

Few-shot Node Classification on Attributed Networks with Graph Meta-learning

Yonghao Liu, Mengyu Li
College of Computer Science and Technology, Jilin University
Changchun, China
{yonghao20,mengyul21}@mails.jlu.edu.cn

Ximing Li
College of Computer Science and Technology, Jilin University
Changchun, China
ximingli@jlu.edu.cn

Fausto Giunchiglia
DISI, University of Trento
Trento, Italy
fausto.giunchiglia@unitn.it

Xiaoyue Feng*, Renchu Guan*
College of Computer Science and Technology, Jilin University
Changchun, China
{fengxy,guanrenchu}@jlu.edu.cn

ABSTRACT

Attributed networks, as a manifestation of data in non-Euclidean domains, have a wide range of applications in the real world, such as molecular property prediction, social network analysis and anomaly detection. Node classification, as a fundamental research problem in attributed networks, has attracted increasing attention among research communities. However, most existing models cannot be directly applied to the data with limited labeled instances (*i.e.*, the few-shot scenario). Few-shot node classification on attributed networks is gradually becoming a research hotspot. Although several methods aim to integrate meta-learning with graph neural networks to address this problem, some limitations remain. First, they all assume node representation learning using graph neural networks in homophilic graphs. Second, existing models based on meta-learning entirely depend on instance-based statistics. Third, most previous models treat all sampled tasks equally and fail to adapt their uniqueness. To solve the above three limitations, we propose a novel graph **Meta**-learning framework called **G**raph learning based on **P**rototype and **S**caling & shifting transformation (**Meta-GPS**). More specifically, we introduce an efficient method for learning expressive node representations even on heterophilic graphs and propose utilizing a prototype-based approach to initialize parameters in meta-learning. Moreover, we also leverage S^2 (scaling & shifting) transformation to learn effective transferable knowledge from diverse tasks. Extensive experimental results on six real-world datasets demonstrate the superiority of our proposed framework, which outperforms other state-of-the-art baselines by up to 13% absolute improvement in terms of related metrics.

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
[SIGIR '22] Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, July 11–15, 2022, Madrid, Spain.
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531978

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Information systems** → **Data mining**.

KEYWORDS

graph neural networks, few-shot learning, node classification

ACM Reference Format:
Yonghao Liu, Mengyu Li, Ximing Li, Fausto Giunchiglia, and Xiaoyue Feng*, Renchu Guan*. 2022. Few-shot Node Classification on Attributed Networks with Graph Meta-learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3531978

1 INTRODUCTION

A large quantity of data can be represented abstractly in the form of networks. Among them, attributed networks are used to model a variety of graph-based systems with many practical application scenarios, such as citation networks [7, 43], social media networks [37, 54], and traffic networks [31, 48], due to their powerful representation capabilities. Among the various analysis tasks of attributed networks, node classification is a very important and fundamental problem. In previous studies [3, 23, 24], graph neural network (GNN)-based methods have been widely used to solve this problem by learning effective low-dimensional embeddings of nodes, with promising results.

However, these classic methods typically require sufficient labeled data to obtain suitable classification performance. Actually, a significant number of node classes in attributed networks contain only a handful of labeled instances [9]. When these methods are directly applied to the few-shot class scenario, they are susceptible to overfitting [57]. Additionally, human labeling is very time-consuming and labor-intensive. Therefore, few-shot learning (FSL) has received increasingly wide academic attention due to its ability to perform well in classifying classes with few labeled data. Recently, to address the FSL problem, numerous meta-learning approaches have been presented for Euclidean data, such as in text domains [10, 19, 20] or image domains [12, 21, 55], and de facto practices have been established. Inspired by the above meta-learning

approaches, many methods for graph meta-learning [9, 18, 28, 57] have been proposed as solutions to the problem of FSL on graph-structured data.

Despite some notable success, existing graph meta-learning methods for FSL on attributed networks still have several limitations, which greatly restrict their learning abilities with graph-structured data. (I) Previous approaches all follow the underlying assumption that attributed networks are homophilic (assortative), where nodes with the same labels tend to be connected [34]. However, this assumption is generally unreasonable since many real-world attributed networks exhibit heterophilic (disassortative) properties, *i.e.*, nodes with distinct classes are often connected. For example, different classes of amino acids tend to react chemically within protein structures [58], and people of different genders are more likely to contact each other in dating graphs [6]. If we directly adopt the neighborhood aggregation mechanism, which is widely used in homophilic graphs, to force the connected nodes' representations to be similar in heterophilic graphs, it greatly impairs the model's performance. (II) Model-agnostic meta learning (MAML) [11], as a prevailing meta-learning method, has been integrated with graph learning due to its excellent properties in solving FSL problems, such as Meta-GNN [57] and G-Meta [18]. Nonetheless, MAML entirely depends on instance-based statistics, which inevitably are affected by data noise and outliers under low-resource settings [10]. One reasonable reason is that MAML uses static, class-independent initialization parameters for all classes. In other words, MAML defaults to using a single initialization for sampled data from different classes in a task and to obtain desired parameters via several gradient steps. However, in attributed networks, the data are not independent and identically distributed (*i.i.d.*) because the inherent dependency between nodes is considered [47]; thus, diverse data sampled from a complex data distribution require different parameters. It is difficult to identify a common initialization that is close to all target parameters. Moreover, a single initialization cannot capture the shared knowledge across classes well, especially when the number of classes is large. (III) Most previous approaches treat all sampled tasks equally. However, there are great differences between tasks. For example, social networks reveal diverse structural patterns for different types of users (*e.g.*, celebrities and regular users) [49]. Therefore, to effectively capture inter-task differences, customizing learnable parameters for each sampled task is beneficial to maximize the model's performance.

To address the limitations mentioned above, we introduce a novel **Meta**-learning framework — **G**raph learning based on **P**rototype and **S**caling & shifting transformation (**Meta-GPS**) for few-shot node classification on attributed networks. Our goal is to develop a classification model that performs effectively for unlabeled nodes whose ground truth labels appear just a few times in the data, as detailed in Section 3. Specifically, Meta-GPS consists of three essential components, including a *network encoder*, *prototype-based parameter initialization*, and S^2 *transformation for different tasks*. To address limitation *I*, we first design a graph layer capable of capturing effective node representations and network structure on heterophilic graphs. To solve limitation *II*, we utilize prototype-aware initialization based on MAML. Moreover, to address limitation *III*, we use two networks to generate scaling and shifting vectors to modulate the prior parameters for dynamically suiting each new task to accommodate the inter-task differences. Overall, the main contributions in our work are as follows:

(1) We propose a novel graph meta-learning framework called Meta-GPS for few-shot node classification, which can quickly adapt to unseen tasks via highly transferable meta-knowledge obtained from meta-training tasks.

(2) We address the limitations by utilizing three key components, which are very effective. To our knowledge, we are the first to consider generalizing FSL to heterophilic graphs and to utilize prototype-based parameter initialization in graph-structured data. As a result, we design a corresponding node representation learning module and parameter initialization module specifically for this purpose. Moreover, we further improve the performance by introducing S^2 transformation.

(3) We conduct extensive experiments on real-world datasets using a wide range of experimental settings. The evaluation results demonstrate the significantly superior performance of the proposed framework Meta-GPS compared to other state-of-the-art models.

2 RELATED WORK

In this section, we briefly review the related work of GNNs, FSL and FSL on graphs, respectively.

