Graph Few-shot Learning via Knowledge Transfer

Huaxiu Yao[1*], **Chuxu Zhang**[2], **Ying Wei**[3], **Meng Jiang**[2], **Suhang Wang**[1]
Junzhou Huang[3], **Nitesh V. Chawla**[2], **Zhenhui Li**[1]
[1]Pennsylvania State University, [2]University of Notre Dame, [3]Tencent AI Lab
{huaxiuyao, szw494, JessieLi}@psu.edu,{czhang11, mjiang2, nchawla}@nd.edu, {judyweiying, joehhuang}@tencent.com

Abstract

Towards the challenging problem of semi-supervised node classification, there have been extensive studies. As a frontier, Graph Neural Networks (GNNs) have aroused great interest recently, which update the representation of each node by aggregating information of its neighbors. However, most GNNs have shallow layers with a limited receptive field and may not achieve satisfactory performance especially when the number of labeled nodes is quite small. To address this challenge, we innovatively propose a graph few-shot learning (GFL) algorithm that incorporates prior knowledge learned from auxiliary graphs to improve classification accuracy on the target graph. Specifically, a transferable metric space characterized by a node embedding and a graph-specific prototype embedding function is shared between auxiliary graphs and the target, facilitating the transfer of structural knowledge. Extensive experiments and ablation studies on four real-world graph datasets demonstrate the effectiveness of our proposed model and the contribution of each component.

1 Introduction

Classifying a node (e.g., predicting interests of a user) in a graph (e.g., a social network on Facebook) in a semi-supervised manner has been challenging but imperative, inasmuch as only a small fraction of nodes have access to annotations which are usually costly. Drawing inspiration from traditional regularization-based (Zhu, Ghahramani, and Lafferty 2003) and embedding-based (Perozzi, Al-Rfou, and Skiena 2014) approaches for graph-based semi-supervised learning, graph neural networks (GNN) (Kipf and Welling 2017; Veličković et al. 2018) have attracted considerable interest and demonstrated promising performance recently.

To their essential characteristics, GNNs recursively update the feature of each node through aggregation (or message passing) of its neighbors, by which the patterns of graph topology and node features are both captured. Nevertheless, considering that adding more layers increases the difficulty of training and over-smoothens node features (Kipf and Welling 2017), most of existing GNNs have shallow layers with a restricted receptive field. Therefore, GNNs are inadequate to characterize the global information, and work not that satisfactorily when the number of labeled nodes is especially small.

Inspired by recent success of few-shot learning, from a innovative perspective, we are motivated to *leverage the knowledge learned from auxiliary graphs to improve semi-supervised node classification in the target graph of our interest.* The intuition behind lies in that auxiliary graphs and the target graph likely share local topological structures as well as class-dependent node features (Shervashidze et al. 2011; Koutra, Joshua T., and Faloutsos 2013). For example, an existing social network group of co-workers at Google offer valuable clues to predict interests of users in a newly emerged social network group of co-workers at Amazon.

Yet it is even more challenging to achieve few-shot learning on graphs than on independent and identically distributed data (e.g., images) which exisiting few shot learning algorithms focus on. The two lines of recent few-shot learning works, including gradient-based methods (Finn, Abbeel, and Levine 2017; Ravi and Larochelle 2016) and metric-based methods (Snell, Swersky, and Zemel 2017; Vinyals et al. 2016), formulate the transferred knowledge as parameter initializations (or a meta-optimizer) and a metric space, respectively. None of them, however, meets the crucial prerequisite of graph few-shot learning to succeed, i.e., transferring underlying structures across graphs.

To this end, we propose a novel **G**raph **F**ew-shot **L**earning (GFL) model. Built upon metric-based few-shot learning, the basic idea of GFL is to learn a transferable metric space in which the label of a node is predicted as the class of the nearest prototype to the node. The metric space is practically characterized with two embedding functions, which embed a node and the prototype of each class, respectively. Specifically, first, GFL learns the representation of each node using a graph autoencoder whose backbone is GNNs. Second, to better capture global information, we establish a relational structure of all examples belonging to the same class, and learn the prototype of this class by applying a prototype GNN to the relational structure. Most importantly, both embedding functions encrypting structured knowledge are transferred from auxiliary graphs to the target one, to remedy the lack

[*]Correspondence to Huaxiu Yao <huaxiuyao@psu.edu>, Ying Wei <judyweiying@tencent.com>, Zhenhui Li <JessieLi@psu.edu>

of labeled nodes. Besides the two node-level structures, note that we also craft the graph-level representation via a hierarchical graph representation gate, to enforce that similar graphs have similar metric spaces.

To summarize, our main contributions are: (1) to the best of our knowledge, it is the first work that resorts to knowledge transfer to improve semi-supervised node classification in graphs; (2) we propose a novel graph few-shot learning model (GFL) to solve the problem, which simultaneouly transfers node-level and graph-level structures across graphs; (3) comprehensive experiments on four node classification tasks empirically demonstrate the effectiveness of GFL.

2 Related Work

In this section, we briefly introduce the relevant research lines of our work: graph neural network and few-shot learning.

Graph Neural Network Recently, a variety of graph neural network models (GNN) have been proposed to exploit the structures underlying graphs to benefit a variety of applications (Kipf and Welling 2017; Zhang et al. 2019; Tang et al. 2019; Huang et al. 2019; Liu et al. 2019; Gao, Wang, and Ji 2018). There are two lines of GNN methods: non-spectral methods and spectral methods. The spectral methods mainly learn graph representations in the spectral domain (Defferrard, Bresson, and Vandergheynst 2016; Henaff, Bruna, and LeCun 2015; Bruna et al. 2013; Kipf and Welling 2017), where the learned filters are based on Laplacian matrices. For non-spectral methods, the basic idea is to develop an aggregator to aggregate a local set of features (Veličković et al. 2018; Hamilton, Ying, and Leskovec 2017). These methods have achieved great success in several graph-based tasks, such as node classification and graph classification. Thus, in this paper, we are motivated to leverage GNN as the base architecture to learn the node and graph representation.

Few-Shot Learning Few-shot/Zero-shot learning via knowledge transfer has achieve great success in a variety of applications (Li et al. 2019; Li et al. 2018; Yao et al. 2019a). There are two popular types of approaches for few-shot learning: (1) gradient-based few-shot learning methods, which aim to learn a better initialization of model parameters that can be updated by a few gradient steps in future tasks (Finn, Abbeel, and Levine 2017; Finn, Xu, and Levine 2018; Lee and Choi 2018; Yao et al. 2019b) or directly use a meta-optimizer to learn the optimization process (Ravi and Larochelle 2016); (2) metric-based few-shot learning methods, which propose to learn a generalized metric and matching functions from training tasks (Snell, Swersky, and Zemel 2017; Vinyals et al. 2016; Yang et al. 2018; Bertinetto et al. 2019). Our proposed GFL falls into the second category. These traditional metric-based few-shot learning methods are dedicated to independent and identically distributed data, between which no explicit interactions exists.

