HOW NEURAL PROCESSES IMPROVE GRAPH LINK PREDICTION

Huidong Liang and Junbin Gao

Discipline of Business Analytics
The University of Sydney, Australia

ABSTRACT

We propose a meta-learning approach with graph neural networks for link prediction: Neural Processes for Graph Neural Networks (NPGNN), which can not only perform both transductive and inductive learning tasks, but also generalize well when only training on a small subgraph. The key idea is to assume the node embeddings follow a *Gaussian Process* parameterised by graph neural networks and then use a meta-learning framework to pass information from the subgraph to the complete graph. Experiments on real-world citation networks are conducted to validate our model, where the results suggest that the proposed method achieves stronger performance compared to other state-of-the-art models.

***Index Terms*—** Link Prediction, Inductive Learning, Neural Processes, Graph Neural Networks

1. INTRODUCTION

Graph, consisting of a set of nodes and links, is a common but special data structure in our daily life. With the advancement in machine learning, designing algorithms for problems with graph-type datasets has been successful in many real-world applications [1]. Link prediction, in which the goal is to predict some unknown links in a graph given other links and nodes [2], is one of the important tasks in graph machine learning. Relevant applications of link prediction involve many areas: in recommendation systems such as friend recommendation [3], movie recommendation [4], and citation recommendation for academic papers [5]; in knowledge discovery in databases (KDD) such as knowledge graph completion [6] and social network analysis [7]; and in health science research such as drug-target interaction [8] and metabolic network reconstruction [9].

Generally, there are two types of link prediction tasks: transductive link prediction and inductive link prediction, as illustrated in Fig. 1. For transductive link prediction, all the nodes information with most of the links are known in training, and the goal is to predict the unknown links in the entire graph. Whereas for inductive link prediction, a small proportion of nodes are not seen when building up the model, and the remaining nodes with their corresponding links information are used for training. At prediction time, this small proportion of nodes will join the graph, and the goal is to infer the unknown links in the entire new graph. In this work, we further consider a more challenging scenario: *few-shot inductive link prediction*, where only a small proportion of nodes and links are available for training, and the task is to predict connectivity of the entire graph.

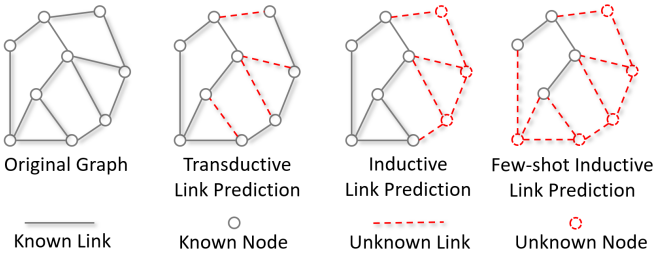

Fig. 1. Illustration on transductive, inductive, and few-shot inductive link predictions.

Currently, the majority of the literature focuses on transductive link prediction, represented by embedding-based approaches such as DeepWalk (DW) [10] and Spectral Clustering (SC) [11]. These methods utilize dimensionality reduction techniques to generate a low-dimensional vector representation for each node's high-dimensional feature in the graph, such that useful information can be exploited efficiently. Another approach with graph embedding, Variational Graph Autoencoders (VGAE) [12], also considers neighbours' information when generating latent representations for nodes via Graph Convolutional Networks (GCN) [13], and has shown strong performance on various datasets for link prediction. Based on VGAE, Adversarially Regularized Graph Autoencoder (ARGA) [14] further proposed an adversarial framework to regularize the latent embedding's distribution and achieves better results. Nevertheless, in many real-world problems, the size of the graph is growing over time, which requires generating embeddings for new nodes (such as new customers and products in a recommendation system) based on the existing graph and hence need to perform inductive link prediction.