Graph Neural Networks. GNNs have attracted tremendous attention from the academic community due to their powerful ability to model graph-structured data. A line of models for GNNs have been developed that aim to solve various practical problems [14, 16, 22, 27, 44, 50]. However, all of the above models are applied to homophilic graphs, while in heterophilic graphs, they achieve unsatisfactory results. As a result, some works are devoted to investigating heterophilic graphs [6, 58]. For example, in [58], the authors present a model called H_2GCN, including several essential designs — higher-order neighborhoods, and combinations of intermediate representations — that facilitate node representation learning under heterophily. In [6], the authors propose a GPR-GNN architecture based on the generalized PageRank algorithm, which can learn the GPR weights and obtain effective node features in heterophilic graphs. Nevertheless, the above models based on GNN cannot directly handle unseen classes with limited labeled instances.

Few-shot Learning. The principal purpose of FSL is to quickly adapt to new tasks with limited labeled instances based on meta-knowledge extracted from many previous training tasks, where meta-knowledge depicts the common features shared among different tasks. If a model is trained directly with few examples, it can easily exhibit overfitting problems. Hence, meta-learning has become the prevalent solution to solve FSL problems. Generally, there are three main types of meta-learning models: *model-based* [4, 39, 59], *metric-based* [40, 41, 45] and *optimization-based* approaches [11, 25, 33, 38]. *Model-based* approaches maintain an internal representation of a task, which can capture task-specific information stored in memory components. For example, CMN [59] employs a key-value memory network schema to yield the optimal video representation in a wider region. *Metric-based* models build a similarity metric between unseen and trained tasks with prior knowledge. For example, prototypical network [40] measures the

Table 1: Descriptions of important symbols

Symbols	Description
$\mathcal{G}, \mathcal{V}, \mathcal{E}$	attributed network, the node and edge set
$\mathbf{X}, \mathbf{A}$	feature matrix, adjacency matrix
$\mathcal{Y}_{train}, \mathcal{Y}_{test}$	seen classes, unseen classes
$\mathcal{T}_{tr}, \mathcal{T}_{te}$	meta-training and meta-testing tasks
$\mathcal{S}_i, \mathcal{Q}_i$	support set and query set in task $\mathcal{T}_i$
$\mathbf{Z}$	final node representations
$\mathbf{P}, \varphi$	class prototype vector, class-specific initialized parameter
$\Theta = \{\theta_e, \theta_p\}$	prior knowledge of Meta-GPS
λ_i, μ_i	scaling and shifting vectors of task $\mathcal{T}_i$
$\Psi = \{\psi_\lambda, \psi_\mu\}$	parameter of S^2 transformation
Θ_i	task-specific parameter for $\mathcal{T}_i$
α, β	step size and meta-learning rate

distance between the support instances' embeddings and their centroids from various classes to perform classification. *Optimization-based* models focus on learning how to optimize model parameters based on gradients from few-shot examples. A famous example is MAML [11], which learns a generic initialization model by optimizing the second-order gradient so that the model converges with few iterations when faced with a new task. A recent work, MIML [10], further extends MAML by leveraging semantic concepts embedded by BERT as meta information, and achieves promising results on few-shot relation classification.

Few-shot Learning on Graphs. Recently, several studies have combined meta-learning and GNNs to achieve large gains in various few-shot scenarios on graph-structured data, such as graph classification [5, 30], link prediction [1, 56] and node classification [9, 18, 28, 29, 49, 52, 57]. Among the models for few-shot node classification, there are three representative ones, namely, Meta-GNN [57], GPN [9] and G-Meta [18]. Meta-GNN adopts MAML to optimize the GNN model, GPN integrates graph structure into Prototypical Network, and G-Meta transfers subgraph-specific information via local subgraphs. Note that other models, such as GFL [52] and MI-GNN [49], incorporate other prevailing techniques [35, 42, 46, 53] and achieve better performance. Nevertheless, they focus on multiple networks and are not the same as our scenario. In addition, the above models for few-shot node classification cannot address the limitations mentioned in the introduction well, while our framework can alleviate them.

3 PROBLEM SETUP

In this section, we formally introduce the studied few-shot node classification problem and elaborate on related details. For convenience, we summarize the most important symbols in this paper in Table 1.

Let us represent an attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{A})$, where $\mathcal{V}$ is the set of nodes $\{v_1, v_2, \ldots, v_n\}$ and $\mathcal{E}$ denotes the set of edges $\{e_1, e_2, \ldots, e_m\}$. Each node is assigned to a $d-$dimensional attribute vector $\mathbf{x}_i \in \mathbb{R}^d$, and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\mathrm{T} \in \mathbb{R}^{n \times d}$ is the attribute matrix of all the nodes. In addition, $\mathbf{A} \in \{0, 1\}^{n \times n}$ indicates the adjacency matrix of an attributed network, where $\mathbf{A}_{ij} = 1$ if two nodes i and j are connected; otherwise $\mathbf{A}_{ij} = 0$.

As defined in [9, 47], given an attributed network $\mathcal{G}$ and the set of labels $\mathcal{Y}$ of its containing nodes, we can divide $\mathcal{Y} = \{y_1, y_2, \ldots, y_l\}$ into two disjoint sets: the *seen* classes $\mathcal{Y}_{train}$ and the *unseen* classes $\mathcal{Y}_{test}$ (*i.e.*, $\mathcal{Y}_{train} \cap \mathcal{Y}_{test} = \emptyset$). Concretely, there exist abundant labeled nodes for each class in $\mathcal{Y}_{train}$ and extremely few labeled nodes (*i.e.*, support set $\mathcal{S}$) for each class in $\mathcal{Y}_{test}$. The ultimate goal of this problem is to develop a classifier that can predict the categories of unlabeled nodes (*i.e.*, query set $\mathcal{Q}$) from *unseen* node classes in $\mathcal{Y}_{test}$ by transferring the meta-knowledge obtained from $\mathcal{Y}_{train}$ during meta-training stage. In particular, if N classes are sampled from $\mathcal{Y}_{test}$ and the support set $\mathcal{S}$ contains K labeled data for each class, we obtain the well-known N-way K-shot classification problem.

Technically, the training of our model also follows the meta-learning paradigm, which generally consists of two phases, called *meta-training* and *meta-testing*. Meta-GPS employs a prevailing episodic training method similar to those of other meta-learning models [11]. Generally, by sampling nodes from $\mathcal{Y}_{train}$, episodic training aims to simulate a genuine test situation, and the model learns general knowledge across various *meta-training* tasks over a substantial number of episodes. Concretely, the entire meta-training procedure is built on a group of Π meta-training tasks $\mathcal{T}_{tr}$, and a meta-training task $\mathcal{T}_i$ includes the support set $\mathcal{S}_i$ and the query set $\mathcal{Q}_i$, which is constructed in each episode. N-way K-shot meta-training tasks $\mathcal{T}_{tr}$ can be denoted as:

$$\begin{aligned} &\mathcal{T}_{tr} = \{\mathcal{T}_i\}_{i=1}^{\Pi}, \qquad \mathcal{T}_i = \{\mathcal{S}_i, \mathcal{Q}_i\}, \\ &\mathcal{S}_i = \{(v_{i,1}, y_{i,1}), (v_{i,2}, y_{i,2}), \ldots, (v_{i,k}, y_{i,k})\}_{k=1}^{N \times K}, \\ &\mathcal{Q}_i = \{(\tilde{v}_{i,1}, \tilde{y}_{i,1}), (\tilde{v}_{i,2}, \tilde{y}_{i,2}), \ldots, (\tilde{v}_{i,k}, \tilde{y}_{i,k})\}_{k=1}^{N \times M} \end{aligned} \tag{1}$$

where the nodes in $\mathcal{S}_i$ and $\mathcal{Q}_i$ are all sampled from $\mathcal{Y}_{train}$. Each category in the support set $\mathcal{S}_i$ has K nodes, while the corresponding category in the query set $\mathcal{Q}_i$ samples M nodes from the remaining data in each class.