3 Preliminaries

Graph Neural Network A graph $\mathcal{G}$ is represented as ($\mathbf{A}$, $\mathbf{X}$), where $\mathbf{A} \in \{0,1\}^{n\times n}$ is the adjacent matrix, and $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{n\times h}$ is the node feature matrix. To learn the node represetation for graph $\mathcal{G}$, an embedding function f with parameter θ are defined. In this work, following the "message-passing" architecture (Gilmer et al. 2017), the embedding function f_θ is built upon graph neural network (GNN) in an end-to-end manner, which is formulated as:

$$\mathbf{H}^{(l+1)} = \mathcal{M}(\mathbf{A}, \mathbf{H}^{(l)}; \mathbf{W}^{(l)}), \quad (1)$$

where $\mathcal{M}$ is the message passing function and has a series of possible implementations (Hamilton, Ying, and Leskovec 2017; Kipf and Welling 2017; Veličković et al. 2018), $\mathbf{H}^{(l+1)}$ is the node embedding after l layers of GNN and $\mathbf{W}^{(l)}$ is learnable weight matrix of layer l. The node feature $\mathbf{X}$ is used as the initial node embedding $\mathbf{H}^{(1)}$, i.e., $\mathbf{H}^{(1)} = \mathbf{X}$. After stacking L graph neural network layers, we can get the final representation $\mathbf{Z} = f_\theta(\mathbf{A}, \mathbf{X}) = \mathbf{H}^{(L+1)} \in \mathbb{R}^{h'}$. For simplicity, we will use $\mathbf{Z} = \mathrm{GNN}(\mathbf{A}, \mathbf{X})$ to denote a GNN with L layers.

The Graph Few-Shot Learning Problem Similar as the traditional few-shot learning settings (Snell, Swersky, and Zemel 2017; Vinyals et al. 2016; Finn and Levine 2018), in graph few-shot learning, we are given a sequence of graphs $\{\mathcal{G}_1, \ldots, \mathcal{G}_{N_t}\}$ sampled from a probability distribution $\mathcal{E}$ over tasks (Baxter 1998). For each graph $\mathcal{G}_i \sim \mathcal{E}$. we are provided with a small set of n^{s_i} labeled *support* nodes set $\mathcal{S}_i = \{(\mathbf{x}_{i,j}^{s_i}, y_{i,j}^{s_i})\}_{j=1}^{n^{s_i}}$ and a *query* nodes set $\mathcal{Q}_i = \{(\mathbf{x}_{i,j}^{q_i}, y_{i,j}^{q_i})\}_{j=1}^{n^{q_i}}$, where $y_{i,j} \in \{1, ...K\}$ is the corresponding label. For each node j in query set $\mathcal{Q}_i$, we are supposed to predict its corresponding label by associating its embedding $f_\theta(\mathbf{A}, \mathbf{x}_{i,j}^{q_i}) : \mathbb{R}^h \rightarrow \mathbb{R}^{h'}$ with representation $(f_\theta(\mathbf{A}, \mathbf{x}_{i,j}^{s_i}), y_{i,j}^{s_i})$ in support set $\mathcal{S}_i$ via the similarity measure d. Specifically, in prototypical network (Snell, Swersky, and Zemel 2017), the prototype $\mathbf{c}_i^k$ for each class k is defined as $\mathbf{c}_i^k = \sum_{\mathbf{x}_{i,j}^{s_i} \in \mathcal{S}_i^k} f_\theta(\mathbf{A}, \mathbf{x}_{i,j}^{s_i})/|\mathcal{S}_i^k|$, where $\mathcal{S}_i^k$ denotes the sample set in $\mathcal{S}_i$ of class k and $|\mathcal{S}_i^k|$ means the number of samples in $\mathcal{S}_i^k$. For each graph $\mathcal{G}_i$, the effectiveness on query set $\mathcal{Q}_i$ is evaluated by the loss $\mathcal{L}_i = \sum_k \mathcal{L}_i^k$, where:

$$\mathcal{L}_i^k = -\sum_{(\mathbf{x}_{i,j}^{q_i}, y_{i,j}^{q_i}) \in \mathcal{Q}_i^k} \log \frac{\exp(-d(f_\theta(\mathbf{A}, \mathbf{x}_{i,j}^{q_i}), \mathbf{c}_i^k))}{\sum_{k'} \exp(-d(f_\theta(\mathbf{A}, \mathbf{x}_{i,j}^{q_i}), \mathbf{c}_i^{k'}))}, \quad (2)$$

where $\mathcal{Q}_i^k$ is the query set of class k from $\mathcal{Q}_i$. The goal of graph few-shot learning is to learn a well-generalized embedding function f_θ from previous graphs which can be used to a new graph with a small support set. To achieve this goal, few-shot learning often includes two-steps, i.e., meta-training and meta-testing. In meta-training, the parameter θ of embedding function f_θ is optimized to minimize the expected empirical loss over all historical training graphs, i.e., $\min_\theta \sum_{i=1}^{N_t} \mathcal{L}_i$. Once trained, given a new graph $\mathcal{G}_t$, the learned embedding function f_θ can be used to improve the learning effectiveness with a few support nodes.

4 Methodology

In this section, we elaborate our proposed GFL whose framework is illustrated in Figure 1. The goal of GFL is to adapt graph-structured knowledge learned from existing graphs to the new graph $\mathcal{G}_t$ by exploiting the relational structure in both node-level and graph-level. In the node level, GFL captures the relational structure among different nodes. In part (a) of Figure 1, for each class k, the corresponding relational structure is constructed by the samples in $\mathcal{S}_i^k$ and a prototype GNN (PGNN) is proposed to learn its prototype representation. For graph-level structures, GFL learns the representation of a whole graph and ensures similar graphs have similar structured knowledge to be transferred. Specifically, as illustrated in part (b) (see Figure 2 for more detailed structure of part (b)), the parameters of PGNN highly depend on the whole graph structure which is represented in a hierarchical way. The matching loss is finally computed via the similarity measure d. To enhance the stability of training and the quality of node representation, we further introduce the auxiliary graph reconstruction structure, i.e., the part (c). In the remaining of this section, we will detail the three components, i.e., *graph structured prototype*, *hierarchical graph representation gate* and *auxiliary graph reconstruction*.

4.1 Graph Structured Prototype

In most of the cases, a node plays two important roles in a graph: one is locally interacting with the neighbors that may belong to different classes; the other is interacting with the nodes of the same class in relatively long distance, which can be globally observed. For example, on a biomedical knowledge graph, it is necessary to model both (1) the local structure among disease nodes, treatment nodes, gene nodes, and chemical nodes, and (2) the structure between disease nodes describing their co-occurrence or evolutionary relationship, as well as the structure between gene nodes describing their co-expression. The embedding results $\mathbf{Z}_i$ of graph $\mathcal{G}_i$ describes the first role of local heterogeneous information. And we need to learn the prototype of each class (as defined in Section 3) for the second role of global homogeneous information. It is non-trivial to model the relational structure among support nodes and learn their corresponding prototype, we thus propose a prototype GNN model denoted as PGNN to tackle this challenge.

