A Comparative Study of Graph Representations for Few-Shot Link Prediction

Abdalgader Abubaker Alsiddig

African Institute for Mathematical Sciences (AIMS)
Kigali, Rwanda
abdalgader.a.a.hassan@gmail.com

Abstract

In this work, we introduce a comparative and ablative study on link prediction task on graphs. Our work focuses on applications where we have access to multiple and sparse graphs from the same application domain. In order to tackle this task, meta-learning gives recipes of how to extract mutual information from prior experiences to fastly adapt to a new sparse graph that has never seen before. We study numerous methods learning how to construct a well-defined graph representation to ease predicting links between sparse graph's entities. Moreover, we compare those meta-learning methods with *Meta-Graph* — a gradient-based meta-learning approach for the few-shot link prediction. Our experiments on Protein-Protein Interaction (PPI) dataset show that our best ablated-method (Meta-GAE) performs better than Meta-Graph with, on average, up to 1.4% over all different density edges ratios. [1]

1 Introduction

Learning or predicting information from a huge amount of data with very complex relationships is a stumbling block for human intelligence. Even it could be impossible when talking about millions or even billions of data that acquired complex links between its components. In many cases, there is an abundant amount of data in the world with very complex relationships, so using graph-structured data becomes necessary and ubiquitous, especially in numerous domains such as social or biological networks. The graphs are extremely useful to describe these complex systems by structure the data features as nodes and the relations between these nodes as edges. A good representation of the graph's information on another space where the model can distinguish the node features and relations will highly contribute to solving particular machine learning tasks on graph data. One of the most popular graph tasks is *link prediction* or *relation prediction*, where the goal is inferring the missing links between the nodes. For instance, in biological networks, we might want to predict whether there is a side-effect between two kind of drugs [11] or infer the proteins interactions in specific human tissues [12] as shown in Figure(1-A) – this will is be our main application in this research. Moreover, in social networks, we may use the link prediction to recommend some content to users [10] or suggest friendships and homophily in social media [1] as shown in Figure(1-B).

Previous work on link prediction task assume that the task is performed only on single graph and that graph is relatively dense which means more than half of the relations or links are observed during training. We focus in the application where we have an access to multiple graphs, and some of these graphs are noisy and sparse which mean that we observe only less than 30% of the edges during the training. We need a learning paradigm that can leverage information across the graphs in

[1]Code available upon request on `http://github/abdalgader-a`

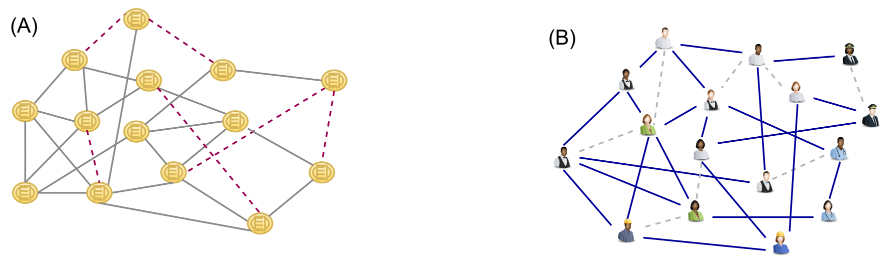

Figure 1: (A) Standard link prediction in application of proteins interactions, (B) Friendship prediction in social networks.

the domain to perform well on missing link prediction task when few edges are observed. We use *meta-learning* or *learn how to learn* [2] [8] as main framework to extract a mutual information and transfer knowledge across multiple graphs that could be useful to infer unknown relationships on new sparse graph that come from the same domain of training graphs. This problem can be views as *few-shot link predict prediction* [3]. In this problem, each graph can be viewed as individual task where the goal is to have a link prediction model can fastly adapt to new sparse graph that has not been seen before.

In this work, we study different approaches of graph representation for link prediction task when we have access to only few information about relations between graph's entities. Also, we ablate and modify a gradient-based meta-learning framework called *Meta-Graph* [3] that use Graph Neural Networks (GNNs) as link prediction model to leverage training data from multiple graphs. This ablation on Meta-Graph is constructed by eliminate what called *graph signature* and replace the architecture of Variational Graph AutoEncoder (VGAE) [7] with simple Graph AutoEncoder (GAE) [6]. Furthermore, we study how well representation can be gotten when Dynamic Graph [9] used as an encoder in the architecture instead of the message-passing based GNNs [5]. We mainly focus on biological networks application, and more precisely, on proteins interaction in the human tissues to infer whether there is interaction between two proteins or not. We got improvement with average 1.4% in AUC better than *cleaned* Meta-Graph on the PPI benchmark.