In each meta-training task $\mathcal{T}_i$, the support set $\mathcal{S}_i$ is used as the labeled training set, on which the model is trained to minimize its prediction loss for query set $\mathcal{Q}_i$. Note that after performing the meta-training phase, the model collects much transferable meta-knowledge. Finally, in the meta-testing phase, a meta-testing task $\mathcal{T}_{te} = \{\mathcal{S}, \mathcal{Q}\}$ samples the data with unseen classes from $\mathcal{Y}_{test}$ instead of $\mathcal{Y}_{train}$, where the composition of support set $\mathcal{S}$ and query set $\mathcal{Q}$ follow the N-way K-shot manner as those in $\mathcal{T}_{tr}$. Notably, the support set $\mathcal{S}$ with labeled nodes in $\mathcal{T}_{te}$ is used to fine-tune the model for adapting to the N new classes, while the query set $\mathcal{Q}$ with unlabeled nodes is utilized to evaluate the model's performance.

4 PROPOSED FRAMEWORK

In this section, we present the proposed framework Meta-GPS for FSL on attributed networks. To better present our framework, the overall architecture of Meta-GPS is provided in Fig. 1. Subsequently, we detail three key parts of the principled framework: the *network encoder*, *prototype-based parameter initialization* and *S^2 transformation for different tasks*.

4.1 Network Encoder

In general, the first step in graph meta-learning is to learn expressive node representations that consider both node attributes and network topology. Most modern GNNs employ a message passing mechanism that repeatedly updates a node's representation by aggregating representations from its neighbors. Specifically, the l-th

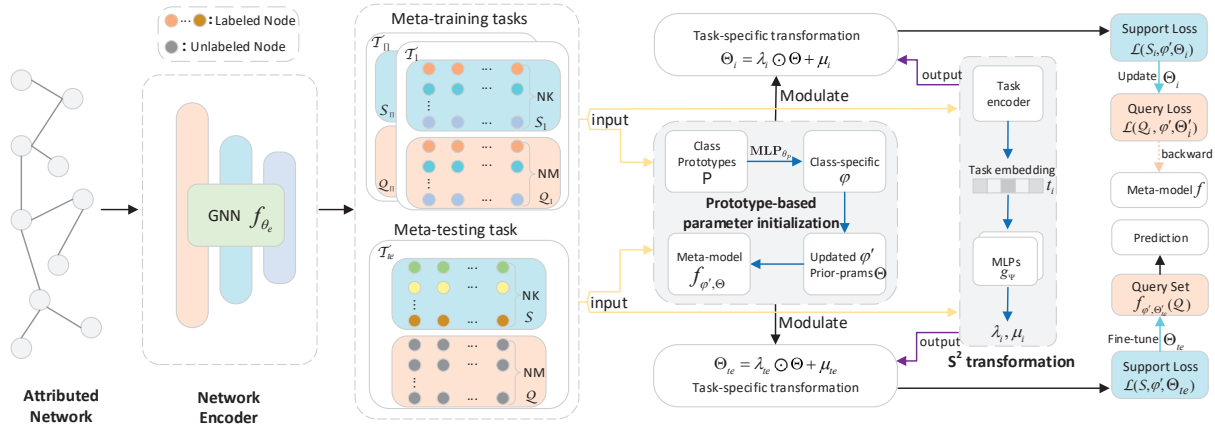


Figure 1: Overall architecture of our framework (best viewed in color).

GNN layer can be formulated as:

$$s_v^l = \textbf{AGGREGATE}(\{h_u^{l-1} : u \in \mathcal{N}_v\})$$
$$h_v^l = \textbf{COMBINE}(h_v^{l-1}, s_v^l) \tag{2}$$

where $\mathcal{N}_v$ is the set of neighbors of node v (**including node** v). Moreover, h_v^l is the updated node v's representation at layer l after performing two crucial functions, *i.e.*, AGGREGATE and COMBINE. There are many GNN architectures [22, 44, 50] due to the different choices of the two functions mentioned above.

However, these GNN-based models perform well in homophilic graphs and poorly in heterophilic graphs. One possible reason is that they do not consider whether the target node and its neighbors belong to the same class. These models simply force the features of neighbors to be added together with those of the target node in a cumulatively aggregated manner, which leads to indistinguishable node representations and inevitably degrades the model's performance [51]. Therefore, a natural idea to address this problem is to update the features of the nodes separately from those of their neighbors to avoid causing their ego-embeddings and neighbor-embeddings to be too similar in heterophilic graphs. More precisely, we replace s_v^l in Eq. 2 with $s_{v,i}^l = \textbf{AGGREGATE}(\{h_u^{l-1} : u \in \widetilde{\mathcal{N}}_i(v)\})$, where $\widetilde{\mathcal{N}}_i(v)$ is the set of i-hop neighbors of node v (**excluding node** v). Additionally, to avoid mixing the node's own features with those of its neighbors, the **COMBINE** function can adopt a concatenation operator instead of the average [22] or weighted average [44], which can be more expressive since non-aggregated representations can evolve separately over multiple rounds of propagation. Eventually, the embeddings learned by different classes of nodes are easily distinguishable.

Inspired by this, we design a graph layer that performs well in heterophilic attributed networks, which can be formulated below:

$$\mathbf{F} = \sigma(\mathbf{X}\mathbf{W}_f), \quad \mathbf{H}_0 \equiv \mathbf{F}, \quad \mathbf{H}_i = \tilde{\mathbf{A}}_i\mathbf{F},$$
$$\mathbf{R} = ||_{i=0}^{\ell}\mathbf{H}_i, \quad \eta = \sigma(\mathbf{R}\mathbf{W}_r), \quad \mathbf{Z} = \mathrm{Squ}(\mathrm{Res}(\eta)\mathbf{R}) \tag{3}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\sigma(\cdot)$ denote the input node features and non-linear transformation, respectively. $\tilde{\mathbf{A}}_i = \bar{\mathbf{D}}_i^{-1/2}\bar{\mathbf{A}}_i\bar{\mathbf{D}}_i^{-1/2}$ represents the i-hop neighbors' normalized symmetric adjacency matrix **without self-loops**, where $\bar{\mathbf{A}}_i$ is the i-hop neighbors' adjacency matrix[1] and $\bar{\mathbf{D}}_i$ is its corresponding diagonal degree matrix. $||$ is the concatenation operator. Moreover, $\mathbf{F}, \mathbf{H}_i, \mathbf{Z} \in \mathbb{R}^{n\times d'}$ symbolize the intermediate node representations, the i-hop aggregated neighbors' embeddings, and the output node embeddings, respectively. $\mathbf{R} \in \mathbb{R}^{n\times(\ell+1)\times d'}$ is the concatenated embeddings, and $\eta \in \mathbb{R}^{n\times(\ell+1)\times 1}$ is the attention coefficient to measure how much information of the aggregated neighbors' representations obtained from different-hop neighbors should be maintained to generate the final representation of each node. Squ and Res represent 'squeeze' and 'reshape' operations to match the matrix's dimensions, respectively.

