GENERATIVE GRAPH CONVOLUTIONAL NETWORK FOR GROWING GRAPHS

Da Xu, Chuanwei Ruan, Kamiya Motwani, Evren Korpeoglu, Sushant Kumar, Kannan Achan

Walmart Labs, Sunnyvale, California, USA

ABSTRACT

Modeling generative process of growing graphs has wide applications in social networks and recommendation systems, where cold start problem leads to new nodes isolated from existing graph. Despite the emerging literature in learning graph representation and graph generation, most of them can not handle isolated new nodes without nontrivial modifications. The challenge arises due to the fact that learning to generate representations for nodes in observed graph relies heavily on topological features, whereas for new nodes only node attributes are available. Here we propose a unified generative graph convolutional network that learns node representations for all nodes adaptively in a generative model framework, by sampling graph generation sequences constructed from observed graph data. We optimize over a variational lower bound that consists of a graph reconstruction term and an adaptive Kullback-Leibler divergence regularization term. We demonstrate the superior performance of our approach on several benchmark citation network datasets.

Index Terms— Graph representation learning, sequential generative model, variational autoencoder, growing graph

1. INTRODUCTION

1.1. Background

Real-world graph structured data is often under dynamic growth as new nodes emerge over time. However, directly modeling the generation process from observed graph data remains a difficult task due to the complexity of graph distributions. Recently, there have been significant advances in learning graph representations by mapping nodes onto latent vector space. The latent factors (embeddings) which have simpler geometric structures can then be used for downstream machine learning analysis such as generating graph structures [1] and various semi-supervised learning tasks [2].

Some early approaches in node embedding such as *graph factorization algorithm* [3], *Laplacian eigenmaps* [4] and *HOPE* [5] are based on deterministic matrix-factorization techniques. Later approaches arise from random walk techniques that provide stochastic measures for analysis, including *DeepWalk* [6], *node2vec* [7] and *LINE* [8]. More recent graph embedding techniques focus on building deep graph covolutional networks (*GCN*) [9] as encoders that aggregate neighborhood information [10]. Variants of *GCN* have been proposed to tackle the computational complexity for large graphs, such as *FastGCN* [11] which applies graph sampling techniques. *GraphSAGE* [12] is another time-efficient inductive graph representation learning approach that implements localized neighborhood aggregations.

On the other side, advancements in generative models compatible with deep neural networks such as variational autoencoders (*VAE*) [13, 14] and generative adversarial networks (*GAN*) [15] have enabled direct modeling for generation of complex distributions. As a consequence, there have been several recent work on deep generative models for graphs [16, 17, 18, 19]. However, many of them only deal with fixed graphs [19, 18] or graphs of very small sizes [17, 1]. Moreover, most graph representation learning methods require at least some topological features from all nodes in order to conduct neighborhood aggregations or random walks, which is clearly infeasible for growing graphs with isolated new nodes. To obtain embeddings and further generate graph structures for both new and old nodes, it is essential to utilize node attributes. Also, instead of learning how the observed graph is generated as a whole, the graph generation should be decomposed into sequences that reflect how new nodes are sequentially fitted into existing graph structures.

1.2. Related methods

Variational Autoencoder Unlike vanilla autoencoder, VAE treats the latent factors $\boldsymbol{z}$ as random variables such that they can capture variations in the observed data $\boldsymbol{x}$ [13]. *VAE* has shown high efficiency in recovering complex multimodal data distributions. The parameters in encoding distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and decoding distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ are optimized over the evidence (variational) lower bound (ELBO)

$$\log p(\boldsymbol{x}) \geq E_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_0(\boldsymbol{z})).$$

The expectation with respect to $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is approximated stochastically by reparametrizing $\boldsymbol{z}$ as $\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ are independent standard Gaussian variables. This is also referred to as 'reparameterization trick' [13]. It allows sampling directly from $\boldsymbol{z}$ so that the backpropagation technique becomes feasible for training deep networks.