Given the representation of each node, we first extract the relational structure of samples belong to class k. For each graph $\mathcal{G}_i$, the relational structure $\mathcal{R}_i^k$ of the sample set $\mathcal{S}_i^k$ can be constructed based on some similarity metrics, such as the number of k-hop common neighbors, the inverse topological distance between nodes. To improve the robustness of $\mathcal{R}_i^k$ and alleviate the effect of outlier nodes, we introduce a threshold μ. If the similarity score w between a pair of nodes is smaller than μ, we set it to a fixed value μ_0, i.e., $w = \mu_0$ ($\mu_0 < \mu$). Then, the PGNN is used to model the interactions between samples in the $\mathcal{S}_i^k$, i.e., $\mathrm{PGNN}_\phi(\mathcal{R}_i^k, f_\theta(\mathcal{S}_i^k))$, where PGNN is parameterized by ϕ. Note that, $\mathrm{PGNN}_\phi(\mathcal{R}_i^k, f_\theta(\mathcal{S}_i^k))$ is a representation matrix, and we use j to indicate the j-th node representation. Thus, the graph structured prototype is calculated as follows,

$$\mathbf{c}_i^k = \mathrm{Pool}_{j=1}^{n^{s_i^k}}(\mathrm{PGNN}_\phi(\mathcal{R}_i^k, f_\theta(\mathcal{S}_i^k))[j]), \tag{3}$$

where Pool operator denotes a max or mean pooling operator over support nodes and $n^{s_i^k}$ represents the number of nodes in support set $\mathcal{S}_i^k$.

4.2 Hierarchical Graph Representation Gate

The above prototype construction process is highly determined by the PGNN with the globally shared parameter ϕ. However, different graphs have their own topological structure, motivating us to tailor the globally shared information to each graph. Thus, we learn a hierarchical graph representation for extracting graph-specific information and incorporate it with the parameter of PGNN through a gate function. Before detailing the structure, we first illustrate the importance of the hierarchical graph representation: the simple, node level representation for a graph can be insufficient for many complex graph structures, especially those high-order structures that widely exist and are valuable in real-world applications (Morris et al. 2019).

Figure 2 illustrates the detailed structure of hierarchical graph representation gate. First, following the popular method of hierarchical graph modeling (Ying et al. 2018), the hierarchical graph representation for each level is accomplished by alternating between two level-wise stages: the node assignment and the representation fusion (see part (a) in Figure 2).

Node Assignment In the assignment step, each low-level node is assigned to high-level community. In level r of graph $\mathcal{G}_i$, we denote the number of nodes as K^r, the adjacency matrix as $\mathbf{A}_i^r$, the feature matrix as $\mathbf{X}_i^r$. For the k^r-th node in r-th level, the assignment value $p_i^{k^r \to k^{r+1}}$ from node k^r to node k^{r+1} in $(r+1)$-th level is calculated by applying softmax function on the output of an assignment GNN (AGNN) as follows,

$$p_i^{k^r \to k^{r+1}} = \frac{\exp(\mathrm{AGNN}(\mathbf{A}_i^r, \mathbf{X}_i^r)[k^r, k^{r+1}])}{\sum_{\bar{k}^{r+1}=1}^{K^{r+1}} \exp(\mathrm{AGNN}(\mathbf{A}_i^r, \mathbf{X}_i^r)[k^r, \bar{k}^{r+1}])}, \tag{4}$$

where $\mathrm{AGNN}(\mathbf{A}_i^r, \mathbf{X}_i^r)[k^r, k^{r+1}] \in \mathbb{R}^1$ denotes the assignment representation value from node k^r in level r to node k^{r+1} in level $r+1$. The whole assignment matrix including every assignment probability $p_i^{k^r \to k^{r+1}}$ is denoted as $\mathbf{P}_i^{r \to r+1} \in \mathbb{R}^{K^r \times K^{r+1}}$.

Representation Fusion After getting the assignment matrix $\mathbf{P}_i^{r \to r+1} \in \mathbb{R}^{K^r \times K^{r+1}}$, for level $r+1$, the adjacent matrix is defined as $\mathbf{A}_i^{r+1} = (\mathbf{P}_i^{r \to r+1})^T \mathbf{A}_i^r \mathbf{P}_i^{r \to r+1}$ and the feature matrix is calculated by applying assignment matrix on the output of a fusion GNN (FGNN), i.e., $\mathbf{X}_i^{r+1} = (\mathbf{P}_i^{r \to r+1})^T \mathrm{FGNN}(\mathbf{A}_i^r, \mathbf{X}_i^r)$. Then, the feature representation $\mathbf{h}_i^{r+1}$ of level $r+1$ can be calculated by aggregating the representation of all nodes, i.e.,

$$\mathbf{h}_i^{r+1} = \mathrm{Pool}_{k^{r+1}=1}^{K^{r+1}}((\mathbf{P}_i^{r \to r+1})^T \mathrm{FGNN}(\mathbf{A}_i^r, \mathbf{X}_i^r)[k^{r+1}]), \tag{5}$$

where $\mathbf{X}_i^{r+1}[k^{r+1}] = (\mathbf{P}_i^{r \to r+1})^T \mathrm{FGNN}(\mathbf{A}_i^r, \mathbf{X}_i^r)[k^{r+1}]$ denotes the feature representation of node k^{r+1}.