In the above approach, we first perform the feature transformation process for all nodes without considering structure information. Then, the aggregated neighbor-embeddings without ego are derived by aggregating the i-hop neighbors' features. These ego-embeddings and different-hop neighbor-embeddings are concatenated to obtain the intermediate representations $\mathbf{R}$. The underlying idea is that nodes and their non-local neighbors are likely to have similar patterns in heterophilic graphs [26]. Finally, the attention mechanism is applied to the intermediate representation to select informative features for classifying nodes.

By adopting the above approach to encoding networks, our framework can learn more expressive node representations in heterophilic networks. Additionally, we denote the prior parameters of the graph layer as $\theta_e = \{\mathbf{W}_f, \mathbf{W}_r\}$.

4.2 Prototype-based Parameter Initialization

A prototype refers to the feature representation of a specific category. To obtain each class's prototype vector in $\mathcal{Y}_{train}$, we can average the embedded nodes belonging to that class. This operation can be defined as follows:

[1]For example, $\bar{\mathbf{A}}_1 = \mathbb{I}(\mathbf{A} - \mathbf{I} > 0)$, $\bar{\mathbf{A}}_2 = \mathbb{I}(\mathbf{A}^2 - \mathbf{A} - \mathbf{I} > 0)$, where $\mathbb{I}(\cdot)$ is the indicator matrix and $\mathbf{I}$ is the identity matrix.

$$\mathbf{P}_j = \frac{1}{|\mathcal{V}_j|} \sum\nolimits_{k \in \mathcal{V}_j} \mathbf{Z}_k \tag{4}$$

where $\mathbf{Z}_k$ is the features of node v_k after implementing the graph layer. $\mathcal{V}_j$ represents the set of nodes whose labels belong to class j. The obtained prototype can reflect the characteristics of a specific class of data, which can be viewed as the meta-information of a class.

Existing graph meta-learning methods based on MAML rely on a single initialization, while nodes on attributed networks do not follow the *i.i.d* assumption. This difference makes it difficult to find a single initialization near all of the desired parameters. In contrast, to achieve better performance, we use the prototype vector as auxiliary information to provide class-specific initialized parameters as strong guidance to enable our model to rapidly seek the most potential parameters via a few gradient steps. Specifically, given the prototype $P_j \in \mathbb{R}^{d'}$ of a class j, we can feed it to the neural network to obtain the class-specific initialized parameters $\varphi_j \in \mathbb{R}^{d'}$ for class j. The formula is as follows:

$$\varphi_j = \mathbf{MLP}(\mathbf{P}_j; \theta_p), \text{where} j = 1, \cdots, N \tag{5}$$

where N is the number of categories of a task (*i.e.*, N-way). The neural network we employed is **MLP** parameterized by θ_p for efficiency and simplicity. Equivalently, other sophisticated networks can be applied, such as convolutional networks or recurrent networks. In this way, the initialized parameters incorporate the prototype information and become category-dependent.

Moreover, when $\varphi \in \mathbb{R}^{N \times d'}$ is directly applied to the labeled nodes in support set $\mathcal{S}_i$ by a feed-forward layer, *i.e.*, $score = \text{softmax}(\mathbf{Z}\varphi^\top + b)$, then $score \in \mathbb{R}^{n \times N}$ can be taken as the probability that a node belongs to a specific class. Subsequently, φ can be further adapted via gradient steps, depending on the classification performance applied to concrete examples in support set $\mathcal{S}_i$ of task $\mathcal{T}_i$, denoted as:

$$\varphi_i' = \varphi - \alpha \nabla_\varphi \mathcal{L}_{\mathcal{T}_i}(f(\mathcal{S}_i; \varphi, \Theta)) \tag{6}$$

where $\Theta = \{\theta_e, \theta_p\}$ and $\mathcal{L}_{\mathcal{T}_i}(\cdot)$ are prior parameters and the cross-entropy loss function described in Eq. 9 below, respectively. Eq. 6 can also be extended to multiple gradient descent steps, similar to MAML. By the procedure of prototype-based parameter initialization (*PI*), the adapted parameters are restricted not to deviate much from the class prototype, even if there are some outliers.

4.3 S^2 Transformation for Different Tasks

Dissimilar tasks have varying distributions of node features and categories. If we directly adopt $score = \text{softmax}(\mathbf{Z}\varphi^\top + b)$ as described in Section 4.2 as the final output, the model cannot capture the inter-task differences. Thus, we employ a conditioning mechanism that customizes the particular parameters for different tasks.

We leverage S^2 transformation to produce a set of parameters to modulate the prior learned meta-parameters conditioned on a given task. Specifically, we first obtain the task-level representation vector t_i using the mean of the node embeddings contained in a given task $\mathcal{T}_i$, which is simple but effective compared to other expensive methods of obtaining task representations, such as RNNs, CNNs or attention mechanisms. Moreover, in this way, tasks with more similar categories tend to be closer in the task representation space. Then, we implement two separate neural networks to generate the scaling vector λ_i and shifting vector μ_i that encode the characteristics of the given task. The specific formula is given by:

$$t_i = \frac{1}{|\mathcal{V}_{t_i}|} \sum\nolimits_{k \in \mathcal{V}_{t_i}} \mathbf{Z}_k, \quad \lambda_i = g(t_i; \psi_\lambda), \quad \mu_i = g(t_i; \psi_\mu) \tag{7}$$

where $\mathcal{V}_{t_i}$ denotes the set of nodes involved in $\mathcal{T}_i$, and $\lambda_i, \mu_i \in \mathbb{R}^{|\Theta|}$ are S^2 coefficients, where $|\Theta|$ is the number of parameters in Θ. Theoretically, the two separated neural networks g can be arbitrarily parameterized functions. Here, we employ two **MLPs** in line with the neural network used in *prototype-based parameter initialization*, parameterized by ψ_λ and ψ_μ.

Finally, the derived scaling and shifting vectors are applied to the task's prior meta-parameters Θ, as follows:

$$\Theta_i = \lambda_i \odot \Theta + \mu_i \tag{8}$$

where $\odot$ is element-wise multiplication since λ_i, μ_i and Θ have the same dimension. This also means that each element in λ_i, μ_i regulates the weights of neurons suited to each task. With the above customization transformation, globally transferable knowledge is adapted to the task-specific parameter θ_i such that similar tasks produce similar scaling and shifting vectors and make it easier to share more information. In essence, the S^2 module does not directly specify the transformation, but it encodes the rules of how to perform the transformation for each task. This is equivalent to a type of hypernetwork [15], in which the task's prior θ is adjusted by a secondary network (parameterized by Ψ) to accommodate the changing input task. Note that the prototype-based parameters φ are kept fixed in the S^2 transformation and that only the meta-learned prior parameters Θ of the task network are updated.

4.4 Meta-optimization

To enable our framework to effectively adapt to new tasks with few labeled nodes, we train it in a meta-learning manner. Following the concept of MAML, Meta-GPS learns using gradient-based optimization. Specifically, the prior parameters are adapted for each task $\mathcal{T}_i$ by performing a few gradient descent steps on the loss of its support set $\mathcal{S}_i$. Then, the task loss computed on its query set $\mathcal{Q}_i$ is backpropagated to optimize the prior parameters.