Graph Convolutional Network The original *GCN* [9] deals with node classification as a semi-supervised learning

task. The layer-wise propagation rule is defined as $\boldsymbol{H}^{l+1} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{H}^{l}\boldsymbol{W}_l)$. Here $\hat{\boldsymbol{A}}$ is the normalized adjacency matrix with $\hat{\boldsymbol{A}}_{i,j} = \frac{\boldsymbol{A}_{i,j}}{\sqrt{deg(i)deg(j)}}$ where $deg(i)$ gives the degree of node i. The $\sigma(.)$ is some activation function such as ReLU. $\boldsymbol{H}^l$ is the output of the $(l-1)^{th}$ layer and $\boldsymbol{W}_l$ is the layer-specific aggregation weights. Here $\boldsymbol{W}_l \in \mathbf{R}^{d_l \times d_{l+1}}$ where d_l is the dimension of the hidden units on l^{th} layer.

Graph Convolutional Autoencoder (*GAE*) *GAE* is an important extension of *GCN* for learning node representations for link prediction [18]. A two-layer *GCN* is used as encoder $\boldsymbol{z} = GCN(\boldsymbol{A}, \boldsymbol{X}) = \hat{\boldsymbol{A}}ReLu(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}_0)\boldsymbol{W}_1$. When adapting *GCN* into *VAE* framework (*GCN-VAE*), the hidden factors $\boldsymbol{z}$ are assumed to follow independent normal distributions which are parameterized by mean $\boldsymbol{\mu}$ and log of standard deviation $\boldsymbol{\sigma}$, where $\boldsymbol{\mu} = GCN_{\mu}(\boldsymbol{A}, \boldsymbol{X})$ and $\boldsymbol{\sigma} = GCN_{\sigma}(\boldsymbol{A}, \boldsymbol{X})$. The pairwise decoding (generative) distribution for link between node i and j is simply $f(\langle \boldsymbol{z}^i, \boldsymbol{z}^j \rangle)$ where $f(.)$ is the sigmoid function. The ELBO is formulated as $E_{q(\boldsymbol{z}|\boldsymbol{X},\boldsymbol{A})}[\log p(\boldsymbol{A}|\boldsymbol{z})] - KL(q(\boldsymbol{z}|\boldsymbol{A}, \boldsymbol{X})||p_0(\boldsymbol{z}))$.

GraphRNN Recently a graph generation approach relying only on topological structure was proposed in [16]. It learns the sequential generation mechanism by training on a set of sampled sequences of decomposed graph generation process. The formation of each new edge is conditioned on the graph structure generated so far. Our approach refers to this idea of sampling from decomposed generation sequences.

1.3. Present Work

This work addresses the challenge of generating graph structure for growing graphs with new nodes that are unconnected to the previous observed graph. It has important meaning for the cold start problems [20] in social networks and recommender systems. The major assumption is that the underlying generating mechanism is stationary during growth. *GraphRNN* neither takes advantage of node attributes nor does it naturally extends to isolated new nodes. Most other graph representation learning methods have similar issues, specifically the isolation from existing graph hinders passing messages or implementing aggregation.

We deal with this problem by learning how graph structures are generated sequentially, for cases where both node attributes and topological information exist as well as for cases where only node attributes are available. To the best of our knowledge, this work is the first of its kind in graph signal processing.

2. METHOD

Let the input be observed undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with associated binary adjacency matrix $\boldsymbol{A}$, node attributes $\boldsymbol{X} \in \mathbb{R}^{n \times d_0}$ and the new nodes $\mathcal{V}^{new}$ with attributes $\boldsymbol{X}^{new}$. Our approach learns the generation of overall adjacency matrix $\boldsymbol{A}^{new}$ for $\mathcal{V} \cup \mathcal{V}^{new}$.