2 Background and Related Works

Definitions Given a multiple graphs dataset $\mathcal{G}$ of specific domain $\mathcal{P}(\mathcal{G})$, we have a portion called *meta-training* dataset $\mathcal{G}_{train} = \{G_1, G_2, ..., G_N\}$ that contain N different graphs, where all graphs come from the same distribution $\mathcal{P}(\mathcal{G})$. In mathematical words, $G_i \sim \mathcal{P}(\mathcal{G})$ where each simple graph $G_i = (V_i, E_i, X_i)$ define as set of node indexes V_i, where every node v is associated with real-value vector $x_v \in \mathbb{R}^d$ that represent the feature of node v, d is the node feature dimension. All node features in the graph G_i are gathered together in one matrix $X_i \in \mathbb{R}^{|V_i| \times d}$. Also define a set of edges $E_i \subseteq V_i \times V_i$ as a a binary relation that connect pair of nodes— also the relations can be represented through the adjacency matrix A. We denote to the edge that going form node $u \in V_i$ to node $v \in V_i$ by the order pair $(u, v) \in E_i$. We assume that all the graphs are simple that means there is at most one edge that link between the pair of nodes. Moreover, all the edges are undirected and without self-edge that point from the node to itself. All the previous assumptions are also applied to the other portions, *meta-testing* dataset $\mathcal{G}_{test} = \{G'_1, G'_2, ..., G'_T\}$, where T is number of graphs this dataset, and *meta-validation* dataset $\mathcal{G}_{val}$. Moreover, if we have, in general, a graph $G = (V, E, X)$, the edges are splitted into train edges, validation edges, and test edges, $E = (E_{train}, E_{val}, E_{test})$ respectively, and the model is trained on a subgraph $G_{e-train} = (V, E_{train}, X)$ to get a well-represent embedding of the node features together with the edge information to predict the edges in the test subgraph $G_{e-test} = (V, E_{test}, X)$.

Link Prediction Problem Formulation our problem is to predict the missing edges on a sparse graph that has not been seen during training. Mathematically, we want to find $p(E^*|X, E)$ of the missing edges E^* given graph information (i.e., node features and ground-truth edges).

Meta-Learning on Link Predication in order to leverage mutual information across the multiple graphs, the model parameters are *pre-trained* on meta-training dataset $\mathcal{G}_{train}$ sampled from specific domain $\mathcal{P}(\mathcal{G})$. More precisely, we pre-train the model on the subgraph $G_{e-train}$ of all G belong to the meta-training dataset $\mathcal{G}_{train}$. When we want to predict the missing edges on unseen graph G' from meta-testing dataset $\mathcal{G}_{test}$, we *fine-tune* the model parameters on the subgraphs $G'_{e-train}$ of all $G' \in \mathcal{G}_{test}$. We can noticed that we have two level of splitting, on graphs level, the full dataset is splitted into meta-training $\mathcal{G}_{train}$, meta-validation $\mathcal{G}_{val}$, and meta-testing $\mathcal{G}_{test}$ datasets. While within each single graph G the edges are splitted into training edges E_{train}, validation edges E_{val} , and testing edges E_{test} that form the subgraphs $G_{e-train}$, G_{e-val}, and G_{e-test} respectively.