Meta-training. To suit a specific task $\mathcal{T}_i$, the nodes in support set $\mathcal{S}_i$ are fed to the model, and then the corresponding task loss $\mathcal{L}_{\mathcal{T}_i}$ is calculated via the cross-entropy loss function. The corresponding formula can be expressed as:

$$\begin{aligned}\mathcal{L}_{\mathcal{T}_i}(\mathcal{S}_i, \varphi_i', \Theta_i) = -\sum_{(v_i, y_i) \in \mathcal{S}_i} \Big(& y_i \log f(v_i; \varphi_i', \Theta_i) + \\ & (1 - y_i) \log(1 - f(v_i; \varphi_i', \Theta_i)) \Big)\end{aligned} \tag{9}$$

where φ_i' and Θ_i are the updated prototype-based initialized parameters in Eq. 6 and task-specific parameters for task $\mathcal{T}_i$, respectively. In addition, $f(*; \varphi_i', \Theta_i)$ is the output layer of the Meta-GPS parameterized by φ_i', Θ_i.

We update Θ_i to Θ_i' that is adapted to task $\mathcal{T}_i$ via one gradient descent step based on the above support task loss, as follows:

$$\Theta_i' = \Theta_i - \alpha \nabla_{\Theta_i} \mathcal{L}_{\mathcal{T}_i}(f(\mathcal{S}_i; \varphi_i', \Theta_i)) \tag{10}$$

Algorithm 1: Graph meta-learning network training

Data: Task Distribution $p(\mathcal{T})$, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{A})$, step sizes α and meta-learning rate β
Result: Trained graph meta-learning model Meta-GPS

randomly initialize: $\Theta = \{\theta_e, \theta_p\}, \Psi = \{\psi_\lambda, \psi_\mu\}$;
while *not done* **do**
  sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$;
  compute each prototype $\mathbf{P}_j$ of $\mathcal{Y}_{train}$ with Eq. 4;
  for *each task* $\mathcal{T}_i$ **do**
    sample support set $\mathcal{S}_i$ and query set $\mathcal{Q}_i$ for task $\mathcal{T}_i$;
    obtain node representations in $\mathcal{S}_i$ and $\mathcal{Q}_i \leftarrow$ Sec.4.1;
    for *j=1 to N* **do**
      $\mathbf{P}_j$-specific parameter initialize φ_j with Eq. 5;
    end
    compute updated φ with gradient descent with Eq. 6;
    compute S^2 vectors λ_i, μ_i with Eq. 7;
    obtain Θ_i to suit $\mathcal{T}_i$ with Eq. 8;
    compute support loss using $\mathcal{S}_i$ with Eq. 9;
    compute adapted parameter Θ_i' with Eq. 10;
    compute query loss using $\mathcal{Q}_i$;
  end
  update Θ, Ψ by backpropagation of total query loss with Eq. 12.
end
return the trained Meta-GPS.

where α is the meta-step size. It is straightforward to extend Eq. 10 to multiple step gradient updates.

After obtaining the updated Θ_i', we utilize it to calculate the loss of query set $\mathcal{Q}_i$ for each task $\mathcal{T}_i$, *i.e.*, $\mathcal{L}_{\mathcal{T}_i}(\mathcal{Q}_i, \varphi_i', \Theta_i)$. Given the distribution of meta-training tasks $p(\mathcal{T})$, our goal is to minimize the meta-objective function, which is defined as follows:

$$\min_{\Theta,\Psi} \mathcal{L}(f_{\varphi',\Theta}, g_\Psi) = \min_{\Theta,\Psi} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f(\mathcal{Q}_i; \varphi_i', \Theta_i')) + \gamma \|\Psi\|_2^2 \quad (11)$$

where γ is a hyperparameter to adjust the parameter regularizer and $\Psi = \{\psi_\lambda, \psi_\mu\}$.

Finally, stochastic gradient descent is used to conduct meta-optimization across tasks, and the model parameters are updated as follows:

$$\Theta = \Theta - \beta \nabla_\Theta \mathcal{L}(f_{\varphi',\Theta}, g_\Psi), \quad \Psi = \Psi - \beta \nabla_\Psi \mathcal{L}(f_{\varphi',\Theta}, g_\Psi) \quad (12)$$

where β is the meta-learning rate. The detailed training procedure of the proposed framework Meta-GPS is presented in Algorithm 1.

Meta-testing. During the meta-testing phase, we generally follow the same procedure as during the meta-training. Specifically, the learned prior parameters Θ, Ψ are adapted to support set $\mathcal{S}$ in meta-testing task $\mathcal{T}_{te}$ via one or a small number of gradient descent steps. Then, we evaluate the model's performance on the query set $\mathcal{Q}$ in $\mathcal{T}_{te}$.

Discussion. Since our work is about few-shot learning, there will be different scenarios (*i.e.*, heterogeneous/homogenous and directed/undirected graphs) where different types of edges exist Although in this work we apply the proposed model to homogeneous graphs, this does not mean that it is not applicable in other scenarios. We simply need to modify the graph layers in the network encoder phase to obtain the corresponding node representations, and the model still achieves satisfactory results.

5 EXPERIMENTS

Table 2: Statistics of the evaluation datasets

Dataset	# Nodes	# Edges	# Features	# Labels	# Homophily
Reddit	232,965	11,606,919	602	41	0.81
ogbn-products	2,449,029	61,859,140	100	47	0.83
Amazon-Clothing	24,919	91,680	9,034	77	0.62
Cora-Full	19,793	65,311	8,710	70	0.59
Amazon-Electronics	42,318	43,556	8,669	167	0.39
DBLP	40,672	288,270	7,202	137	0.29

Datasets. Since FSL models need to be trained with a large number of tasks consisting of nodes from various categories in the meta-training phase to extract effective meta-knowledge, the widely used node classification benchmark datasets containing few categories are not suitable for this work. In our experiments, we employ six real-world datasets for few-shot node classification used in earlier research [8, 9, 47], which are described in detail as follows:

(I) **Reddit** [16] is a post-to-post graph extracted from the Reddit internet forum. Nodes represent posts, and two posts are linked if they are both commented by the same person. Additionally, the label for each node is the community to which the corresponding post belongs. The attributes of nodes are embedded by off-the-shelf 300-dimensional GloVe word vectors. (II) **ogbn-products** [17] is a product co-purchasing network from Open Graph Benchmark (OGB). In this network, nodes represent products, and the presence of edges between two products indicates that they are purchased jointly. The node label is the product category. Moreover, the features of nodes are obtained by bag-of-words of the product description. (III) **Amazon-Clothing** [32] is an Amazon product network made up of items from the categories of "Clothing, Shoes and Jewelry". Each product is regarded as a node, and the node features are built using its description. The edges between two products denote that they are viewed by the same user. In addition, the fine-grained product categories determine the node labels. (IV) **Cora-Full** [2] is a paper citation network, a full version of well-known Cora dataset. In this dataset, nodes represent papers and edges represent citation links. The node is labeled based on the paper topic. Node attributes are generated by using bag-of-words for the title and abstract of the paper. (V) **Amazon-Electronics** [32] is a network within the "Electronics" category of Amazon products. Similar to 'Amazon-Clothing', nodes mean products, and edges between two products are created if they are bought together. The product description is used to generate node features. Node class labels are made up of low-level product categories. (VI) **DBLP** [43] is also a citation network where papers are considered nodes and the link is a citation relationship of papers. The abstracts of papers are utilized to create node features, and the papers are labeled with the venues.