2.1. Proposed Approach

We start by treating incoming nodes as being added one-by-one into the graph. Let $\boldsymbol{A}_t^{\pi} \in \mathbb{R}^{t \times t}$ be the observed adjacency matrix up to the t^{th} step according to the ordering π. When the $(t+1)^{th}$ node is presented, we treat it as connected to all of previous nodes with the same probability $\tilde{p}$, where $\tilde{p}$ may reflect the overall sparsity of the graph. Hence the new candidate adjacency matrix denoted by $\tilde{\boldsymbol{A}}_{t+1}^{\pi}$ is given by

$$\begin{aligned} &(\tilde{\boldsymbol{A}}_{t+1}^{\pi})_{t+1,t+1} = 1, (\tilde{\boldsymbol{A}}_{t+1}^{\pi})_{1:t,1:t} = \boldsymbol{A}_t^{\pi}, \\ &p((\tilde{\boldsymbol{A}}_{t+1}^{\pi})_{k,t+1} = 1) = \tilde{p} \text{ for } k = 1, 2, \cdots, t. \end{aligned} \tag{1}$$

Similar to *GraphRNN*, we obtain the marginal distribution for graph by sampling the auxiliary π from the joint distribution of $p(\mathcal{G}, (\boldsymbol{A}^{\pi}, \boldsymbol{X}^{\pi}))$ with

$$p(\mathcal{G}) = \sum_{\pi} p((\boldsymbol{A}^{\pi}, \boldsymbol{X}^{\pi}) \mathbb{1}[f_G(\boldsymbol{A}^{\pi}, \boldsymbol{X}^{\pi}) = \mathcal{G}]),$$

where $f_G(\boldsymbol{A}^{\pi}, \boldsymbol{X}^{\pi})$ maps the tuple $(\boldsymbol{A}^{\pi}, \boldsymbol{X}^{\pi})$ back to a unique graph $\mathcal{G}$. Each sampled π gives a $(\boldsymbol{A}^{\pi}, \boldsymbol{X}^{\pi})$ that constitutes one-sample mini-batch that drives the stochastic gradient descent (SGD) for updating parameters.

To illustrate the sequential generation process, we decompose joint marginal log-likelihood of $(\boldsymbol{A}_{\leq n}, \boldsymbol{X}_{\leq n})$ under the node ordering π into

$$\begin{aligned} \log p(\boldsymbol{A}_{\leq n}^{\pi}, \boldsymbol{X}_{\leq n}^{\pi}) = &\sum_{i=1}^{n-1} \log p(\boldsymbol{A}_{\leq i+1}^{\pi}, \boldsymbol{X}_{\leq i+1}^{\pi} | \boldsymbol{A}_{\leq i}^{\pi}, \boldsymbol{X}_{\leq i}^{\pi}) \\ &+ \log p(\boldsymbol{A}_1^{\pi}, \boldsymbol{X}_1^{\pi}). \end{aligned} \tag{2}$$

The log-likelihood term of initial state $\log p(\boldsymbol{A}_1^{\pi}, \boldsymbol{X}_1^{\pi})$ is not of interest as we focus on modeling transition steps.

Following *VAE* framework with hidden factors as Gaussian variables, for each transition step, we use encoding distribution $q_{\phi}^i(\boldsymbol{z}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i})$, generating distribution $p_{\theta}^i(\boldsymbol{A}|\boldsymbol{z}^i)$, and conditional prior $p_0^i(\boldsymbol{z}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i})$. From now on we treat the conditional on π as implicit for simplicity of notation. The variational lower bound for each step is given by:

$$\begin{aligned} &\log p(\boldsymbol{A}_{\leq i+1}, \boldsymbol{X}_{\leq i+1} | \boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i}) \\ &\geq E_{q_{\phi}^i(\boldsymbol{z}^{i+1}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1})}[\log p_{\theta}^i(\boldsymbol{A}_{\leq i}|\boldsymbol{z}^i)] \\ &- KL(q_{\phi}^i(\boldsymbol{z}^{i+1}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1}) \| p_0^i(\boldsymbol{z}^{i+1}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i})) + C \\ &\equiv ELBO_i + C. \end{aligned} \tag{3}$$

Here $C = \log \int_{\mathcal{Z}} p_{\theta}^i(\boldsymbol{X}_{\leq i}|\boldsymbol{z}^i) q_{\phi}^i(\boldsymbol{z}^{i+1}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1}) d\nu(\boldsymbol{z})$ is the reconstruction term for node attributes, which is not our target. We will discuss the interpretation for our evidence lower bound in Section 2.2. Given that we have assumed the consistency of underlying generating mechanism, we use the same set of parameters for each step.