Neural Processes [15], which utilizes *meta-learning* and assumes the observation's latent representation follows a *Gaussian Process* parameterised by neural networks, provides a bridge that connects patterns in the existing data and

new data. Previous literature investigated the application of Conditional Neural Processes [16] (a non-generative approach) on graphs in edge imputation [17], in this paper, we will use Neural Processes framework to perform transductive and inductive link prediction from a generative perspective. Our main contributions can be summarized as follows:

- We propose a novel framework: Neural Processes for Graph Neural Network (NPGNN) that uses *Gaussian Processes* and GCN to transfer the topological information from subgraph to complete graph.
- We use transductive and inductive link prediction tasks to demonstrate the performance of our method and compare it against some state-of-the-art algorithms.
- A few-shot inductive experiment is further conducted to validate the generalization ability of our method.

2. PROPOSED METHOD: NPGNN

2.1. Setup and Framework

First, we define an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{A})$, where $\mathbf{V}$ is a set of nodes with features $\mathbf{x}_i \in \mathbf{X}$, and $\mathbf{A}$ is the adjacency matrix with $\mathbf{A} = 1$ if there is a link between $\mathbf{v}_i$ and $\mathbf{v}_j$, and $\mathbf{A}_{ij} = 0$ otherwise. Then we randomly select a subset of nodes $\mathbf{V_C} \in \mathbf{V}$ with their related features $\mathbf{X_C} \in \mathbf{X}$ and adjacency matrix $\mathbf{A_C}$ to construct a context subgraph $\mathbf{G_C}$. We assume there are n nodes in the complete graph $\mathbf{G}$, and the first m nodes are the context nodes $\mathbf{V_C}$. Our goal is to model the adjacency matrix $\mathbf{A}$ for the complete graph $\mathbf{G}$ conditional on the context subgraph $\mathbf{G_C}$, as illustrated in Fig. 2.

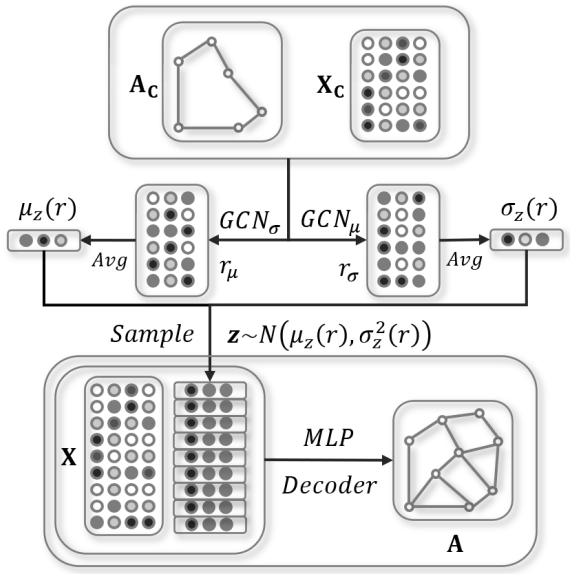

Fig. 2. Framework of Neural Processes for Graph Neural Networks

2.2. Graph Convolutional Encoder

We first use two GCNs to encode the context subgraph $\mathbf{G_C}$ into latent representation $\mathbf{r}_{\mathbf{c}_i} \in \mathbb{R}^d$ for each context node $\mathbf{v}_i \in \mathbf{V_C}$, and assume the latent embeddings follow *Gaussian* distribution $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{c}_i}, \text{diag}(\boldsymbol{\sigma}^2_{\mathbf{c}_i}))$:

$$\boldsymbol{\mu}^{(\ell)}_{\mathbf{c}} = \text{ReLU}(\tilde{\mathbf{D}}^{-\frac{1}{2}}_{\mathbf{C}} \tilde{\mathbf{A}}_{\mathbf{C}} \tilde{\mathbf{D}}^{-\frac{1}{2}}_{\mathbf{C}} \mathbf{Z}^{(\ell-1)}_{\mathbf{C}_\mu} \mathbf{W}^{(\ell)}_{\boldsymbol{\mu}}), \quad (1)$$