For these datasets, we use the same train/validation/test splits as in previous studies [9, 47]. The comprehensive statistics of the above datasets can be found in Table 2. Note that "#Homophily" in Table 2 is used to measure the homophily level in a network. Here, we

adopt the node homophily $\mathbf{H} = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\frac{|\{(u,v):u\in\mathcal{N}_v\wedge y_u=y_v\}|}{|\mathcal{N}_v|}$, defined in [34], where higher values denote stronger homophily and lower values indicate stronger heterophily.

Baselines. To verify the superiority of our proposed model, a series of baseline models are selected for comparison. Detailed information on the baseline models compared is listed below.

• **DeepWalk** [36]: It first extracts some vertex sequences from the graph using the random walk algorithm, and then performs the skip-gram model for sampled sequences to learn low-dimensional node embeddings.

• **Node2vec** [13]: It is an enhanced version of the DeepWalk model with biased random walks, allowing the generated random walks to reflect the characteristics of both depth-first and breadth-first sampling, thus improving the effectiveness of network embeddings.

• **GCN** [22]: It leverages the first-order approximation of the Chebyshev polynomial to mimic the graph convolution for learning latent node embeddings.

• **SGC** [50]: It is modified from vanilla GCN, which reduces unnecessary complexity by removing the layer-wise non-linear activation function. It finally converts the convolution function of GCN into a linear transformation.

• **Protonet** [40]: Prototypical network is a classical metric-based meta-learning method. It maps the sample data of each class into a space and computes their mean to represent the class as a prototype.

• **MAML** [11]: It is a well-known optimization-based meta-learning method that enables fast adaptation to new tasks with only few data by performing one or more gradient update steps on top of the model's random initial parameters.

• **Meta-GNN** [57]: It directly combines GNNs with standard MAML for graph few-shot node classification, adapting to new tasks according to prior knowledge obtained from previous tasks.

• **GPN** [9]: It learns the prototype of each class with a GNN-based encoder and evaluator that measures the importance of the nodes. Finally, its label is predicted by calculating the Euclidean distance between the node and the prototype.

• **G-Meta** [18]: The core idea of this model is to create a subgraph for each node and use local subgraphs to learn transferable knowledge via meta gradients.

The above baselines can be divided into four main categories. (I) *Random walk-based models* include two representative unsupervised methods, DeepWalk [36] and node2vec [13]. Note that we apply a Logistic Regression classifier for node classification after obtaining node representations. (II) *GNN-based models* consist of two widely used methods, *i.e.*, GCN [22] and SGC [50]. (III) *Classic meta-learning models* contain Protonet [40] and MAML [11], mainly for *i.i.d* data. (IV) *Graph meta-learning models* include Meta-GNN [57], GPN [9] and G-Meta [18], which are designed specifically for graph structure data. To suit the few-shot node classification task, for the first two types of baseline models, we follow the experimental setup in [57]. For other remaining baselines, we use the parameters recommended by their original papers.

Model Implementation. In the *network encoder* stage, we adopt 2-hop neighborhoods, *i.e.*, $\ell = 2$ in Eq. 3. Additionally, we set the dimension d' of final learned embeddings to 16. For all MLPs employed in the last two stages in Eqs. 5 and 7, we uniformly use one hidden layer with 16 units activated by the ReLU function and one corresponding linear output layer. The step size α, the meta-learning rate β and the regularization coefficient γ are set to be 0.5, 0.001 and 0.001, respectively. Notably, the task number in every batch and the query size M of each task are both set to 10. Moreover, the steps of gradient descent update in meta-training and meta-testing are uniformly 5. Finally, we perform an early-stopping strategy to avoid overfitting according to the performance of the validation set. If the loss of the validation set does not decrease for 50 consecutive episodes, we stop the model training.

Evaluation Metric. The evaluation metrics in our experiments are Accuracy (ACC) and Micro-F1 score (F1), which are widely used in few-shot node classification on attributed networks.

6 RESULTS

Model Performance. We conduct extensive experiments on the proposed method and other baseline approaches for each dataset under four few-shot settings, including 5-way 3-shot, 5-way 5-shot, 10-way 3-shot and 10-way 5-shot. To guarantee experimental fairness and stability, we randomly sample 200 meta-testing tasks from $\mathcal{Y}_{test}$ to evaluate each model's performance. Furthermore, the above procedure is repeated ten times, and the average experimental results are presented in Table 3. Based on the quantitative results, we elaborate on our in-depth observations and analyses as follows.

• Overall, our model achieves significantly superior performance on all datasets under different experimental settings compared to other competitive baselines, illustrating its powerful generalization capability. The reason is that our model produces prototype-based parameter initialization instead of a single random initialization, which provides category-guidance information, making it possible to alleviate the reliance on support data and to enhance the robustness of the model to outliers. We also employ scaling and shifting vectors to provide customized trainable parameters for training tasks; this allows for a more efficient extraction of transferable knowledge for adaptation to new tasks.

• We find that Meta-GPS achieves the best performance in both homophilic and heterophilic attributed networks, while other baselines generally perform relatively poorly on heterophilic datasets. An interesting finding is that as the homophily level of a network decreases, our model achieves higher absolute improvements compared to other baselines. Taking the example of our model's performance on the task of 5-way 3-shot in Reddit, Cora-Full and Amazon-Electronics, the model achieves 4%, 7.6% and 13.3% absolute improvements in Accuracy, respectively, compared to that of the best-performing baseline G-Meta. We attribute this to the use of a specifically designed graph layer for learning more expressive node representations in a non-aggregated manner. When a network exhibits high homophily, both our designed graph layer and classic graph layers can extract discriminative features of different node classes. In contrast, when the network is highly heterophilic, other models still cumulatively learn node representations, which introduces more noise to the learned node representations and is not beneficial for learning the essential patterns of different classes.

• Random walk-based models (*e.g.* DeepWalk, node2vec) and GNN-based models (*e.g.* GCN, SGC) achieve unsatisfactory performance in FSL scenarios, lagging far behind graph meta-learning models.

Table 3: Results of the Accuracy and Micro-F1 score on various datasets. Underline: runner-up. Ab Impro: absolute improvement.

Method	Reddit (H=0.81)								ogbn-products (H=0.83)							
	5-way 3-shot		5-way 5-shot		10-way 3-shot		10-way 5-shot		5-way 3-shot		5-way 5-shot		10-way 3-shot		10-way 5-shot	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DeepWalk	0.267	0.261	0.301	0.297	0.176	0.171	0.188	0.186	0.249	0.241	0.279	0.264	0.167	0.155	0.192	0.179
node2vec	0.271	0.256	0.312	0.298	0.198	0.186	0.234	0.226	0.253	0.247	0.305	0.299	0.176	0.163	0.212	0.202
GCN	0.388	0.381	0.455	0.441	0.290	0.270	0.357	0.324	0.341	0.326	0.394	0.375	0.256	0.259	0.302	0.295
SGC	0.444	0.421	0.468	0.425	0.297	0.268	0.316	0.277	0.375	0.366	0.412	0.406	0.294	0.301	0.337	0.331
Protonet	0.346	0.333	0.376	0.364	0.198	0.180	0.233	0.214	0.291	0.286	0.305	0.297	0.216	0.195	0.276	0.263
MAML	0.291	0.268	0.311	0.291	0.152	0.122	0.179	0.156	0.351	0.343	0.402	0.391	0.262	0.241	0.294	0.270
Meta-GNN	0.608	0.583	0.627	0.612	0.449	0.421	0.515	0.471	0.492	0.504	0.523	0.514	0.347	0.349	0.401	0.396
GPN	0.655	0.662	0.684	0.690	0.534	0.558	0.577	0.592	0.556	0.532	0.572	0.566	0.396	0.392	0.425	0.412
G-Meta	0.726	0.721	0.735	0.731	0.583	0.571	0.621	0.616	0.605	0.599	0.624	0.611	0.461	0.452	0.491	0.480
Meta-GPS	**0.766**	**0.753**	**0.779**	**0.772**	**0.627**	**0.606**	**0.655**	**0.651**	**0.654**	**0.646**	**0.665**	**0.653**	**0.509**	**0.497**	**0.531**	**0.522**
Ab Impro	4%	3.2%	4.4%	4.1%	4.4%	3.5%	3.4%	3.5%	4.9%	4.7%	4.1%	4.2%	3.8%	4.5%	4%	4.2%

