text{rec}}$  are the biases for the  $l^{th}$ -layer, embedding layer and reconstruction layer, in their respective order.

**Optimization** The model parameters are trained by minimizing the reconstruction errors on the weight and feature matrices  $\mathbf{M}$  and  $\mathbf{D}$ , respectively as follows:

$$\begin{aligned} \mathcal{L}_C^u &= \|(\mathbf{A}_u - \tilde{\mathbf{A}}_u) \odot \mathbf{s} + (\mathbf{D}_u - \tilde{\mathbf{D}}_u) \odot \mathbf{s}\|_2^2 \\ \mathcal{L}_O^u &= \|(\mathbf{M}_u - \tilde{\mathbf{M}}_u) \odot \mathbf{s}\|_2^2 \\ \mathcal{L}_I^u &= \|(\mathbf{M}_u^T - \tilde{\mathbf{M}}_u^T) \odot \mathbf{s}\|_2^2 \\ \mathcal{L} &= \min_{\Theta} \sum_{u \in N} \mathcal{L}_C^u + \mathcal{L}_O^u + \mathcal{L}_I^u \end{aligned}$$

where  $\odot$  is an element-wise multiplication,  $\mathbf{A} = \mathbf{M}|\mathbf{M}^T$  is the undirected adjacency matrix,  $\Theta$  denotes the set of all learnable model parameters, and  $\mathbf{s}$  is a simple weighting trick adopted from [4] to avoid the trivial solution of reconstructing the zeros for sparse matrices. That is, for the  $j^{th}$  entry,  $\mathbf{s}_j = \mu > 1$  if  $\mathbf{A}_{u,j} \geq 1$  otherwise  $\mathbf{s}_j = 1$ . In all the experiments we use  $\mu = 10$ .

Therefore,  $\mathcal{L}_C$  enables us to preserve the combined neighborhood, in and out, and the content similarity between nodes.  $\mathcal{L}_O$  and  $\mathcal{L}_I$  enables us to preserve the outgoing and incoming neighborhood of nodes, respectively.

## 2.2 Edge Model

The limit of `DIAGRAMnode` is that it does not satisfy the constraint that we specify in Remark 1. To solve this, we propose `DIAGRAMedge` that has the same architecture depicted in Fig. 2, but iterates over the edges. For each directed edge  $(u, v) \in E$ , we execute `DIAGRAM` on both  $u$  and  $v$ ; we use however a simple trick by adjusting  $\mathcal{L}_I$  for  $u$  as:

$$\mathcal{L}_I^u = \|(\mathbf{M}_v^T - \tilde{\mathbf{M}}_u) \odot \mathbf{s}\|_2^2$$

In other words, the reconstructions of  $u$ 's out-neighborhood,  $\tilde{\mathbf{M}}_u$ , is ensured to be consistent with  $v$ 's actual incoming neighborhood,  $\mathbf{M}_v^T$ , consequently reducing the distance  $\mathbf{o}_u - \mathbf{i}_v$  between  $u$ 's outgoing and  $v$ 's incoming embedding, and hence projecting them close to each other.

Although iterating over the edges in large networks might be expensive, we employ *transfer learning* and use the trained parameters of the node model as a starting point. In our experiments, transfer learning ensures convergence in one or two epochs, otherwise  $\text{DIAGRAM}_{edge}$  requires at least 30 epochs.

For both  $\text{DIAGRAM}_{node}$  and  $\text{DIAGRAM}_{edge}$  we use dropout regularization as our datasets are extremely sparse.

### 3 Experimental Evaluation

We have evaluated the performance of our algorithm against two popular citation network datasets summarized in Table 1, comparing it against the following state-of-the-art baselines.

#### 3.1 Baselines

We include six baselines that preserve both symmetric and asymmetric neighborhoods, as well as content similarity.

1. HOPE [1] is a direction-aware algorithm that uses a generalized variant of Singular Value Decomposition.
2. DEEPWALK [6] and NODE2VEC [7] are algorithms that learn a symmetric neighborhood of nodes using SKIPGRAM [16] by leveraging sampled random walks.
3. DANE [3], ANRL [5] and VGAE [17] preserve both neighborhood and content similarity of nodes, yet only in a symmetric manner. DANE and ANRL use different architectures of autoencoders, whereas VGAE uses a variational autoencoder.

We use three kinds of common network analysis tasks, which are *network reconstruction*, *link prediction*, and *node classification* in order to evaluate the performance of the embedding algorithms.

#### 3.2 Settings

The parameters of  $\text{DIAGRAM}$  are tuned using random search, while for the baselines we use the source code provided by the authors and the optimal values reported in the corresponding papers. For both datasets,  $\text{DIAGRAM}$ 's variants have the same configuration. The encoder layer configuration is  $[|V|, 512, 256]$ , and the embedding size  $k$  is 128. The decoder layer configuration is  $[128, 256, 512]$ , and for reconstruction layer the configuration is  $[512, |V| + |F|]$ . Dropout rate is equal to 0.2 (Cora) and 0.1 (Citeseer); the learning rate is 0.0001 for both.