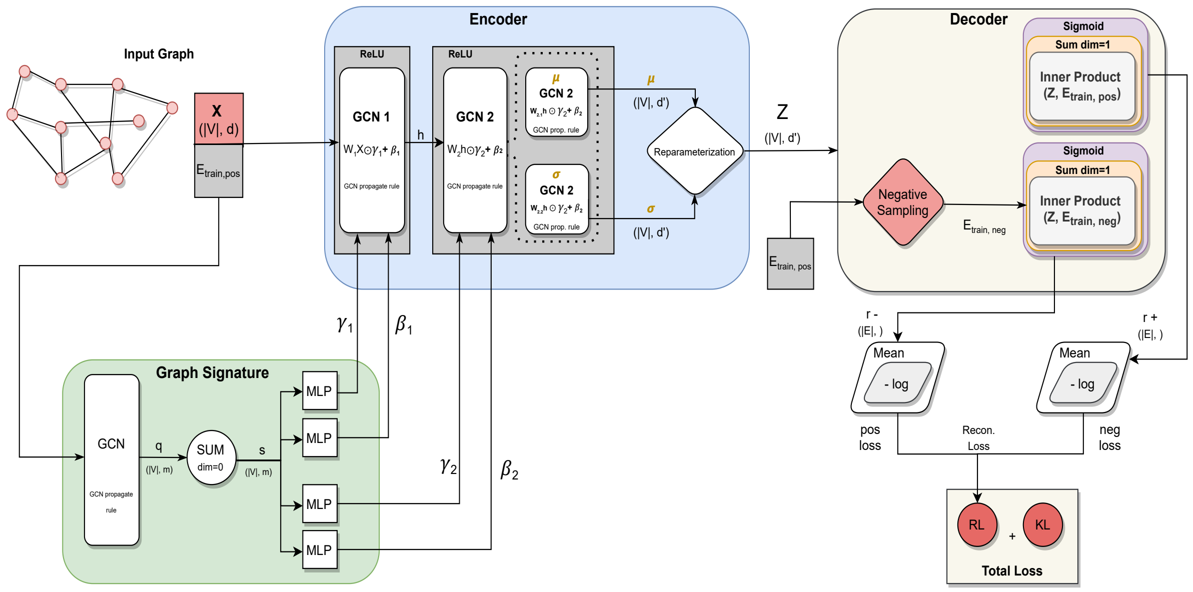

Figure 2: Architecture of Meta-Graph.

2.1 Meta-Graph

In this part, we demonstrate the Meta-Graph [3] — the method that we ablate and modify its architecture — and we illustrate, in details, the main components that form the full architecture. Meta-Graph is a gradient-based meta-learning approach to tackle the problem of the link prediction task for sparse graphs through leveraging mutual information from multiple graphs of the same domain. This approach builds on the VGAE as the base graph neural networks model and combined with Model Agnostic Meta-Learning (MAML) proposed by Finn [4] as gradient-based meta-learning. Meta-Graph can be viewed as adaptation of MAML for link prediction setup, where it uses second-order gradient descent to optimize the initialization of the global parameter for the VGAE. Additionally, the model contains a function called *Graph Signature* (GS) that map the input graph to a graph-related values called *signature* which is used for modulate the initialization of the GNN layers-parameters. In other words, Meta-Graph is not more than VGAE with adapted MAML in addition to the signature function to optimize and modulate the initialization of the global parameter respectively. The architecture can be broke down into two main components:

- **VGAE** is the base link prediction model for any input graph $G = (V, E, X)$ that composed from *encoder* and *decoder*. The encoder maps the node v to the corresponding low-dimensional embedding z_v by try to learn an:

-Inference model q_ϕ , that defines a distribution $q_\phi(Z|E, X)$ over the graph node embeddings, where ϕ is the model's global parameter and $Z \in \mathbb{R}^{|V| \times d'}$ is the node embeddings tensor with d' embedding length. Each node embedding use to score the likelihood of a link existing between the node itself and each other nodes in the graph. Because the variationality, the node embeddings are sampled from the posterior normal distribution $q_\phi(z_v|E, X) = \mathcal{N}(z_v|\mu, diag(\sigma^2))$, where z_v is the sampled node embedding of node $v \in V$. The normal distribution parameters μ and σ are learned via GNNs paradaigm with the form of neural message passing. The network layers are use the notion of Graph Convolutional Networks (GCNs) [6] to embed the node features through out the aggregation using the edges information. Figure 2 shows the input graph fed into the encoder as well as into graph signature (explained below) at the same time.