$$\log \boldsymbol{\sigma}^{(\ell)}_{\mathbf{c}} = \text{ReLU}(\tilde{\mathbf{D}}^{-\frac{1}{2}}_{\mathbf{C}} \tilde{\mathbf{A}}_{\mathbf{C}} \tilde{\mathbf{D}}^{-\frac{1}{2}}_{\mathbf{C}} \mathbf{Z}^{(\ell-1)}_{\mathbf{C}_\sigma} \mathbf{W}^{(\ell)}_{\boldsymbol{\sigma}}), \quad (2)$$

where $\tilde{\mathbf{D}}_{\mathbf{C}_{ii}} = \sum_j \tilde{\mathbf{A}}_{\mathbf{C}_{ij}}$ is the degree matrix of $\tilde{\mathbf{A}}_{\mathbf{C}}$, and $\tilde{\mathbf{A}}_{\mathbf{C}} = \mathbf{A_C} + \mathbf{I}$. Both Eq. (1) and Eq. (2) share the same parameters in their first layer (i.e. $\mathbf{W}^{(0)}_{\boldsymbol{\mu}} = \mathbf{W}^{(0)}_{\boldsymbol{\sigma}}$) with $\mathbf{Z}^{(0)}_{\mathbf{C}_\sigma} = \mathbf{Z}^{(0)}_{\mathbf{C}_\mu} = \mathbf{X_C}$, and use $\text{ReLU}(t) = \max(0, t)$ as the activation function.

2.3. Aggregation and Gaussian Processes

We assume node embeddings on the subgraph $\mathbf{G_C}$ follow a *Gaussian Process* $\mathcal{GP}\big(\boldsymbol{\mu}_{\mathbf{z}}(\mathbf{r_c}), \text{diag}(\boldsymbol{\sigma}^2_{\mathbf{z}}(\mathbf{r_c}))\big)$, where mean and kernel functions are formulated by aggregation over the latent embeddings (here we use average):

$$\boldsymbol{\mu}_{\mathbf{z}}(\mathbf{r_c}) = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\mu}_{\mathbf{c}_i}, \qquad \log \boldsymbol{\sigma}_{\mathbf{z}}(\mathbf{r_c}) = \frac{1}{m}\sum_{i=1}^{m} \log \boldsymbol{\sigma}_{\mathbf{c}_i}. \quad (3)$$

Then we can define a global latent variable $\mathbf{z}$ with distribution $q(\mathbf{z}|\mathbf{A_C}, \mathbf{X_C}) = \mathcal{N}\big(\boldsymbol{\mu}_{\mathbf{z}}(\mathbf{r_c}), \text{diag}(\boldsymbol{\sigma}^2_{\mathbf{z}}(\mathbf{r_c}))\big)$, which contains the topological structure of subgraph $\mathbf{G_C}$. If we assume the latent representations on subgraph $\mathbf{G_C}$ and complete graph $\mathbf{G}$ follow the same *Gaussian Process*, then we can use this $\mathbf{z}$ to deduce information about the complete graph $\mathbf{G}$ from subgraph $\mathbf{G_C}$.

2.4. Multilayer Perceptron with Inner Product Decoder

After aggregation, we combine a sampled $\mathbf{z}$ with every feature $\mathbf{x}_i \in \mathbf{X}$ as $\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}$ (i.e. $\tilde{\mathbf{x}}_i = [\mathbf{x}_i^\top \ \mathbf{z}^\top]^\top$). As such, the topological information from context subgraph $\mathbf{G_C}$ flows to the complete graph $\mathbf{G}$ via this latent variable $\mathbf{z}$. Then we send them to an MLP decoder to produce the latent embedding $\mathbf{u}_i \in \mathbf{U}$ for nodes on the complete graph $\mathbf{G}$ with sigmoid activation function:

$$\mathbf{U}^{(\ell)} = \sigma(\mathbf{U}^{(\ell-1)}\mathbf{W}^{(\ell)} + \mathbf{b}^{(\ell)}), \quad (4)$$

where $\mathbf{U}^{(0)} = \tilde{\mathbf{X}}$ and $\sigma(t) = 1/(1+\exp(-t))$.