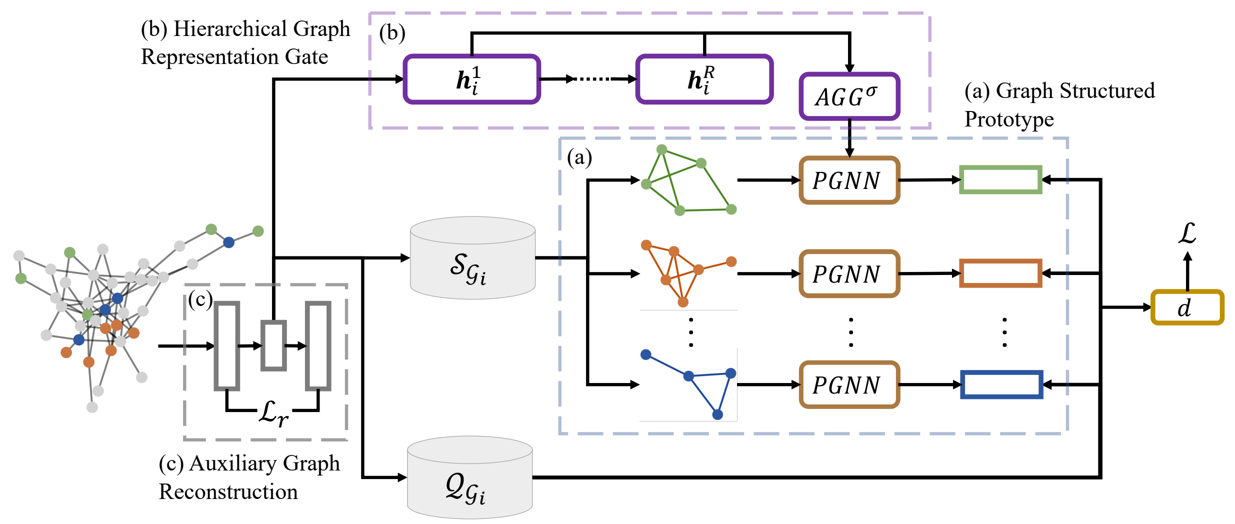

(b) Hierarchical Graph Representation Gate
(b)
h_i^1
h_i^R
AGG^σ
(a) Graph Structured Prototype
(a)
$PGNN$
$PGNN$
$PGNN$
$\mathcal{L}$
d
$\mathcal{S}_{\mathcal{G}_i}$
(c)
$\mathcal{L}_r$
(c) Auxiliary Graph Reconstruction
$\mathcal{Q}_{\mathcal{G}_i}$

Figure 1: The framework of proposed GFL including three components. (a) Graph structured prototype: we extract the graph relation structure for the support set $\mathcal{S}_i^k$ of each class k and use a prototype GNN (PGNN) to learn its representation $\mathbf{c}_i^k$; (b) Hierarchical graph representation gate: we learn the hierarchical representations of graph from level 1 to R, i.e., $\mathbf{h}_i^1$ to $\mathbf{h}_i^R$, and use the aggregated representation to modulate the parameters of PGNN; (c) Auxiliary graph reconstruction: we construct the graph autoencoder to improve the training stability and the quality of node representation.

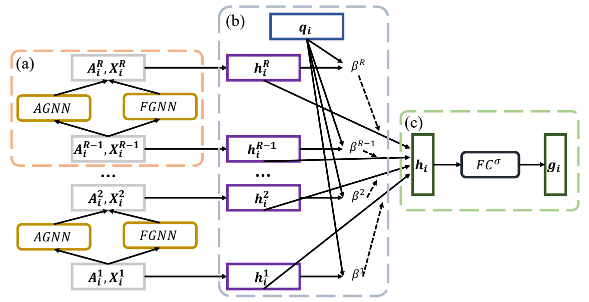


Figure 2: The detailed framework of hierarchical graph representation gate: (a) the basic block for learning hierarchical representation $\{\mathbf{h}_i^1, \dots, \mathbf{h}_i^R\}$; (b) aggregator to aggregate $\{\mathbf{h}_i^1, \dots, \mathbf{h}_i^R\}$, where the learnable query vector $\mathbf{q}_i$ is introduced to calculate the attention weight $\beta^1 \cdots \beta^R$. Note that, we only illustrate the attention aggregator. (c) graph-specific gate construction, where $\mathbf{h}_i$ is used to calculate gate $\mathbf{g}_i$.

By calculating the representation of each level, we get the representation set $\{\mathbf{h}_i^1, \dots, \mathbf{h}_i^R\}$ which encrypts the graph structure from different levels. Then, to get the whole graph representation $\mathbf{h}_i$, the representation of each level is aggregated via an aggregator AGG. In this work, we propose two candidate aggregators: mean pooling aggregator and attention aggregator (see part (b) in Figure 2). For mean pooling aggregator, the graph representation $\mathbf{h}_i$ is defined as:

$$\mathbf{h}_i = \mathrm{AGG}_{\mathrm{mean}}(\{\mathbf{h}_i^1, \dots, \mathbf{h}_i^R\}) = \frac{1}{R}\sum_{r=1}^{R} \mathbf{h}_i^r. \tag{6}$$

Considering the representation of different levels may have different contributions on the whole representation $\mathbf{h}_i$, for attention aggregator, we first introduce a learnable query vector as $\mathbf{q}_i$, and then the formulation is

$$\mathbf{h}_i = \mathrm{AGG}_{\mathrm{att}}(\{\mathbf{h}_i^1, \dots, \mathbf{h}_i^R\}) = \sum_{r=1}^{R} \beta_i^r \mathbf{h}_i^r = \sum_{r=1}^{R} \frac{\mathbf{q}_i^T \mathbf{h}_i^r}{\sum_{r'=1}^{R} \mathbf{q}_i^T \mathbf{h}_i^{r'}} \mathbf{h}_i^r. \tag{7}$$

After the aggregating process, the final representation $\mathbf{h}_i$ is expected to be graph-specific. Inspired by previous findings (Xu et al. 2015): similar graphs may activate similar parameters (i.e., parameter ϕ of the PGNN), we introduce a gate function $\mathbf{g}_i = \mathcal{T}(\mathbf{h}_i)$ (see part (c) in Figure 2) to tailor graph structure specific information. Then, the global transferable knowledge (i.e., ϕ) is adapted to the structure-specific parameter via the gate function, which is defined as follows:

$$\phi_i = \mathbf{g}_i \circ \phi = \mathcal{T}(\mathbf{h}_i) \circ \phi, \tag{8}$$

where $\circ$ represents element-wise multiplication. $\mathbf{g}_i = \mathcal{T}(\mathbf{h}_i)$ maps the graph-specific representation $\mathbf{h}_i$ to the same space of parameter ϕ, which is defined as:

$$\mathbf{g}_i = \mathcal{T}(\mathbf{h}_i) = \sigma(\mathbf{W}_g \mathbf{h}_i + \mathbf{b}_g), \tag{9}$$

where $\mathbf{W}_g$ and $\mathbf{b}_g$ are learnable parameters. Thus, PGNN_ϕ in Eqn. (3) would be PGNN_{ϕ_i}.

4.3 Auxiliary Graph Reconstruction

In practice, it is difficult to learn an informative node representation using only the signal from the matching loss, which motivates us to design a new constraint for improving the training stability and the quality of node representation. Thus, for the node embedding function $f_\theta(\cdot)$, we refine it by using a graph autoencoder. The reconstruction loss for training autoencoder is defined as follows,

$$\mathcal{L}_r(\mathbf{A}_i, \mathbf{X}_i) = \|\mathbf{A}_i - \mathrm{GNN}_{dec}(\mathbf{Z}_i)\mathrm{GNN}_{dec}^T(\mathbf{Z}_i)\|_F^2, \tag{10}$$