Thereafter, along with the edges information, the node embeddings are fed into the *decoder* that define a:

-Generative component, that compute the likelihood of a link existing between pair of nodes which is proportional to the dot product of their node embeddings. The dot product is computed for the node embeddings that has *positive edges* between them — i.e the ground-truth edges between the nodes in the graph. Using these positive edges, the negative sampler generate *negative edges* samples of the graph with same dimension as positive edges to compute the dot product of the corresponding node embeddings for each negative pair of nodes. Mathematically, for each two pair of nodes u and v that either connected positively or negatively, the generative component of the VGAE is then defined as

$$p(E|Z) = \prod_u \prod_v p(E_{u,v}|z_u, z_v), \qquad with \quad p(E_{u,v}|z_u, z_v) = \sigma(z_u^T z_v) \tag{1}$$

Then, the inference model q_ϕ is trained to minimize the variation lower bound on training data.

$$\mathcal{L}_G = \mathbf{E}_{q_\phi}[log\, p(E^{train}|Z)] - KL[q_\phi(Z|X, E^{train})||p(z)] \tag{2}$$

where, $p(z)$ is a Gaussian prior.

- **Graph Signature** $s_G = \psi(G)$, is a function that capture structure of each graph by feed the input graph to GCN, then to multilayer perceptrons (MLPs) that gives values that used to modulate parameters initialization of the inference model ϕ as shown in Figure 2.

3 Methods

Our end-goal for graph representation is to find an architecture that gives a well-representing embedding of the graph information that suite for our link prediction task. In this research, we studied different approaches for representing graphs and compare it with Meta-Graph approach for link prediction task. In this section, we illustrate the architecture of each approach and in the following section we show how we train the models.

- **Meta-GAE + Signature.** Similar to Meta-Graph, it is a gradient-based meta-learning approach (MAML-based) where the architecture's backbone is simple GAE. The only difference is that there is no variational sampling of the graph embedding from normal distribution at the encoder part. The graph signature as in the Meta-Graph is used for modulate parameter initialization of the inference model. Meta-GAE+Signature has almost same performance in compare with the reproduced results from Meta-Graph.

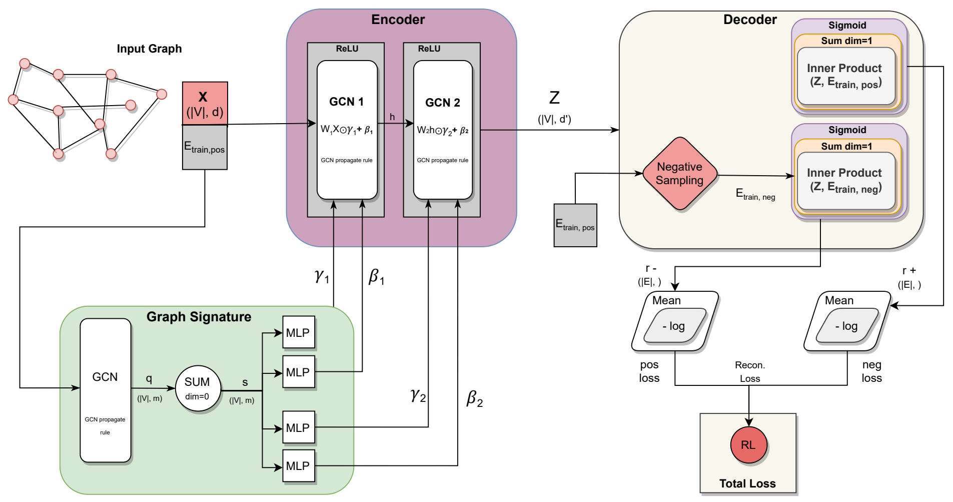

Figure 3: Architecture of Meta-GAE + Signature.

- **Meta-GAE.** In this approach, we simplify the Meta-GAE+Signature by ablate the signature part from the architecture. So, the parameters of the inference model are no longer modulated in the initialization and they are only randomly initialized. Meta-GAE outperform all approaches with 1.4% on average over all the density edge ratios — i.e the ratio of the observed edges in a single graph that used in the training. We reach to conclusion show that the signature does not add any improvements. But, in contrast, it hurt the model performance as we show in result section.