Finally, we take inner product for each pair of $(\mathbf{u}_i, \mathbf{u}_j)$, and use sigmoid function to calculate the probability of edge existence between two nodes $\mathbf{v}_i, \mathbf{v}_j \in \mathbf{V}$, as well as the likelihood $p(\mathbf{A}|\mathbf{X}, \mathbf{z})$ for reconstructing $\mathbf{A}$ (i.e. similarity matrix):

$$p(\mathbf{A}|\mathbf{X}, \mathbf{z}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(\mathbf{A}_{ij}|\mathbf{u}_i, \mathbf{u}_j), \quad (5)$$

$$p(\mathbf{A}_{ij} = 1|\mathbf{u}_i, \mathbf{u}_j) = \sigma(\mathbf{u}_i^\top \mathbf{u}_j). \quad (6)$$

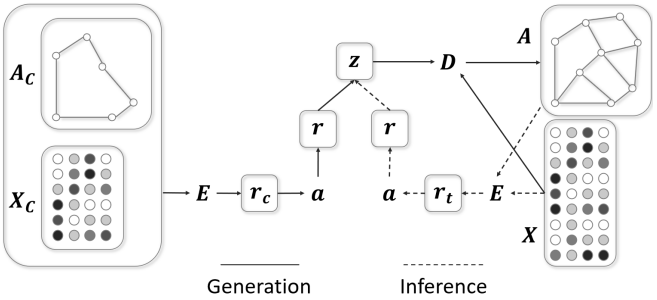

Fig. 3. Inference of NPGNN. Explanation for notations in the figure: E - encoder, D - decoder and a - aggregator.

2.5. Inference and Learning

Inference for our model is carried out by *Variational Inference* and is demonstrated in Fig. 3.

In *Bayesian Inference* under our context, we wish to estimate the *posterior* distribution of the global latent variable $p(\mathbf{z}|\mathbf{A},\mathbf{X})$ defined as follows:

$$p(\mathbf{z}|\mathbf{A},\mathbf{X}) = \frac{p(\mathbf{A}|\mathbf{X},\mathbf{z})p(\mathbf{z})}{p(\mathbf{A})}, \tag{7}$$

where $p(\mathbf{z})$ is a certain *prior* distribution and $p(\mathbf{A})$ is the *marginal likelihood* or *evidence*.

The objective in *Variational Inference* is to approximate $p(\mathbf{z}|\mathbf{A},\mathbf{X})$ by a variational distribution $q(\mathbf{z}|\mathbf{A},\mathbf{X})$ that maximizes the *variational lower bound* $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}|\mathbf{A},\mathbf{X})}[\log p(\mathbf{A}|\mathbf{X},\mathbf{z})] - \mathrm{KL}[q(\mathbf{z}|\mathbf{A},\mathbf{X})||p(\mathbf{z})],$$

where $\mathrm{KL}[\cdot||\cdot]$ is the Kullback–Leibler divergence, $q(\mathbf{z}|\mathbf{A},\mathbf{X})$ is parameterised by the encoder in Eq. (1) and Eq. (2), and $p(\mathbf{A}|\mathbf{X},\mathbf{z})$ is parameterised by our decoder model in Eq. (5).

Now, instead of choosing a standard *Gaussian* as the prior $p(\mathbf{z})$, we choose: $p(\mathbf{z}) = q(\mathbf{z}|\mathbf{A_C},\mathbf{X_C})$, which means we use the variational distribution inferred from the random subgraph $\mathbf{G_C}$ as the prior information. Therefore, maximizing the variational lower bound $\mathcal{L}$ is effectively maximizing the cross-entropy loss of reconstructing $\mathbf{A}$, and meanwhile penalizing the information inferred by subgraph $\mathbf{G_C}$ for deviating from the information inferred by the complete graph $\mathbf{G}$. In other words, we are forcing the *Gaussian Processes* of the latent node embeddings over $\mathbf{G_C}$ and $\mathbf{G}$ to be identical.