Algorithm 1 Training Process of GFL

```
Require: $\mathcal{E}$: distribution over graphs; $L$: # of layers in hierarchical structure; $\alpha$: stepsize; $\gamma$: balancing parameter for loss
1: Randomly initialize $\Theta$
2: while not done do
3:     Sample a batch of graphs $\mathcal{G}_i \sim \mathcal{E}$ and its corresponding adjacent matrices $\mathbf{A}_i$ and feature matrices $\mathbf{X}_i$
4:     for all $\mathcal{G}_i$ do
5:         Sample support set $\mathcal{S}_i$ and query set $\mathcal{Q}_i$
6:         Compute the embedding $f_\theta(\mathbf{A}_i, \mathbf{X}_i)$ and its reconstruction error $\mathcal{L}_r(\mathbf{A}_i, \mathbf{X}_i)$ in Eqn. (10)
7:         Compute the hierarchical representation $\{\mathbf{h}_i^1, \dots, \mathbf{h}_i^R\}$ in Eqn. (5) and gated parameter $\phi_i$ in Eqn. (8)
8:         Construct relational graphs $\{\mathcal{R}_i^1, ..., \mathcal{R}_i^K\}$ for samples in $\mathcal{S}_i$ and compute graph prototype $\{\mathbf{c}_i^1, ..., \mathbf{c}_i^K\}$ in Eqn. (3).
9:         Compute the matching score using the query set $\mathcal{Q}_i$ and evaluate loss in Eqn. (2)
10:    end for
11:    Update $\Theta \leftarrow \Theta - \alpha \nabla_\Theta \sum_{i=1}^{N_t} \mathcal{L}_i(\mathbf{A}_i, \mathbf{X}_i) + \gamma \mathcal{L}_r(\mathbf{A}_i, \mathbf{X}_i)$
12: end while
```

where $\mathbf{Z}_i = \text{GNN}_{enc}(\mathbf{A}_i, \mathbf{H}_i)$ is the representation for each node in graph $\mathcal{G}_i$ and $\|\cdot\|_F$ represents the Frobenius norm. Recalling the objectives for a prototypical network in Section 3, we reach the optimization problem of GFL as $\min_\Theta \sum_{i=1}^{N_t} \mathcal{L}_i + \gamma \mathcal{L}_r(\mathbf{A}_i, \mathbf{X}_i)$, where Θ represents all learnable parameters. The whole training process of GFL is detailed in Alg. 1.

5 Experiments

In this section, we conduct extensive experiments to demonstrate the benefits of GFL, with the goal of answering the following questions: (1) Can GFL outperform baseline methods? (2) Can our proposed graph structured prototype and hierarchical graph representation gate improve the performance? (3) Can our approach learn better representations for each class?

Table 1: Data Statistics.

Dataset	Colla.	Reddit	Cita.	Pubmed
# Nodes (avg.)	4,496	5,469	2,528	2,901
# Edges (avg.)	14,562	7,325	14,710	5,199
# Features/Node	128	600	100	500
# Classes	4	5	3	3
# Graphs (Meta-train)	100	150	30	60
# Graphs (Meta-val.)	10	15	3	5
# Graphs (Meta-test)	20	25	10	15

Dataset Description We use four datasets of different kinds of graphs: Collaboration, Reddit, Citation and Pubmed. (1): *Collaboration data*: Our first task is to predict research domains of different academic authors. We use the collaboration graphs extracted from the AMiner data (AMi 2019). Each author is assigned with a computer science category label according to the majority of their papers' categories. (2): *Reddit data*: In the second task, we predict communities of different Reddit posts. We construct post-to-post graphs from Reddit community data (Hamilton, Ying, and Leskovec 2017), where each edge denotes that the same user comments on both posts. Each post is labeled with a community id. (3): *Citation data*: The third task is to predict paper categories. We derive paper citation graphs from the AMiner data and each paper is labeled with a computer science category label. (4): *Pubmed data*: Similar to the third task, the last task is to predict paper class labels. The difference is that the citation graphs are extracted from the PubMed database (Veličković et al. 2018) and each node is associated with diabetes class id. The statistics of these datasets are reported in Table 1.

Experimental Settings In this work, we follow the traditional few-shot learning settings (Finn, Abbeel, and Levine 2017; Snell, Swersky, and Zemel 2017). For each graph, N labeled nodes for each class are provided as support set. The rest nodes are used as query set for evaluating the performance. Like (Kipf and Welling 2017), the embedding structure (i.e., θ in Eqn. (3)) is a two-layer graph convolutional structure (GCN) with 32 neurons in each layer. For PGNN in Eqn. (3), each AGNN in Eqn. (4) and each FGNN in Eqn. (5), we use one-layer GCN as the proxy of GNN. The distance metric d is defined as the inner product distance. For our proposed GFL, we use GFL-mean and GFL-att to represent the type of hierarchical representation aggregator (i.e., GFL-mean represents mean pooling aggregator in Eqn. (6) and GFL-att represents attention aggregator in Eqn. (7)). The threshold μ_0 in Section 4.1 for constructing relation structure of support set is set as 0.5. Detailed hyperparameter settings can be found in Appendix A.

Baseline Methods For performance comparison of node classification, we consider three types of baselines: (1) *Graph-based semi-supervised methods* including Label Propagation (LP) (Zhu and Ghahramani 2002) and Planetoid (Yang, Cohen, and Salakhudinov 2016); (2) *Graph representation learning methods* including Deepwalk (Perozzi, Al-Rfou, and Skiena 2014), node2vec (Grover and Leskovec 2016), Non-transfer-GCN (Kipf and Welling 2017). Note that, for Non-transfer-GCN, we train GCN on each meta-testing graph with limited labeled data rather than transferring knowledge from meta-training graphs; (3) *Transfer/few-shot methods* including All-Graph-Finetune (AGF), K-nearest-neighbor (K-NN), Matching Network (Matchingnet) (Vinyals et al. 2016), MAML (Finn, Abbeel, and Levine 2017), Prototypical Network (Protonet) (Snell, Swersky, and Zemel 2017). Note that, for All-Graph-Finetune and K-NN methods, follow the settings of (Triantafillou et al. 2019), we first learn the parameters of the embedding structure by feeding all meta-graphs one by one. Then, we finetune the parameters or use K-NN to classify nodes based on the learned parameters of embedding structure. Each transfer/few-shot learning method uses the same embedding structure (i.e., two layers GCN) as GFL. More detailed descriptions and implementation of

Table 2: Comparison between GFL and other node classification methods on four graph datasets. Performance of Accuracy±95% confidence intervals on 10-shot classification are reported.