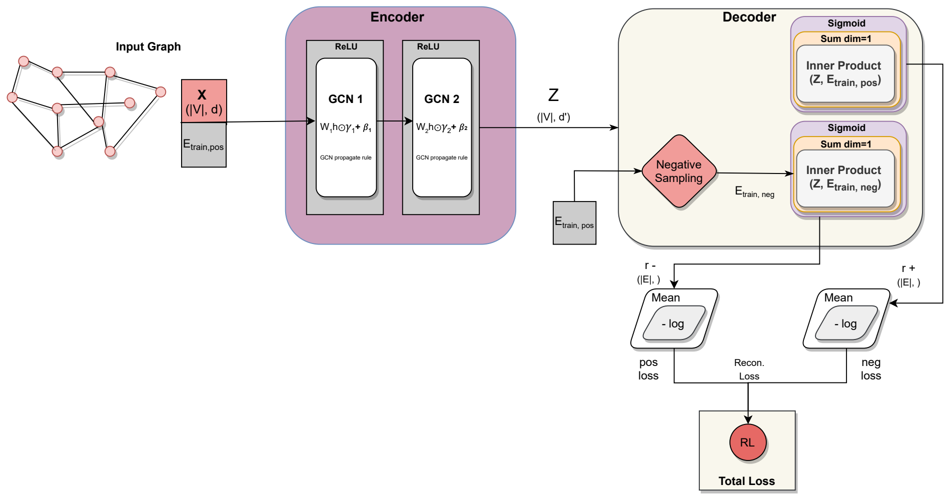

Figure 4: Architecture of Meta-GAE.

- **Dynamic-GAE.** In this method, we moved away from using GNN formalism and specifically using the neural message passing framework to encode the graph information. Instead, we use the KNN-based method called *Dynamic Graph CNN (DGCNN)* that proposed by Y.Wang in [9] as an encoder to represent the node features. Unlike the GNN formalism which use the edges information in order to find the graph representation as in message passing, DGCNN use what called *EdgeConv* operation that only convolve the node features to construct a subgraph from the feature domain after each layer. But in the decoder side, the loss function is still optimized based on the edge connections by take the inner product of the linked node embeddings. The EdgeConv operation is defined by applying a channelwise symmetric aggregation operation $\square$ on the edge features $h_\theta(x_i, x_j)$ associated with all the edges emanating from each node. The output of EdgeConv at the i-th node is given by:

$$x_i^{'} = \square_{j:(i,j)\in E} h_\theta(x_i, x_j) \tag{3}$$

4 Experiments

In this section, we show techniques that we follow to run and train the models that we introduced in Section 3. Here we have two kind of experiments: **1) Meta experiments**, where the model use meta-learning regime (MAML) to optimize the *global* or *slow* parameters for fastly adaption to new graph/task, and we pre-train the model on meta-training dataset with inner training on local subgraphs and then fine-tune on the meta-testing dataset by training on local subgraphs. **2) No-meta experiments**, we take the meta-learning settings out of model, in these kind of experiments our target is to compare which of the above methods that gives a best graph representation for link prediction task. In No-meta experiments, the model is either transfer learning from subgraphs of the training dataset or directly (No transfer learning) train on subgraphs of the testing datasets. We show, in depth, the training processes and main hyperparameters that are used for training. For fair comparison, we run our experiments using the setup that has been used in Meta-Graph. We also use the same edges splitting data loader that developed, and all hyperparameters that have been optimized in Meta-Graph.

4.1 Datasets

Our benchmark is the biological dataset that describe the interaction between the proteins in human tissues [12]. This protein-protein interactions network is represented as multiple graphs where each node indicating a specific protein that contains positional gene sets, motif gene sets and immunological signatures as a node features. And the edges indicating the interaction between the proteins/nodes.

Dataset Information				
Dataset Name	**No. Graphs**	**Avg. Nodes**	**Avg. Edges**	**Node Features**
PPI	24	2,331	64,569	50

Table 1: The table as reported in [3] highlight the statistic of the each dataset. It shows the name of the dataset, number of graphs in each dataset, average number of nodes and edges over graphs, and the dimension of the node features.