When formulating encoding distribution, due to the efficiency of *GCN* in node classification and linkage prediction, we adopt their convolutional layers. The two-layer encoder for the i^{th} step is then given by:

$$\begin{aligned}\boldsymbol{\mu}(\boldsymbol{z}^i|\boldsymbol{X}_{\leq i+1}) &= \hat{\tilde{\boldsymbol{A}}}_{\leq i+1}\sigma(\hat{\tilde{\boldsymbol{A}}}_{\leq i+1}\boldsymbol{X}_{\leq i+1}\boldsymbol{W}_0)\boldsymbol{W}_1,\\ \mathrm{diag}(\boldsymbol{\Sigma}(\boldsymbol{z}^i|\boldsymbol{X}_{\leq i+1}) &= \hat{\tilde{\boldsymbol{A}}}_{\leq i+1}\sigma(\hat{\tilde{\boldsymbol{A}}}_{\leq i+1}\boldsymbol{X}_{\leq i+1}\boldsymbol{W}_0)\boldsymbol{W}_2,\end{aligned} \tag{4}$$

where $\sigma(.)$ is activation function and $\hat{\tilde{\boldsymbol{A}}}$ denotes the normalized candidate adjacency matrix constructed by (1). We also adopt the pairwise inner product decoder for edge generation:

$$p_{i,j} = p(\boldsymbol{A}_{i,j} = 1|\boldsymbol{z}_i, \boldsymbol{z}_j) = f(\langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle), \tag{5}$$

with $f(.)$ being the sigmoid function. Another reason for using simple decoder being that in *VAE* framework if the generative distribution is too expressive, the latent factors are often ignored [21].

As for conditional priors of hidden factors, standard Gaussian priors are no longer suitable because we already have information from previous i nodes at the $(i+1)^{th}$ step. Hence, we use what the model has informed us till the i^{th} step in an adaptive way by treating $\boldsymbol{z}^{i+1} \in \mathbb{R}^{(i+1)\times d_2}$ as $[\boldsymbol{z}^{i+1}_{1:i}, \boldsymbol{z}^{i+1}_{i+1}]$, where $\boldsymbol{z}^{i+1}_{1:i}$ are the hidden factors for previous nodes and $\boldsymbol{z}^{i+1}_{i+1}$ is for the new node. For $\boldsymbol{z}^{i+1}_{1:i}$ we can use the encoding distribution $p_\phi(\boldsymbol{z}_{1:i}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1})$ where the candidate adjacency matrix $\tilde{\boldsymbol{A}}_{\leq i}$ passes information from previous steps. For the new node we keep using standard Gaussian prior. This gives us

$$\begin{aligned}&p^i_0(\boldsymbol{z}^{i+1}_{1:i}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i}) = p_\phi(\boldsymbol{z}_{1:i}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1})\\ &\boldsymbol{z}^{i+1}_{i+1}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i} \sim N(0, I)\end{aligned} \tag{6}$$

We use the sum of negative ELBO in each transition step as loss function ($L = -\sum_{i=1}^{n-1} ELBO_i$) and obtain optimal aggregation weights $[\boldsymbol{W}_0, \boldsymbol{W}_1, \boldsymbol{W}_2]$ by minimizing this loss.

In practice, it's not necessary to consider adding new nodes one-by-one. Instead, the new nodes can be added in a batch-wise fashion to alleviate computational costs. In preliminary experiments we also observe that sampling uniformly at random from all node permutations gives very similar results to sampling from BFS orderings, hence we report results with the uniform sampling schema.

2.2. Adaptive Evidence Lower Bound

The loss function can be rearranged into (7) (with $\beta = 1$):

$$\begin{aligned}&-\sum_{i=1}^{n-1} E_{q^i_\phi(\boldsymbol{z}^{i+1}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1})}[\log p_\theta(\boldsymbol{A}_{\leq i}|\boldsymbol{z}^i)]\\ &+\beta\sum_{i=1}^{n-1} KL(q^i_\phi(\boldsymbol{z}^{i+1}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1})\|p^i_0(\boldsymbol{z}^{i+1}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i})).\end{aligned} \tag{7}$$

The first term sums up the reconstruction loss in each generation step. The second term serves as an adaptive regularizer that enforces the posterior of latent factors for observed nodes to remain close to their priors which contain information from previous steps. This can prevent the model from overfitting the edges of the new nodes, which is quite helpful in our batched version where new edges can outnumber original edges, as we are fitting new nodes into original structure.