Dataset	$ N $	$ E $	$d =  V $	#Labels
Cora	2,708	5,278	3,703	7
Citeseer	3,312	4,660	1,433	6

**Table 1.** Dataset Summary

Dataset	Algorithms	P@K(%)			
		K=2500	K=5000	K=7500	K=10000
Citeseer	DIAGRAM <sub>edge</sub>	<b>57</b>	<b>44</b>	<b>37</b>	<b>31</b>
	DIAGRAM <sub>node</sub>	46	36	29	25
	DANE	3	11	13	12
	ANRL	3	2	2	2
	VGAE	16	13	12	11
	HOPE	46	38	30	24
	DEEPWALK	5	5	4	4
	NODE2VEC	19	16	16	12
	Cora	DIAGRAM <sub>edge</sub>	<b>59</b>	<b>49</b>	<b>42</b>
DIAGRAM <sub>node</sub>		46	38	31	30
DANE		10	10	9	0
ANRL		3	3	2	2
VGAE		21	18	15	14
HOPE		48	42	36	30
DEEPWALK		9	9	9	8
NODE2VEC		6	6	7	10

Table 2. P@K results of the network reconstruction experiment

### 3.3 Network Reconstruction

A good embedding algorithm should be able to preserve the structure of the network, and the goodness is usually verified by evaluating its performance in reconstructing the network. Following the practice of related studies, we quantify the performance using the precision-at-k (P@K) metric.

To this end, we first compute the pairwise proximity between all nodes. For every pair of nodes  $u$  and  $v$ , a direction-aware algorithm computes proximity  $\text{prx}(u, v)$  between  $u$  and  $v$  as

$$\text{prx}(u, v) = \frac{1}{1 + e^{-\text{dot}(\mathbf{o}_u, \mathbf{i}_v)}}$$

and for direction-oblivious algorithms we simply have a single embedding  $\mathbf{z}_u$  and  $\mathbf{z}_v$  for  $u$  and  $v$ , respectively, and hence  $\text{prx}$  is computed as:

$$\text{prx}(u, v) = \frac{1}{1 + e^{-\text{dot}(\mathbf{z}_u, \mathbf{z}_v)}}$$

Finally, node pairs are ranked according to their proximity score, and P@K is simply the precision measured with respect to the ground truth edges at cut-off  $K$ .

The results of the network reconstruction experiments are reported in Table 2; our algorithms—DIAGRAM<sub>edge</sub> in particular—outperforms all the baselines with a significant margin. In addition, as a consequence of satisfying Remark 1, DIAGRAM<sub>edge</sub> performs much better than DIAGRAM<sub>node</sub>.

The Cora dataset for instance has  $\approx 5000$  edges, and we see that for  $K = 5000$ , out of the  $\approx 7,000,000$  possible pairs DIAGRAM<sub>edge</sub> correctly filters out almost half of the ground truth edges. These results are 7% and 11% better than HOPE and DIAGRAM<sub>node</sub>, respectively, and they are in stark contrast to the direction-oblivious baselines.

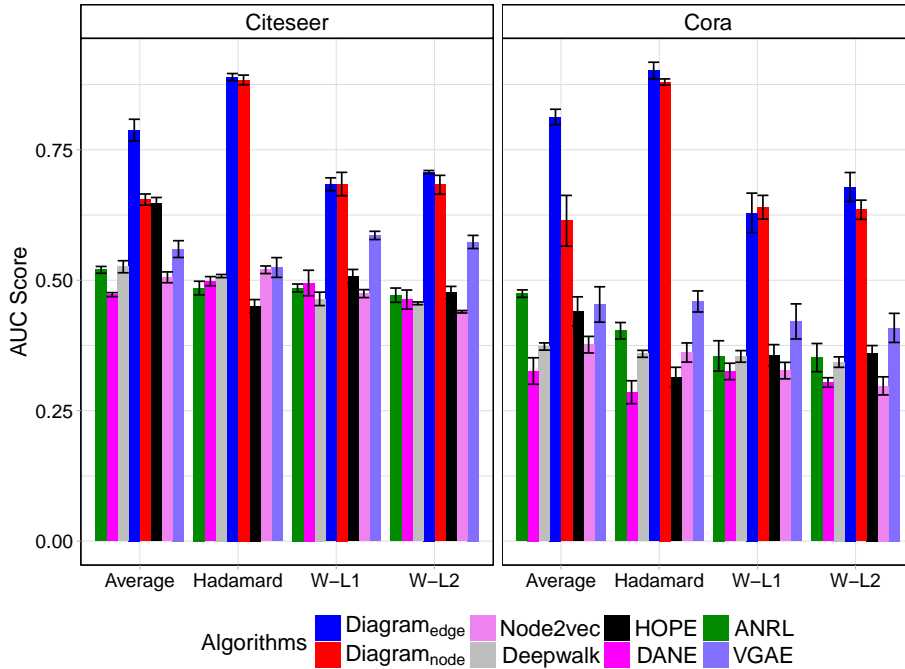


Fig. 3. AUC results for link prediction, and  $p = 10$ .