4.2 Training Process

As we mentioned, for the fair comparison we follow Meta-Graph setup for training and testing models. Moreover, for more reliability we add an early stopping regularization technique to all meta experiments to save the point when the model performance stop improving in the meta-validation dataset. The training process differ from whether it is meta or no-meta experiments:

- For meta experiments, we take 80% of the graphs for meta-training and the 20% for meta-validation and the last 20% for meta-testing. Using specific density edge ratio we split the edges of any graph into train edges E_{train} to form subgraph $G_{e-train}$, validation edges E_{val} to form subgraph G_{e-val}, and test edges E_{test} to form subgraph G_{e-test} as it mentioned in Section 2. Since each model is MAML-based, the model use the ***slow weights*** (global parameters) that optimized over the full graphs to initialize the ***fast weights*** that optimised thought doing many inner steps on each subgraph $G_{e-train} \subset G_i$. The model is pre-trained using meta-training dataset and then fine-tune on meta-validation and met-testing datasets using the optimized slow weights. Table 3 in next section shows meta-experiments results.

- In no-meta experiments, we also use 80% of the graphs for the training dataset (since no meta-learning we remove the prefix "meta" from dataset and experiments names) and the rest divided equally for validation and test datasets. The same way of splitting edges with the same density edge ratio is applied. However, the model's parameters are initialized from *random*, and the model has only one kind of parameters (fast weights). In the case of normal *transfer learning*, we pre-train the model on training graphs by optimize the parameters many times on training subgraphs $G_{e-train} \subset G_i \in \mathcal{G}_{train}$. Then, fine-tune the model on validation and test datasets via training on subgraphs $G_{e-train} \subset G_i \in \mathcal{G}_{val}, \mathcal{G}_{test}$. While in the case of *no transfer learning*, the model is directly trained on training local subgraphs $G_{e-train}$ of the validation/test graphs $G_i \in \mathcal{G}_{val}, \mathcal{G}_{test}$.

4.3 Hyperparameters

Table 2 highlights most of the hyperparamters that used through all experiments for different density train-edges ratio.

Train-edges ratio \ Hyper.	Epochs	Inner-steps	Inner-lr	Meta-lr	Batch size	Embedding dim.
10%	46	2	2.24e-3	2.727e-3	1	16
20%	32	23	4.949e-2	2.834e-3	1	16
30%	39	15	3.545e-3	1.493e-2	1	16
40%	36	30	8.618e-4	1.1192e-2	1	16
50%	7	15	6.07e-3	1.337e-2	1	16
60%	36	9	4.14e-4	1.42e-3	1	16
70%	18	29	2.592e-2	1.729e-3	1	16

Table 2: Model hyperparameters for different density of train-edges ratio (taken from [3] for fare comparison).

5 Results and Discussion

Results of the meta and no-meta experiments are summarised in Table 3 and 4, respectively. All the results show the accuracy of link prediction models discussed on PPI dataset. Furthermore, we report the mean and the standard deviation for five runs with different seeds. Table 3 show three results of Meta-Graph, first one is taken form the paper [3] where the reproduce result is less than the one reported. This show that Meta-Graph is hard to produce. So, we clean the code and we add an early stopping regularization technique for more reliable result. We compare our methods with the cleaned Meta-Graph as shown in the table below. Also, the results show no differences between using VGAE and GAE. We show that by ablating the graph signature we enhance the accuracy of the model with average 1.4% better than cleaned Meta-Graph over the different density train-edges ratios.

PPI results

Density train-edges ratio	10%	20%	30%	40%	50%	60%	70%
Meta-Graph (paper)[†]	0.795	0.831	0.846	0.853	0.848	0.853	0.855
Meta-VGAE + Sig (Meta-Graph)(reproduced)	0.768 ± 0.002	0.817 ± 0.005	0.766 ± 0.068	0.839 ± 0.013	0.830 ± 0.012	0.852 ± 0.002	0.856 ± 0.003
Meta-VGAE + Sig (Meta-Graph) (reproduced -ES)	0.757 ± 0.004	0.803 ± 0.009	0.819 ± 0.021	0.837 ± 0.009	0.820 ± 0.024	0.844 ± 0.007	0.840 ± 0.003
Meta-GAE + Sig (ES)	0.756 ± 0.001	0.794 ± 0.015	0.805 ± 0.056	0.832 ± 0.015	0.826 ± 0.016	0.851 ± 0.003	0.838 ± 0.007
Meta-GAE (ES)	$\mathbf{0.774 \pm 0.003}$	$\mathbf{0.815 \pm 0.002}$	$\mathbf{0.827 \pm 0.001}$	$\mathbf{0.839 \pm 0.002}$	$\mathbf{0.847 \pm 0.001}$	$\mathbf{0.856 \pm 0.001}$	$\mathbf{0.852 \pm 0.002}$

Table 3: The compared results of meta experiments for PPI dataset. (†) results are included from Meta-Graph paper. (ES) indicate to early stopping regularization.