Method	Amazon-Clothing (H=0.62)								Cora-Full (H=0.59)							
	5-way 3-shot		5-way 5-shot		10-way 3-shot		10-way 5-shot		5-way 3-shot		5-way 5-shot		10-way 3-shot		10-way 5-shot	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DeepWalk	0.367	0.363	0.465	0.466	0.213	0.191	0.353	0.329	0.236	0.221	0.259	0.247	0.153	0.132	0.170	0.157
node2vec	0.362	0.358	0.419	0.407	0.175	0.151	0.326	0.302	0.237	0.219	0.254	0.235	0.139	0.125	0.152	0.146
GCN	0.543	0.514	0.593	0.566	0.413	0.375	0.448	0.403	0.346	0.339	0.398	0.376	0.292	0.284	0.341	0.352
SGC	0.568	0.552	0.622	0.615	0.431	0.416	0.463	0.447	0.395	0.402	0.445	0.432	0.351	0.336	0.395	0.389
Protonet	0.537	0.536	0.635	0.637	0.415	0.419	0.448	0.462	0.336	0.311	0.365	0.339	0.249	0.245	0.272	0.261
MAML	0.552	0.545	0.661	0.678	0.456	0.433	0.468	0.456	0.371	0.352	0.475	0.461	0.266	0.259	0.316	0.312
Meta-GNN	0.721	0.726	0.748	0.775	0.614	0.597	0.642	0.629	0.522	0.510	0.591	0.571	0.472	0.461	0.533	0.526
GPN	0.724	0.732	0.761	0.790	0.650	0.661	0.677	0.679	0.532	0.523	0.603	0.594	0.509	0.491	0.562	0.556
G-Meta	0.732	0.731	0.766	0.786	0.663	0.657	0.680	0.671	0.575	0.569	0.624	0.616	0.539	0.526	0.581	0.575
Meta-GPS	**0.803**	**0.796**	**0.826**	**0.816**	**0.731**	**0.732**	**0.741**	**0.744**	**0.651**	**0.643**	**0.692**	**0.686**	**0.612**	**0.603**	**0.642**	**0.641**
Ab Impro	6.1%	6.4%	6%	5.9%	6.8%	7.1%	6.1%	6.5%	7.6%	7.4%	6.8%	7%	7.3%	7.7%	6.1%	6.6%

Method	Amazon-Electronics (H=0.39)								DBLP (H=0.29)							
	5-way 3-shot		5-way 5-shot		10-way 3-shot		10-way 5-shot		5-way 3-shot		5-way 5-shot		10-way 3-shot		10-way 5-shot	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DeepWalk	0.235	0.222	0.261	0.257	0.147	0.129	0.160	0.147	0.447	0.431	0.624	0.604	0.338	0.308	0.451	0.430
node2vec	0.255	0.237	0.271	0.243	0.151	0.131	0.177	0.155	0.407	0.385	0.586	0.572	0.315	0.278	0.412	0.396
GCN	0.538	0.498	0.596	0.553	0.423	0.384	0.474	0.483	0.596	0.549	0.683	0.66	0.439	0.390	0.512	0.476
SGC	0.546	0.534	0.608	0.594	0.432	0.415	0.500	0.476	0.573	0.547	0.650	0.621	0.402	0.368	0.503	0.464
Protonet	0.535	0.556	0.597	0.615	0.399	0.400	0.450	0.448	0.372	0.367	0.434	0.443	0.262	0.260	0.326	0.328
MAML	0.533	0.521	0.590	0.583	0.374	0.361	0.434	0.413	0.397	0.397	0.455	0.437	0.308	0.253	0.347	0.312
Meta-GNN	0.632	0.615	0.679	0.668	0.582	0.558	0.608	0.601	0.709	0.703	0.742	0.744	0.607	0.604	0.631	0.622
GPN	0.646	0.628	0.709	0.706	0.603	0.607	0.624	0.637	0.745	0.739	0.761	0.758	0.626	0.626	0.640	0.644
G-Meta	0.695	0.691	0.736	0.726	0.664	0.646	0.680	0.672	0.751	0.743	0.776	0.769	0.632	0.636	0.642	0.646
Meta-GPS	**0.828**	**0.827**	**0.852**	**0.830**	**0.785**	**0.782**	**0.793**	**0.791**	**0.849**	**0.847**	**0.859**	**0.855**	**0.759**	**0.751**	**0.779**	**0.772**
Ab Impro	13.3%	13.6%	11.6%	10.4%	12.1%	13.6%	11.3%	11.9%	9.8%	10.4%	9.3%	8.6%	12.7%	11.5%	11.7%	12.6%

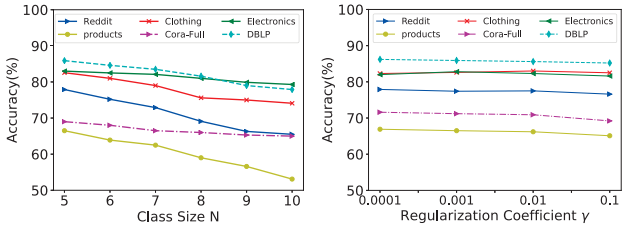
Table 4: Test accuracy on different datasets with various noisy data ratios. Ab Impro: absolute improvement.

Model	Reddit (H=0.81)						Amazon-Clothing (H=0.62)						Amazon-Electronics(H=0.39)					
	5-way 5-shot			10-way 5-shot			5-way 5-shot			10-way 5-shot			5-way 5-shot			10-way 5-shot		
	10%	20%	30%	10%	20%	30%	10%	20%	30%	10%	20%	30%	10%	20%	30%	10%	20%	30%
Meta-GNN	0.603	0.576	0.553	0.490	0.465	0.439	0.747	0.721	0.692	0.619	0.601	0.579	0.647	0.615	0.586	0.590	0.566	0.542
GPN	0.662	0.637	0.611	0.556	0.531	0.502	0.761	0.732	0.697	0.651	0.625	0.594	0.682	0.651	0.633	0.610	0.601	0.576
G-Meta	0.713	0.693	0.670	0.605	0.582	0.556	0.737	0.715	0.696	0.659	0.637	0.619	0.714	0.697	0.675	0.661	0.633	0.606
Meta-GPS	**0.771**	**0.759**	**0.742**	**0.646**	**0.635**	**0.619**	**0.819**	**0.807**	**0.792**	**0.736**	**0.725**	**0.710**	**0.841**	**0.829**	**0.814**	**0.786**	**0.775**	**0.759**
Ab Impro	5.8%	6.6%	7.2%	4.1%	5.3%	6.3%	8.2%	9.2%	9.6%	7.7%	8.8%	9.1%	12.7%	13.2%	13.9%	12.5%	14.2 %	15.3 %

A possible reason is that these models typically require large quantities of labeled data to obtain competitive results and suffer from severe overfitting problems when only a small number of labeled nodes are available. Notably, despite the fruitful success of traditional meta-learning methods (*e.g.* Protonet, MAML) in few-shot image classification, these methods perform worse than the GNN-based models in our few-shot node classification scenarios. This can be attributed to the fact that they neglect the graph topology information of attributed networks, which adversely affects the learned node representation. Since simultaneously considering the meta-learning and graph topology, graph meta-learning models (*e.g.* Meta-GNN, GPN, G-Meta) obtain significant gains compared to the other two types of models. However, due to the limitations mentioned in the introduction, their performance is still far behind Meta-GPS.