### 3.4 Link Prediction

One of the most common real-world applications of graph embedding algorithms is link prediction. In this experiment, similar to [7, 4, 18] we first sample  $p$  percent of the edges from the graph. We refer to these edges as *true samples* and remove them from the graph by ensuring that the residual graph remains connected. We also sample  $p$  percent of pair of nodes that are not connected by an edge, referred to as *false samples*.

To show how robust our approach is, we train the variants of DIAGRAM on the residual graph, while the baselines are trained on the complete graph. Then we use the learned embeddings of the algorithms to predict edges as follows.

For all the sampled edges, true and false, we construct edge features using the techniques proposed by [7], which are the average, element wise multiplication, weighted-L1, and weighted-L2 (Average, Hadamard, W-L1, W-L2, respectively) of the embeddings of the source and target nodes of each edge. For direction-aware algorithms (DIAGRAM and HOPE), edge features are asymmetric,  $\mathbf{o}$  and  $\mathbf{i}$  are used, while the other baselines are symmetric, only  $\mathbf{z}$  is used. Then, for each sample, we assign the label ‘1’ if it comes from the true ones, ‘0’ otherwise.

We finally train a binary classifier using a 3-fold cross validation and we report the mean of the area under the curve (AUC) score in Fig. 3, along with the error margin. Although DIAGRAM’s variants have been trained on the residual graph, yet we see that they significantly outperform all the baselines across all feature construction techniques.

Training Ratio	Algorithm	Dataset			
		Cora		Citeseer	
		Micro-F1(%)	Macro-f1(%)	Micro-F1(%)	Macro-F1(%)
10%	DIAGRAM <sub>edge</sub>	75	73	60	56
	DIAGRAM <sub>node</sub>	74	72	60	56
	DANE	<b>77</b>	<b>75</b>	61	57
	ANRL	74	72	<b>66</b>	<b>62</b>
	VGAE	75	72	58	52
	HOPE	63	61	41	37
	DEEPWALK	70	68	48	45
	NODE2VEC	73	72	51	47
30%	DIAGRAM <sub>edge</sub>	<b>80</b>	<b>79</b>	68	64
	DIAGRAM <sub>node</sub>	79	78	68	63
	DANE	79	77	66	62
	ANRL	77	75	<b>71</b>	<b>67</b>
	VGAE	77	74	60	53
	HOPE	73	71	49	45
	DEEPWALK	77	76	55	51
	NODE2VEC	79	78	58	54
50%	DIAGRAM <sub>edge</sub>	<b>82</b>	<b>80</b>	70	65
	DIAGRAM <sub>node</sub>	81	79	69	64
	DANE	81	79	70	65
	ANRL	78	76	<b>73</b>	<b>68</b>
	VGAE	78	75	60	53
	HOPE	76	75	50	46
	DEEPWALK	80	79	56	52
	NODE2VEC	81	<b>80</b>	60	55

**Table 3.** F1-Micro and F1-Macro results for node classification experiments on Cora and Citeseer datasets. Each experiment is carried out using a fraction (10, 30, and 50 percent) of the data as training set and the rest as test set.

This is due to the fact that DIAGRAM uses direction-aware embeddings. Even though HOPE uses such kind of embeddings as well, its linear model is not powerful enough to capture the highly non-linear structure of real graphs [4]. Furthermore, DIAGRAM<sub>edge</sub> performs better than DIAGRAM<sub>node</sub>, mostly with smaller variance, with exception of W-L1.

### 3.5 Node Classification

Following the most common practices in these kinds of experiments, we train a multi-class one-vs-rest classifier using logistic regression. We use learned embeddings as node features to predict the corresponding label. Yet again, we perform a 10-fold cross-validation experiment; Table 3 reports the average Micro-F1 and Macro-F1 metrics.

For Cora, in most cases DIAGRAM performs better than DANE, ANRL, and VGAE, methods known for their superior performance in node classification. For Citeseer, however, DIAGRAM performs worst than ANRL, comparable to DANE and better than VGAE.



Note that DIAGRAM is significantly better than HOPE, the direction-aware baseline. This is due to the fact that DIAGRAM attempts to balance both direction and content by incorporating textual features and the undirected neighborhood.