Note to calculate $q(\mathbf{z}|\mathbf{A},\mathbf{X})$, we only need to substitute $\mathbf{A_C}$ and $\mathbf{X_C}$ by $\mathbf{A}$ and $\mathbf{X}$ in the encoder, which will return us $q(\mathbf{z}|\mathbf{A},\mathbf{X}) = \mathcal{N}\big(\boldsymbol{\mu}_\mathbf{z}(\mathbf{r_t}), \mathrm{diag}(\boldsymbol{\sigma}^2_\mathbf{z}(\mathbf{r_t}))\big)$, with $\mathbf{r_t}$ being the latent embeddings for the complete graph $\mathbf{G}$ (at training stage we have full information of $\mathbf{G}$). At prediction time, the original complete graph $\mathbf{G}$ for training becomes the "subgraph $\mathbf{G_C}$" for context information such that we can infer information on the "bigger" graph for prediction. Carrying such an objective, we will use $\mathbf{G}$ to generate $\mathbf{z}$ in the forward pass. The algorithm for learning is summarised in Algorithm 1.

Algorithm 1 Neural Processes for Graph Neural Networks

Input: Complete Graph $\mathbf{G} = (\mathbf{V},\mathbf{A})$ with features $\mathbf{X}$
Encoder $Encode(\cdot,\cdot)$; Decoder $Decode(\cdot,\cdot)$
Aggregation $Aggregate(\cdot)$; Total iteration T
Output: Optimized $Encode(\cdot,\cdot)$ and $Decode(\cdot,\cdot)$.

for iteration $= 0, 1, 2, ..., T$ **do**
  Generate a random subgraph $\mathbf{G_C}$ with $\mathbf{A_C}$ and $\mathbf{X_C}$
  $\mathbf{r_c} \leftarrow Encode(\mathbf{A_C},\mathbf{X_C})$ $\mathbf{r_t} \leftarrow Encode(\mathbf{A},\mathbf{X})$
  $\mathbf{r'} \leftarrow Aggregate(\mathbf{r_c})$ $\mathbf{r} \leftarrow Aggregate(\mathbf{r_t})$
  Compute $q(\mathbf{z}|\mathbf{A_C},\mathbf{X_C}) = \mathcal{N}\big(\boldsymbol{\mu}_\mathbf{z}(\mathbf{r'}), \mathrm{diag}(\boldsymbol{\sigma}^2_\mathbf{z}(\mathbf{r'}))\big)$
  Compute $q(\mathbf{z}|\mathbf{A},\mathbf{X}) = \mathcal{N}\big(\boldsymbol{\mu}_\mathbf{z}(\mathbf{r}), \mathrm{diag}(\boldsymbol{\sigma}^2_\mathbf{z}(\mathbf{r}))\big)$
  Sample a number of L $\{\mathbf{z}^l\}_{l=1}^L \sim q(\mathbf{z}|\mathbf{A},\mathbf{X})$
  $p(\mathbf{A}|\mathbf{X},\mathbf{z}) \leftarrow Decode(\mathbf{X},\mathbf{z})$
  Compute $\nabla\mathcal{L}$ for $\mathcal{L} = \frac{1}{L}\sum_{l=1}^L \log p(\mathbf{A}|\mathbf{X},\mathbf{z}^{(l)}) - \mathbf{KL}[q(\mathbf{z}|\mathbf{A_C},\mathbf{X_C})||q(\mathbf{z}|\mathbf{A},\mathbf{X})]$
  Update $Encode(\cdot,\cdot)$ and $Decode(\cdot,\cdot)$ when optimizing $\mathcal{L}$ with $\nabla\mathcal{L}$
end for
return $Encode(\cdot,\cdot)$ and $Decode(\cdot,\cdot)$