Similar to β-*VAE* [22], we also introduce the tuning parameter β as shown in (7) to control the tradeoff between the reconstruction term and the adaptive regularization.

3. EXPERIMENT

We test our generative graph convolution network (*G-GCN*) for growing graphs on two tasks: link prediction for isolated new nodes, and for nodes in observed graph. We use three benchmark citation network datasets: Cora, Citeseer and Pubmed. Their details are described in [23]. Node attributes are informative for all three datasets, which is indicated by the results of *GCN-VAE* in [18].

3.1. Baselines

We compare our approach against *GCN-VAE* and a multilayer perceptron *VAE* (*MLP-VAE*) [13]. Here the encoder of *MLP-VAE* is constructed by replacing the adjacency matrices in *GCN-VAE* with non-informative identity matrices. Their decoders are the same as our approach in (5). The difference is that *GCN-VAE* uses both topological information and node attributes, while *MLP-VAE* only considers node attributes. When predicting edges for isolated new nodes, for all three methods, we plug the 'candidate' adjacency matrix $\tilde{\boldsymbol{A}}$ formulated in (1) with $\tilde{p} = 0$ into the encoder-decoder frameworks and recover the true adjacency matrix.

We choose these two methods to compare with, instead of others, because both of them are able to utilize node attributes and follow from *VAE* framework. As we mentioned, most other graph embedding and graph generation techniques do not work for growing graphs without nontrivial modifications.

3.2. Experiment Setup

Link prediction for isolated new nodes

For each citation network, a growing graph is constructed by randomly sampling an observed subgraph containing 70% of all nodes. The left-out nodes are treated as isolated new nodes. The subgraph is used for training and the validation and test sets are formed by the edges between nodes in observed subgraph and the new nodes as well as the edges among the new nodes according to the original full graph. As we are treating new nodes as being added in a batch-wise fashion, the size of new node batch is set to be $\frac{\#\{\text{training nodes}\}}{3}$.

Link prediction for nodes in observed graph

Fig. 1. An illustration for the workflow of our approach at a transition step. The growing graph has an observed subgraph with three nodes (node 0, 1 and 2) and an incoming new node (node 3). The informative conditional priors for latent factors $z_{0:2}^{(3)}$ contain structural information of the observed subgraph. The encoding distributions for all latent factors are formed according to the candidate adjacency matrix where candidate edges (the dashed edges) are added to original subgraph.

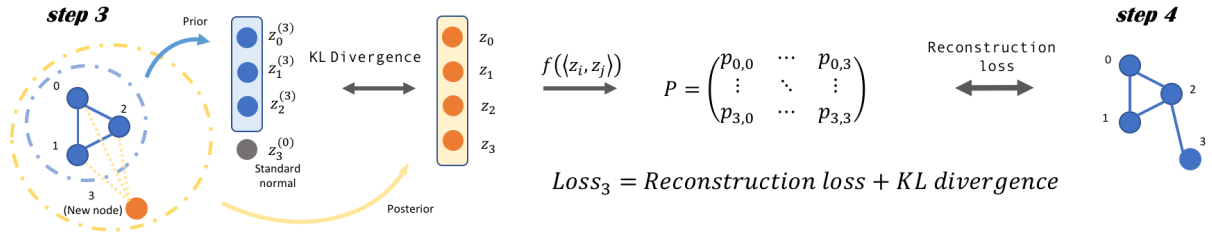

Table 1. Results for link prediction tasks in citation networks. Standard error is computed over 10 runs with random initializations on random dataset splits. The first three rows are results for the first task on new nodes, and last three rows are results for the second task on nodes in observed graph.