Model	Collaboration	Reddit	Citation	Pubmed
LP (Zhu and Ghahramani 2002)	$61.09 \pm 1.36\%$	$23.40 \pm 1.63\%$	$67.00 \pm 4.50\%$	$48.55 \pm 6.01\%$
Planetoid (Yang, Cohen, and Salakhudinov 2016)	$62.95 \pm 1.23\%$	$50.97 \pm 3.81\%$	$61.94 \pm 2.14\%$	$51.43 \pm 3.98\%$
Deepwalk (Perozzi, Al-Rfou, and Skiena 2014)	$51.74 \pm 1.59\%$	$34.81 \pm 2.81\%$	$56.56 \pm 5.25\%$	$44.33 \pm 4.88\%$
node2vec (Grover and Leskovec 2016)	$59.77 \pm 1.67\%$	$43.57 \pm 2.23\%$	$54.66 \pm 5.16\%$	$41.89 \pm 4.83\%$
Non-transfer-GCN (Kipf and Welling 2017)	$63.16 \pm 1.47\%$	$46.21 \pm 1.43\%$	$63.95 \pm 5.93\%$	$54.87 \pm 3.60\%$
All-Graph-Finetune (AGF)	$76.09 \pm 0.56\%$	$54.13 \pm 0.57\%$	$88.93 \pm 0.72\%$	$83.06 \pm 0.72\%$
K-NN	$67.53 \pm 1.33\%$	$56.06 \pm 1.36\%$	$78.18 \pm 1.70\%$	$74.33 \pm 0.52\%$
Matchingnet (Vinyals et al. 2016)	$80.87 \pm 0.76\%$	$56.21 \pm 1.87\%$	$94.38 \pm 0.45\%$	$85.65 \pm 0.21\%$
MAML (Finn, Abbeel, and Levine 2017)	$79.37 \pm 0.41\%$	$59.39 \pm 0.28\%$	$95.71 \pm 0.23\%$	$88.44 \pm 0.46\%$
Protonet (Snell, Swersky, and Zemel 2017)	$80.49 \pm 0.55\%$	$60.46 \pm 0.67\%$	$95.12 \pm 0.17\%$	$87.90 \pm 0.54\%$
GFL-mean (Ours)	$83.51 \pm 0.38\%$	$62.66 \pm 0.57\%$	$\mathbf{96.51 \pm 0.31}\%$	$\mathbf{89.37 \pm 0.41}\%$
GFL-att (Ours)	$\mathbf{83.79 \pm 0.39}\%$	$\mathbf{63.14 \pm 0.51}\%$	$95.85 \pm 0.26\%$	$88.96 \pm 0.43\%$

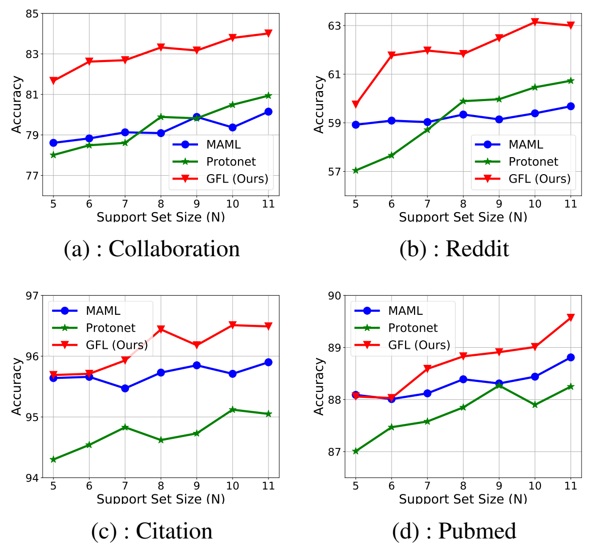
baselines can be found in Appendix B.

5.1 Results

Overall Results For each dataset, we report the averaged accuracy with 95% confidence interval over meta-testing graphs of 10-shot node classification in Table 2. Comparing with graph-based semi-supervised methods and graph representation learning methods, first, we can see that transfer/few-shot methods (i.e., AGF, K-NN, Matchingnet, MAML, Protonet, GFL) significantly improve the performance, showing the power of knowledge transfer from previous learned graphs. Second, both GFL-mean and GFL-att achieve the best performance than other transfer/few-shot methods on four datasets, indicating the effectiveness by incorporating graph prototype and hierarchical graph representation. In addition, as a metric distance based meta-learning algorithm, GFL not only outperforms other algorithms from this research line (i.e., Matchingnet, Protonet), but also achieves better performance than MAML, a representative gradient-based meta-learning algorithm.

Ablation Studies Since GFL integrates three essential components (i.e., graph structured prototype, hierarchical graph representation gate, auxiliary graph reconstruction), we conduct extensive ablation studies to understand the contribution of each component. Table 3 shows the results of ablation studies on each dataset, where the best results among GFL-att and GFL-mean are reported as GFL results. Performance of accuracy are reported in this table. For the graph structured prototype, in (M1a), we first report the performance of protonet for comparison since Protonet use mean pooling of node embedding instead of constructing and exploiting relational structure for each class.

To show the effectiveness of hierarchical graph representation gate, we first remove this component and report the performance in (M2a). The results are inferior, demonstrating that the effectiveness of graph-level representation. In addition, we only use the flat representation structure (i.e., $R = 1$) in (M2b). The results show the effectiveness of hierarchical representation.



(a) : Collaboration (b) : Reddit

(c) : Citation (d) : Pubmed

Figure 3: Effect of support set size, which is represented by shot number N

For auxiliary graph reconstruction, we remove the decoder GNN and only use the encoder GNN to learn the node representation in (M3). GFL outperforms (M3) as the graph reconstruction loss refines the learned node representation and enhance the stability of training.

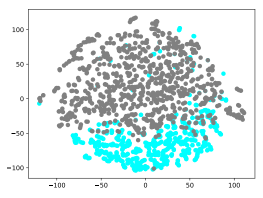
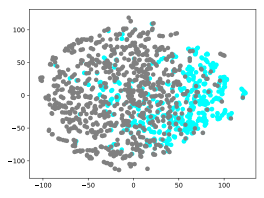
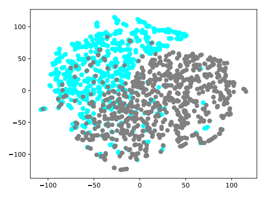
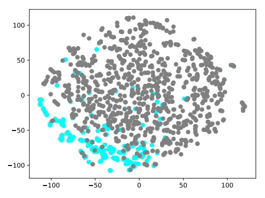
5.2 Sensitivity Analysis

In this section, we analyze the sensitivities of the model to the size of the support set, threshold μ in the construction of the graph prototype, the similarity function for constructing relational structure $\mathcal{R}_i^k$, and different distance functions d.

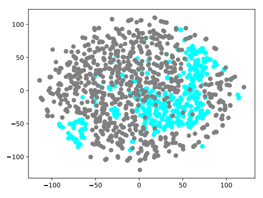
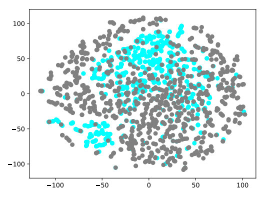
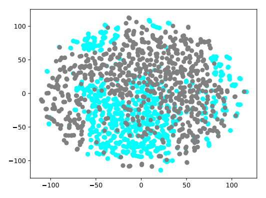
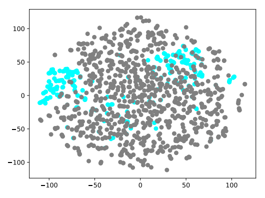
Effect of Support Set Size We analyze the effect of the support set size, which is represented by the shot number N. For comparisons, we select two representative few-shot learning methods: Protonet (metric-learning based model) and

Table 3: Results of Ablation Study. Performance of Accuracy$\pm 95\%$ confidence intervals are reported. We select the best performance of GFL-mean and GFL-att as GFL in this table.