3. EXPERIMENTS

3.1. Experimental Set-up

3.1.1. Dataset, metrics and code:

We conduct transductive link prediction and inductive link prediction to validate our model's performance on three popular citation networks: Cora, Citeseer and PubMed, as illustrated in Table 1. We will also test our model's generalizability in a small sample size scenario on Cora and Citeseer. AUC score (the Area Under the receiver operating characteristic Curve) and AP score (Average Precision) are selected as the metrics for each experiment, where we repeat the algorithm ten times with different random seeds to report their means and standard errors.[1]

Table 1. Dataset Statistics

Dataset	# Nodes	# Links	# Features	# Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
PubMed	19,717	44,338	500	3

3.1.2. Baselines and hyper-parameters settings:

We compare our model's performance against other state-of-the-art embedding-based methods: Graph Autoencoder (GAE), Variational Graph Autoencoder (VGAE) [12], Adversarially Regularized Graph Autoencoder (ARGA), and ARGA's variational version ARVGA [14]. For a fair comparison purpose, we choose our model size similar to ARGA

[1]Code for replicating the experiment results can be found at `https://github.com/LeonResearch/NPGNN`.

Table 2. Transductive link prediction results

Method	Cora		Citeseer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
GAE	91.0 (0.02)	92.0 (0.03)	89.5 (0.04)	89.9 (0.05)	96.4 (0.00)	96.5 (0.00)
VGAE	91.4 (0.01)	92.6 (0.01)	90.8 (0.02)	92.0 (0.02)	94.4 (0.02)	94.7 (0.02)
ARGA	92.4 (0.003)	93.2 (0.003)	91.9 (0.003)	93.0 (0.003)	**96.8 (0.001)**	**97.1 (0.001)**
ARVGA	92.4 (0.004)	92.6 (0.004)	92.4 (0.003)	93.0 (0.003)	96.5 (0.001)	96.8 (0.001)
NPGNN	**93.1 (0.003)**	**93.7 (0.004)**	**94.0 (0.004)**	**95.0 (0.002)**	95.3 (0.001)	95.2 (0.001)

Table 3. Inductive link prediction results

Method	Cora		Citeseer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
GAE	73.7 (0.025)	70.5 (0.024)	80.0 (0.019)	78.2 (0.021)	84.3 (0.008)	81.2 (0.007)
VGAE	77.6 (0.025)	73.2 (0.018)	82.2 (0.021)	79.5 (0.028)	84.3 (0.006)	80.8 (0.007)
ARGA	74.8 (0.016)	72.7 (0.011)	78.5 (0.020)	76.7 (0.014)	83.9 (0.014)	80.8 (0.015)
ARVGA	70.2 (0.015)	69.1 (0.021)	71.6 (0.014)	70.8 (0.012)	77.0 (0.007)	73.2 (0.007)
NPGNN	**85.0 (0.024)**	**85.9 (0.024)**	**91.0 (0.012)**	**91.8 (0.013)**	**94.0 (0.001)**	**94.0 (0.001)**

[14]: the encoder contains 32 neurons in the first encoding layer and 16 neurons in the second embedding layer; and the decoder contains two hidden layers with 64 neurons and 16 neurons respectively. We train our model 400 iterations on Cora and Citeseer by Adam optimizer [18] with a learning rate of 0.01. Since the PubMed dataset is relatively large, we train our model for 5,000 iterations with a learning rate of 0.05. For other baseline models, we maintain the settings in the corresponding papers.

3.2. Transductive Experiment

We maintain the setting used in VGAE [12] that randomly masks 10% edges for testing, 5% edges for validation, and the rest for training. During optimization, 10% of the training edges are randomly selected to construct the subgraph $\mathbf{G_C}$.

The results are summarized in Table 2. It shows that by incorporating Neural Processes framework into GCN, our model outperforms other GCN-related baselines on Cora and Citeseer, and achieves competitive results with ARGA on PubMed. One possible reason would be the patterns in Cora and Citeseer are relatively "sophisticated" compared to patterns in PubMed, since Cora and Citeseer have seven and six classes respectively, whereas PubMed has only three classes. As in transductive task all the nodes' information are known during training, the extra flexibility of *Gaussian Processes* could be penalized for capturing unnecessary details.