## 4 Related Work

Classical graph embedding techniques rely on matrix factorization techniques. Fairly recently, however, several studies have been proposed based on shallow and deep neural networks [2–7, 9, 18]. The earlier ones, such as DEEPWALK [6] and NODE2VEC [7], are techniques based on sampled random walks that are intended to capture the local neighborhood of nodes. Virtually all methods based on random walks rely on the SKIP-GRAM model [16], whose objective is to maximize the probability of a certain neighbor node  $v \in N$  of an anchor node  $u \in N$ , given the current embedding  $\mathbf{z}_u$  of  $u$ . One of the main drawbacks of random walk based methods is their sampling complexity.

To address this limitation, several follow up studies, e.g. [3–5], have been proposed based on deep-feed forward neural networks, such as Deep Autoencoders. However, all these methods are inherently applicable for undirected networks, as they do not explicitly care for directionality.

A method called Hope [1] has been proposed for preserving important properties in directed networks. However, it relies on matrix factorization, and do not effectively capture the highly non-linear structure of the network [4].

Our algorithm addresses the aforementioned limitations, however the results of the node-classification experiment are not satisfactory and we intend to carefully investigate the problem and design a better algorithm in the complete version of this paper.

## 5 Conclusion

In this study we propose an ongoing, yet novel direction-aware, text-enhanced graph embedding algorithm called DIAGRAM. Unlike most existing studies that trade-off between learning high-quality embeddings for either link prediction or node classification, DIAGRAM is capable to balance both.

We have empirically shown DIAGRAM’s superior performance in asymmetric link prediction and network reconstruction and comparable performance in node classification over the state-of-the-art. However, we are still investigating directions to improve DIAGRAM so as to improve its performance on node classification as well and we also seek to incorporate more datasets. We shall cover all of these aspects in a future version of this study.

## References

1. M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, “Asymmetric transitivity preserving graph embedding,” in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: ACM, 2016, pp. 1105–1114. [Online]. Available: <http://doi.acm.org/10.1145/2939672.2939751>
2. S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, “Tri-party deep network representation,” in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI’16. AAAI Press, 2016, pp. 1895–1901. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3060832.3060886>
3. H. Gao and H. Huang, “Deep attributed network embedding,” in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 7 2018, pp. 3364–3370. [Online]. Available: <https://doi.org/10.24963/ijcai.2018/467>
4. D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: ACM, 2016, pp. 1225–1234. [Online]. Available: <http://doi.acm.org/10.1145/2939672.2939753>
5. Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, and C. Wang, “Anrl: Attributed network representation learning via deep neural networks,” in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 7 2018, pp. 3155–3161. [Online]. Available: <https://doi.org/10.24963/ijcai.2018/438>
6. B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’14. ACM, 2014, pp. 701–710.
7. A. Grover and J. Leskovec, “Node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. ACM, 2016, pp. 855–864.
8. J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “LINE: large-scale information network embedding,” *CoRR*, vol. abs/1503.03578, 2015. [Online]. Available: <http://arxiv.org/abs/1503.03578>
9. C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, “Network representation learning with rich text information,” in *Proc. of the 24th Int. Conf. on Artificial Intelligence*, ser. IJCAI’15. AAAI Press, 2015, pp. 2111–2117. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2832415.2832542>
10. D. Yang, S. Wang, C. Li, X. Zhang, and Z. Li, “From properties to links: Deep network embedding on incomplete graphs,” in *Proc. of the 2017 ACM on Conf. on Information and Knowledge Management*, ser. CIKM ’17. ACM, 2017, pp. 367–376. [Online]. Available: <http://doi.acm.org/10.1145/3132847.3132975>
11. W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *CoRR*, vol. abs/1706.02216, 2017. [Online]. Available: <http://arxiv.org/abs/1706.02216>
12. T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *CoRR*, vol. abs/1609.02907, 2016. [Online]. Available: <http://arxiv.org/abs/1609.02907>
13. P. Velickovi, G. Cucurull, A. Casanova, A. Romero, P. Li, and Y. Bengio, “Graph attention networks,” in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=rJXMpikCZ>
14. R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems,” *CoRR*, vol. abs/1806.01973, 2018. [Online]. Available: <http://arxiv.org/abs/1806.01973>

15. A. Epasto and B. Perozzi, “Is a single embedding enough? learning node representations that capture multiple social contexts,” in *The World Wide Web Conference*, ser. WWW '19. New York, NY, USA: ACM, 2019, pp. 394–404. [Online]. Available: <http://doi.acm.org/10.1145/3308558.3313660>
16. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 3111–3119. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2999792.2999959>
17. T. N. Kipf and M. Welling, “Variational graph auto-encoders,” *NIPS Workshop on Bayesian Deep Learning*, 2016.
18. N. Sheikh, Z. Kefato, and A. Montresor, “GAT2VEC: Representation learning for attributed graphs,” *Computing*, 2018.