Method	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
Isolated new nodes						
GCN-VAE	75.12 ± 0.4	76.32 ± 0.3	79.36 ± 0.3	82.13 ± 0.1	85.52 ± 0.2	85.43 ± 0.1
MLP-VAE	75.59 ± 0.7	75.64 ± 0.5	81.76 ± 0.6	83.67 ± 0.4	77.13 ± 0.4	77.24 ± 0.3
G-GCN	**83.30** ± 0.3	**85.03** ± 0.3	**89.54** ± 0.2	**91.30** ± 0.2	**87.49** ± 0.2	**87.24** ± 0.1
Nodes in observed graph						
GCN-VAE	93.15 ± 0.4	94.42 ± 0.2	93.27 ± 0.4	94.42 ± 0.1	96.74 ± 0.4	96.94 ± 0.3
MLP-VAE	86.55 ± 0.2	87.21 ± 0.3	87.13 ± 0.2	89.34 ± 0.1	79.39 ± 0.5	79.53 ± 0.3
G-GCN	**94.07** ± 0.4	**95.15** ± 0.2	**94.62** ± 0.7	**95.93** ± 0.7	**96.96** ± 0.6	**97.27** ± 0.5

We then test our model on the original link predictions task [18], which predicts the existence of unseen edges between nodes in observed graph. In this task we adopt their experiment setup, where 10% and 5% of the edges are removed from the training graph and used as positive validation set and test set respectively. The same amount of unconnected node pairs are sampled and constitute the negative examples.

In both tasks we use a 400-dim hidden layer and 200-dim latent variables, and train for 200 iterations using the *Adam* optimizer with a learning rate of 0.001 for all methods. Notice that all three methods use encoding layers of the same form and their decoding layers are all parameter-free, so they already have the same number of parameters. The implementation of *GCN-VAE* on the second task is conducted using their official Tensor-Flow code. The rest are conducted with our own implementations with PyTorch.

3.3. Results

We report *area under the ROC curve* (AUC) and average precision (AP) scores for each model on the test sets for the two tasks (Table 1).

Firstly, our approach outperforms both baselines in new node link prediction task across all three datasets, in terms of both AUC and AP. By comparing to *MLP-VAE* we show our advantage of learning with topological information, and our better performance over *GCN-VAE* indicates the importance of modeling the sequential generating process when making predictions on new nodes.

Secondly, *G-GCN* has comparable or even slightly better results than *GCN-VAE* on link prediction task in observed graph, which suggests that our superior performance on isolated new nodes is not at the cost of the performance on nodes in observed graph. This is within expectation since our approach learns the generation process as graph structure keeps growing under our sequential training setup, where new nodes are added in each step. It targets on nodes in observed graph as well as new nodes while not overfitting either of them. In a nutshell, our approach achieves better performance on link prediction task for the growing graphs as a whole.

4. CONCLUSION AND FUTURE WORK

We propose a generative graph convolution model for growing graphs that incorporates graph representation learning and graph convolutional network into a sequential generative model. Our approach outperforms others in all benchmark datasets on link prediction for growing graphs.

However, scalability remains a major issue as the computational complexity depends on the size of full graph. The idea of localized convolution from *GraphSAGE* [12] and graph sampling from *FastGCN* [11] may have pointed out promising directions, which we leave to future work.

5. REFERENCES

[1] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia, "Learning deep generative models of graphs," *arXiv preprint arXiv:1803.03324*, 2018.

[2] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *arXiv preprint arXiv:1603.08861*, 2016.

[3] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 37–48.

[4] Mikhail Belkin and Partha Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[5] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1105–1114.

[6] 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*. ACM, 2014, pp. 701–710.

[7] 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*. ACM, 2016, pp. 855–864.

[8] 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*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

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

[10] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

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

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

[13] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[16] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec, "Graphrnn: A deep generative model for graphs," *arXiv preprint arXiv:1802.08773*, 2018.

[17] Martin Simonovsky and Nikos Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," *arXiv preprint arXiv:1802.03480*, 2018.

[18] Thomas N Kipf and Max Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[19] Aditya Grover, Aaron Zweig, and Stefano Ermon, "Graphite: Iterative generative modeling of graphs," *arXiv preprint arXiv:1803.10459*, 2018.

[20] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Systems with Applications*, vol. 41, no. 4, pp. 2065–2073, 2014.

[21] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel, "Variational lossy autoencoder," *arXiv preprint arXiv:1611.02731*, 2016.

[22] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, 2017.

[23] 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, 2008.