Ablation Model	Collaboration	Reddit	Citation	Pubmed
(M1a): use the mean pooling prototype (i.e., protonet)	$80.49 \pm 0.55\%$	$60.46 \pm 0.67\%$	$95.12 \pm 0.17\%$	$87.90 \pm 0.54\%$
(M2a): remove the hierarchical representation gate	$82.63 \pm 0.45\%$	$61.99 \pm 0.27\%$	$95.33 \pm 0.35\%$	$88.15 \pm 0.55\%$
(M2b): use flat representation rather than hierarchical	$83.45 \pm 0.41\%$	$62.55 \pm 0.65\%$	$95.76 \pm 0.37\%$	$89.08 \pm 0.47\%$
(M3): remove the graph reconstruction loss	$82.98 \pm 0.37\%$	$62.58 \pm 0.47\%$	$95.63 \pm 0.27\%$	$89.11 \pm 0.43\%$
GFL (Ours)	$\mathbf{83.79 \pm 0.39\%}$	$\mathbf{63.14 \pm 0.51\%}$	$\mathbf{96.51 \pm 0.31\%}$	$\mathbf{89.37 \pm 0.41\%}$

(a) : Class 1: GFL (Ours) (b) : Class 2: GFL (Ours) (c) : Class 3: GFL (Ours) (d) : Class 4: GFL (Ours)

(e) : Class 1: Protonet (f) : Class 2: Protonet (g) : Class 3: Protonet (h) : Class 4: Protonet

Figure 4: Embedding visualization of positive (cyan nodes) and negative (grey nodes) data samples for each class.

MAML (gradient-based model). The results of each dataset are shown in Figure 3a-3d. We notice that when the support set size is small, Protonet performs worse than MAML. One potential reason could be that Protonet is sensitive to outliers, as it calculates prototype by averaging values over samples with equal weights. Thus, more data is expected to derive a reliable prototype. However, by extracting the relational structure among samples of the same class, our algorithm is more robust and achieves best performance in all scenarios.

Table 4: Effect of threshold μ in relational structure construction of graph prototype. Results of Accuracy are reported.

μ	Collaboration	Reddit	Citation	Pubmed
0.5	$\mathbf{83.79}\%$	63.14%	$\mathbf{96.51}\%$	89.37%
0.6	83.26%	$\mathbf{63.43}\%$	96.19%	89.42%
0.7	83.31%	63.17%	95.35%	$\mathbf{89.60}\%$
0.8	83.14%	63.21%	95.18%	89.21%

Effect of Threshold μ in Graph Prototype Construction We further analyze the effect of threshold μ for constructing relational structure of the graph prototype. The results of $\mu = \{0.5, 0.6, 0.7, 0.8\}$ are reported in Table 4. In this table, the best threshold varies among different datasets. The effectiveness of the threshold demonstrates that the proposed model is robust to outliers.

Table 5: Effect of different similarity functions for calculating relational structure $\mathcal{R}_i^k$. Results of Accuracy are reported.

Method	Collab.	Reddit	Cita.	Pubmed
Jaccard	82.98%	62.71%	95.18%	88.91%
Adamic-Adar	83.70%	62.87%	95.49%	89.21%
PageRank	$\mathbf{84.14}\%$	63.08%	95.93%	$\mathbf{90.02}\%$
Top-k CN	83.79%	$\mathbf{63.14}\%$	$\mathbf{96.51}\%$	89.37%

Effect of Different Similarity Functions for Constructing Relational Structure $\mathcal{R}_i^k$ We further analyze the effect of different similarity functions for constructing relational structure of the graph prototype. Jaccard Index, Adamic-Adar (Adamic and Adar 2003), PageRank and Top-k Common Neighbors (Top-k CN) are selected and the results are reported in Table 5. Note that, in previous results, we all use Top-k CN as similarity function. The results show that GFL is not very sensitive to the similarity on a dataset may be achieved by different functions.

Effect of Distance Functions d In addition, we replace the distance function d in Eqn. (2) from inner product (used in the original setting) for cosine distance. The results are reported in Table 6. Compared with inner product, the similar results show that GFL is not very sensitivity to the distance function d.

Table 6: Effect of different distance functions d.

Method	Collab.	Reddit	Cita.	Pubmed
GFL (inner product)	83.79%	63.14%	96.51%	89.37%
GFL (cosine)	84.02%	62.95%	96.02%	89.25%

5.3 Analysis of Learned Representation

To better compare the learned representation of our model and Protonet, for each class, we use t-SNE (Maaten and Hinton 2008) to visualize the embedding (i.e., f_θ) of positive samples (belong to this class) and 1000 negative samples (not belong to this class). The results of GFL and Protonet on collabration data are shown in Figure 4. Compared with Protonet, in this figure, we can see that our model can better distinguish the positive and negative samples.

6 Conclusion

In this paper, we introduce a new framework GFL to improve the effectiveness of semi-supervised node classification on a new target graph by transferring knowledge learned from auxiliary graphs. Built upon the metric-based few-shot learning, GFL integrates local node-level and global graph-level knowledge to learn a transferable metric space charaterized by node and prototype embedding functions. The empirical results demonstrate the effectiveness of our proposed model on four node classification datasets.

Acknowledgement

The work was supported in part by NSF awards #1652525, #1618448, #1849816, #1925607, and #1629914, the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

[Adamic and Adar 2003] Adamic, L. A., and Adar, E. 2003. Friends and neighbors on the web. *Social networks* 25(3):211–230.

[AMi 2019] 2019. Aminer. https://aminer.org/.

[Baxter 1998] Baxter, J. 1998. Theoretical models of learning to learn. In *Learning to learn*. Springer. 71–94.

[Bertinetto et al. 2019] Bertinetto, L.; Henriques, J. F.; Torr, P. H.; and Vedaldi, A. 2019. Meta-learning with differentiable closed-form solvers. In *ICLR*.

[Bruna et al. 2013] Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

[Defferrard, Bresson, and Vandergheynst 2016] Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 3844–3852.

[Finn, Abbeel, and Levine 2017] Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 1126–1135.

[Finn and Levine 2018] Finn, C., and Levine, S. 2018. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *ICLR*.

[Finn, Xu, and Levine 2018] Finn, C.; Xu, K.; and Levine, S. 2018. Probabilistic model-agnostic meta-learning. In *NIPS*.

[Gao, Wang, and Ji 2018] Gao, H.; Wang, Z.; and Ji, S. 2018. Large-scale learnable graph convolutional networks. In *KDD*. ACM.

[Gilmer et al. 2017] Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *ICML*, 1263–1272. JMLR. org.