Moreover, Table 4 report no-meta experiments results of GAE when using GCN as encoder and compare it with using Dynamic Graph as encoder (Dynamic-GAE). In transfer learning, the results show that Dynamic-GAE achieve better result than GAE when the graph is sparse (i.e, observe up to

40% of the edges in the training). While the GAE is perform better when we observe more than 40% (i.e, when we observe 50%, 60%, and 70% of the edges in the training). However, when no transfer learning the GAE perform better than Dynamic-GAE over most of the train-edges ration as shown in the bottom part of the table below.

PPI results

Density train-edges ratio	10%	20%	30%	40%	50%	60%	70%
Transfer Learning							
GAE	0.658 ± 0.004	0.729 ± 0.002	0.769 ± 0.003	0.794 ± 0.003	$\mathbf{0.812 \pm 0.001}$	$\mathbf{0.820 \pm 0.002}$	$\mathbf{0.828 \pm 0.002}$
GAE - lr/10	0.656 ± 0.001	0.728 ± 0.002	0.764 ± 0.001	0.790 ± 0.001	0.806 ± 0.001	0.819 ± 0.001	0.828 ± 0.002
Dynamic-GAE	$\mathbf{0.667 \pm 0.004}$	$\mathbf{0.741 \pm 0.002}$	$\mathbf{0.781 \pm 0.001}$	$\mathbf{0.797 \pm 0.002}$	0.807 ± 0.003	0.801 ± 0.001	0.811 ± 0.004
Dynamic-GAE - lr/10	0.676 ± 0.009	0.742 ± 0.002	0.776 ± 0.002	0.792 ± 0.002	0.801 ± 0.002	0.798 ± 0.003	0.804 ± 0.004
No Transfer Learning							
GAE	$\mathbf{0.698 \pm 0.004}$	$\mathbf{0.754 \pm 0.001}$	0.782 ± 0.004	$\mathbf{0.803 \pm 0.001}$	$\mathbf{0.818 \pm 0.002}$	$\mathbf{0.827 \pm 0.002}$	$\mathbf{0.832 \pm 0.002}$
Dynamic-GAE	0.650 ± 0.026	0.745 ± 0.005	$\mathbf{0.788 \pm 0.001}$	0.799 ± 0.003	0.807 ± 0.003	0.801 ± 0.003	0.811 ± 0.003

Table 4: Compare results of using GCN and Dynamic Graph as encoder in both transfer learning and no transfer learning. Also, experiments when we divide the learning rate by 10 at fine-tuning are shown.

6 Conclusion

We have introduced a comparative and ablative study on the problem of few-shot link prediction. We target kind of applications where we have access to multiple and sparse graphs. Our meta methods (MAML-based) that constructed by ablating and simplify Meta-Graph, show better performance than Meta-Graph itself. Our best method (Meta-GAE) enhance the performance with up to 1.4% on average over different sparsity ratio on PPI dataset. Future work can include using Dynamic-GAE under meta-learning frameworks and apply discussed methods on other datasets with multiple graphs.

Acknowldgement

I would like to thank my supervisor Samira E. Kahou for kindly devoted time and insightful guidance, and special thanks go to Mohammed Havaei for the tremendous effort and helpful advice. My thanks outreach to Bianca Dumitrascu and Mojtaba Faramarzi for kindly help.

References

[1] Luca Maria Aiello, Alain Barrat, Rossano Schifanella, Ciro Cattuto, Benjamin Markines, and Filippo Menczer. Friendship prediction and homophily in social media. *ACM Trans. Web*, 6(2), June 2012.

[2] Y. Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a synaptic learning rule. 01 2002.

[3] Avishek Joey Bose, Ankit Jain, Piero Molino, and William L. Hamilton. Meta-graph: Few shot link prediction via meta learning, 2019.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[5] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

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

[7] TN Kipf and M Welling. Variational graph auto-encoders. arxiv 2016. *arXiv preprint arXiv:1611.07308*.

[8] Jurgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May 1987.

[9] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[10] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[11] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):457–466, 2018.

[12] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.