3.3. Inductive Experiment

For inductive link prediction, we randomly select 5% nodes and use the links adjacent to these nodes for testing (roughly 10% of all links in Cora, 7% in Citeseer, and 11% in PubMed), and use the same method to select 2.5% nodes with their corresponding links for validation. During optimization, the subgraph $\mathbf{G_C}$ is also constructed inductively by randomly selecting 10% training nodes and the exclusive links among them. The results are summarized in Table 3. It shows that our proposed method outperforms the other baselines on all three datasets by a significant margin. As discussed earlier, this is because under inductive learning, new nodes are not seen during training, hence their topological structure can not be exploited by GCN. On the other hand, NPGNN takes advantage of the Neural Processes framework to transfer such information from the training "subgraph" to the testing "complete graph", thus leading to better performance.

Table 4. Few-shot inductive link prediction results

Training Graph Size	Cora		Citeseer	
	AUC	AP	AUC	AP
30% nodes (10% links)	74.8 (0.005)	76.6 (0.007)	84.0 (0.006)	85.6 (0.006)
50% nodes (25% links)	78.8 (0.009)	80.4 (0.009)	87.3 (0.004)	88.7 (0.003)
70% nodes (50% links)	81.9 (0.011)	83.3 (0.012)	89.4 (0.006)	90.6 (0.005)

To test our model's generalizability, we further reduce the number of training nodes to three different proportions: 70%, 50% and 30%, which contains 50%, 25% and 10% links respectively, and use the rest for testing. As shown in Table 4, when NPGNN is trained on 30% nodes with 10% links, it still achieves decent results on Cora and Citeseer, which are even generally better than other baselines trained on 85% nodes with over 80% links.

4. CONCLUSION

In this paper, we introduced a novel approach, Neural Processes for Graph Neural Networks, that generates a global latent embedding to connect subgraph and complete graph via *Gaussian Processes*, which enables graph neural networks to perform both transductive and inductive learning. We also showed our proposed model experimentally on different real-world graphs for three types of link prediction, where NPGNN achieves strong performance compared with state-of-art models and generalizes well when only training on a relatively small subgraph.

5. REFERENCES

[1] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, 2020.

[2] David Liben-Nowell and Jon Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[3] Lada A Adamic and Eytan Adar, "Friends and neighbors on the web," *Social Networks*, vol. 25, no. 3, pp. 211–230, 2003.

[4] Yehuda Koren, Robert Bell, and Chris Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[5] Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar, "Content-based citation recommendation," *arXiv:1802.08301*, 2018.

[6] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.

[7] Pinghua Xu, Wenbin Hu, Jia Wu, and Bo Du, "Link prediction with signed latent factors in signed social networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1046–1054.

[8] Yiding Lu, Yufan Guo, and Anna Korhonen, "Link prediction in drug-target interactions network using similarity indices. . ;18(1):39. published 2017 jan 17. doi:10.1186/s12859-017-1460-z," *BMC Bioinformatics*, vol. 18, no. 1, pp. 39, Jan 2017.

[9] Tolutola Oyetunde, Muhan Zhang, Yixin Chen, Yinjie Tang, and Cynthia Lo, "Boostgapfill: improving the fidelity of metabolic network reconstructions through integrated constraint and pattern-based methods," *Bioinformatics*, vol. 33, no. 4, pp. 608–611, 2017.

[10] 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*, 2014, pp. 701–710.

[11] Lei Tang and Huan Liu, "Leveraging social media networks for classification," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 447–478, 2011.

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

[13] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[14] S Pan, R Hu, G Long, J Jiang, L Yao, and C Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *International Joint Conference on Artificial Intelligence*, 2018.

[15] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh, "Neural processes," *arXiv:1807.01622*, 2018.

[16] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami, "Conditional neural processes," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1704–1713.

[17] Andrew Carr and David Wingate, "Graph neural processes: Towards bayesian graph neural networks," *arXiv:1902.10042*, 2019.

[18] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2014.