[Grover and Leskovec 2016] Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *KDD*, 855–864. ACM.

[Hamilton, Ying, and Leskovec 2017] Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*, 1024–1034.

[Henaff, Bruna, and LeCun 2015] Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

[Huang et al. 2019] Huang, C.; Wu, X.; Zhang, X.; Zhang, C.; Zhao, J.; Yin, D.; and Chawla, N. V. 2019. Online purchase prediction via multi-scale modeling of behavior dynamics. In *KDD*. ACM.

[Kipf and Welling 2017] Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[Koutra, Joshua T., and Faloutsos 2013] Koutra, D.; Joshua T., V.; and Faloutsos, C. 2013. Deltacon: A principled massive-graph similarity function. In *SDM*.

[Lee and Choi 2018] Lee, Y., and Choi, S. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, 2933–2942.

[Li et al. 2018] Li, Z.; Wei, Y.; Zhang, Y.; and Yang, Q. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *AAAI*.

[Li et al. 2019] Li, Z.; Li, X.; Wei, Y.; Bing, L.; Zhang, Y.; and Yang, Q. 2019. Transferable end-to-end aspect-based sentiment analysis with selective adversarial learning. In *EMNLP-IJCNLP*.

[Liu et al. 2019] Liu, Z.; Chen, C.; Li, L.; Zhou, J.; Li, X.; Song, L.; and Qi, Y. 2019. Geniepath: Graph neural networks with adaptive receptive paths. In *AAAI*.

[Maaten and Hinton 2008] Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9(Nov):2579–2605.

[Morris et al. 2019] Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*.

[Perozzi, Al-Rfou, and Skiena 2014] Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710.

[Ravi and Larochelle 2016] Ravi, S., and Larochelle, H. 2016. Optimization as a model for few-shot learning. *ICLR*.

[Shervashidze et al. 2011] Shervashidze, N.; Schweitzer, P.; Leeuwen, E. J. v.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *JMLR* 12(Sep):2539–2561.

[Snell, Swersky, and Zemel 2017] Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *NIPS*, 4077–4087.

[Tang et al. 2019] Tang, X.; Li, Y.; Sun, Y.; Yao, H.; Mitra, P.; and Wang, S. 2019. Robust graph neural network against poisoning attacks via transfer learning. *arXiv preprint arXiv:1908.07558*.

[Triantafillou et al. 2019] Triantafillou, E.; Zhu, T.; Dumoulin, V.; Lamblin, P.; Xu, K.; Goroshin, R.; Gelada, C.; Swersky, K.; Manzagol, P.-A.; and Larochelle, H. 2019. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*.

[Veličković et al. 2018] Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *ICLR*.

[Vinyals et al. 2016] Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *NIPS*, 3630–3638.

[Xu et al. 2015] Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2048–2057.

[Yang et al. 2018] Yang, F. S. Y.; Zhang, L.; Xiang, T.; Torr, P. H.; and Hospedales, T. M. 2018. Learning to compare: Relation network for few-shot learning. In *CVPR*.

[Yang, Cohen, and Salakhudinov 2016] Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 40–48.

[Yao et al. 2019a] Yao, H.; Liu, Y.; Wei, Y.; Tang, X.; and Li, Z. 2019a. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *WWW*. ACM.

[Yao et al. 2019b] Yao, H.; Wei, Y.; Huang, J.; and Li, Z. 2019b. Hierarchically structured meta-learning. In *International Conference on Machine Learning*, 7045–7054.

[Ying et al. 2018] Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. In *NIPS*, 4805–4815.

[Zhang et al. 2019] Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous graph neural network. In *KDD*.

[Zhu and Ghahramani 2002] Zhu, X., and Ghahramani, Z. 2002. Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer.

[Zhu, Ghahramani, and Lafferty 2003] Zhu, X.; Ghahramani, Z.; and Lafferty, J. D. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*.

A Additional Hyperparameter Settings

For more detailed hyperparameter settings, the learning rate is set as 0.01, the dimension of hierarchical graph representation $\mathbf{h}_i$ is set as 32. The reconstruction loss weight γ is set as 1.0. Additionally, in order to construct the relational graph of few-shot labeled nodes for each class, we compute the similarity score between each two nodes by counting the number of k-hop (k=3) common neighbors and further smooth this similarity value by a sigmoid function. The computed similarity matrix is further fed into GNN for generating prototype embedding.

B Detailed Descriptions of Baselines

We detail three types of baselines in this section. Note that, all GCN used in baselines are two-layers GCN with 32 neurons each layer. The descriptions are as follows:

- **Graph-based semi-supervised methods**:
  - **Label Propagation (LP)** (Zhu and Ghahramani 2002): Label Propagation is a traditional semi-supervised learning methods.
  - **Planetoid** (Yang, Cohen, and Salakhudinov 2016) Planetoid is a semi-supervised learning method based on graph embeddings. We use transductive formulation of Planetoid in this paper.
- **Graph representation learning methods**:
  - **Deepwalk** (Perozzi, Al-Rfou, and Skiena 2014): Deepwalk learns the node embedding in an unsupervised way. We concatenate the learned node embedding and the node features, then feed them to the multiclass classifier (using Scikit-Learn[1]). Few-shot node labels in each graph are available for training classifer.
  - **node2vec** (Grover and Leskovec 2016): This method is similar to the Deepwalk while we use node2vec model to learn node embedding.
  - **Non-transfer-GCN** (Kipf and Welling 2017) In Non-transfer-GCN, we only train GCN on each meta-testing network without transferring knowledge from meta-training networks.
- **Transfer/Few-shot methods**
  - **All-Graph-Finetune (AGF)** In AGF, we train GCN by feeding each meta-training graph one-by-one. Then, we can learn the initialization of GCN parameters from meta-training graphs. In meta-testing process, we finetune the learned initialization on every meta-testing graphs. The hyperparameters (e.g., learning rate) of AGF are the same as GFL.
  - **K-nearest-neighbor** (K-NN) Similar as the settings of (Triantafillou et al. 2019), we first learn the initialization of GCN parameters by using all meta-training graphs. Then, in the testing process, we use the learned embedding function (i.e., GCN) to learn the representation of support nodes and each query node. Finally, we use k-NN for classification.
  - **Matching Network (Matchingnet)** (Vinyals et al. 2016): Matching network is a metric-based few-shot learning method. Since traditional matching network focuses on one-shot learning, in our scenario, we still use each query node representation to match the most similar on in support set and use its label as the predicted label for the query node.
  - **MAML** (Finn, Abbeel, and Levine 2017): MAML is a representative gradient-based meta-learning method, which learn a well-generalized model initialization which can be adapted with a few gradient steps. For MAML, we set the learning rate of inner loop as 0.001.

[1]https://scikit-learn.org/stable/

- **Prototypical Network (Protonet)** (Snell, Swersky, and Zemel 2017) Prototypical network is a representative metric-based few-shot learning method, which constructs the prototype by aggregating nodes belong to the same class using mean pooling.