Robustness to Noisy Data. As mentioned in the introduction, existing methods based on meta-learning rely entirely on instance-based statistics, and their performance is inevitably greatly affected by data noise. To further verify the robustness of Meta-GPS when the support instances are noisy, following [10], we randomly convert a certain percentage of the support instances of each class to noisy instances sampled from other classes. Since we sample many meta-tasks for training, it can be thought that a certain percentage of noise exists in each class of the meta-task. Table 4 shows that Meta-GPS is the least susceptible to data noise as the noisy

data ratio increases, illustrating its powerful robustness and noise-reduction. This is mainly due to the prototype-based parameter initialization module, which helps Meta-GPS overcome the inherent shortcomings of the MAML-based methods and achieve significant superiority. Additionally, other models show different degrees of performance degradation when subjected to data noise disturbances, illustrating that most of the existing graph meta-models are vulnerable to noisy data. Moreover, it shows the same trend as Table 3 that as the network's heterophily increases, Meta-GPS achieves greater improvements, which is in accordance with our expectation.



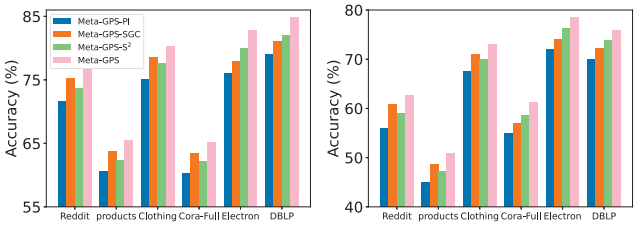
(a) N-way 5-shot (b) 5-way 5-shot

Figure 2: Evaluation performance for different hyperparameters. (Other settings show similar trends, omitted for space.)



(a) Meta-GPS (b) Meta-GPS-SGC

Figure 3: Visualization of learned node embeddings by different methods on partial classes of the test set of DBLP dataset.



(a) 5-way 3-shot (b) 10-way 3-shot

Figure 4: Ablation results of model variants on all datasets.

Hyperparameter Sensitivity. We further empirically exploit the effects of the choices of different hyperparameters by observing the performance of our model on all evaluated datasets. Specifically, the primary hyperparameters used in Meta-GPS are the node class size (N-way) and the regularization coefficient γ in Eq. 9, respectively. Fig. 2 shows the test accuracy results with different hyperparameters. The model's performance decreases with increasing node class N, indicating that more uncertainty is introduced when the node classes N increase. Moreover, as shown in Fig. 2 (b), for smaller γ values, our model tends to perform better. This is because the large value leads to very little scaling and shifting, significantly reduce the role of the S^2 transformation and thus suffers performance degradation.

Ablation Study. We design some model variants to study the ablation experiments of three crucial components to examine whether they play an important role in our model. (I) *Meta-GPS-SGC:* In this version, we replace our designed graph layer with simplifying graph convolution to encode attributed networks. (II) *Meta-GPS-PI:* In this version, we remove the prototype-based initialization parameters and use random initialization parameters. (III) *Meta-GPS-S^2:* In this version, we exclude the S^2 transformation that generates the scaling and shifting vectors and treat all tasks equally. First, we conduct visualization experiments to verify that concatenating is better than averaging in learning the node embedding. From Fig. 3, we observe that our designed graph layer using the concatenation operator can lead to better class separation compared to SGC using the average operator, which further indicates that the designed graph layer can work well in heterophilic networks. Moreover, we can obtain the following insights and analysis by observing the results presented in Fig. 4. First, when we eliminate any of the modules, the model's performance receives a different degree of degradation, indicating that all these parts play a critical role in the model. Second, Meta-GPS-SGC performs comparably in homophilic networks but poorly in heterophilic networks. One reasonable explanation is that Meta-GPS-SGC cannot learn powerful node representations from heterophilic networks due to mixing the ego- and neighbor- embeddings. Third, Meta-GPS-PI obtains the worst performance among all model variants, illustrating that adopting prototype-based initialization facilitates the model to learn the optimal parameters and provides guarantees for stable model learning. Finally, customizing the weight parameters for each task is beneficial to improve the generalization of the model.

7 CONCLUSION

In this work, we propose a novel graph meta-learning framework, namely, Meta-GPS, to address the challenge of few-shot node classification on heterophily attributed networks. More precisely, we design three components called the network encoder, prototype-based parameter initialization, and S^2 transformation for suiting different tasks, which can effectively address the limitations of previous approaches. The empirical results of extensive experiments on real-world datasets under various settings demonstrate the effectiveness of our proposed framework compared with that of other state-of-the-art baseline models.

ACKNOWLEDGMENTS

The authors are grateful for the support of the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, National Key R&D Program of China [No. 2021ZD0112501 and No. 2021ZD0112502], National Natural

Science Foundation of China [No. 61972174 and No.62172187], the Science and Technology Planning Project of Guangdong Province [No. 2020A0505100018], Guangdong Universities' Innovation Team Project [No. 2021KCXTD015], Guangdong Key Disciplines Project [No. 2021ZDJS138]. Fausto Giunchiglia's work is funded by National Project PRIN 2017 "Delphi".

REFERENCES

[1] Jinheon Baek, Dong Bok Lee, and Sung Ju Hwang. 2020. Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction. In *Advances in Neural Information Processing Systems*.

[2] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*.

[3] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[4] Qi Cai, Yingwei Pan, Ting Yao, Chenggang Yan, and Tao Mei. 2018. Memory matching networks for one-shot image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[5] Jatin Chauhan, Deepak Nathani, and Manohar Kaul. 2020. Few-shot learning on graphs via super-classes based on graph spectral measures. In *International Conference on Learning Representations*.

[6] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2020. Adaptive universal generalized pagerank graph neural network. *International Conference Learning Representations* (2020).

[7] Kaize Ding, Jundong Li, Nitin Agarwal, and Huan Liu. 2020. Inductive anomaly detection on attributed networks. In *International Joint Conference on Artificial Intelligence*.

[8] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep anomaly detection on attributed networks. In *SIAM International Conference on Data Mining*.

[9] Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. 2020. Graph prototypical networks for few-shot learning on attributed networks. In *International Conference on Information & Knowledge Management*.

[10] Bowen Dong, Yuan Yao, Ruobing Xie, Tianyu Gao, Xu Han, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. 2020. Meta-Information Guided Meta-Learning for Few-Shot Relation Classification. In *Proceedings of the International Conference on Computational Linguistics*.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*.

[12] Victor Garcia and Joan Bruna. 2018. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*.

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[14] Renchu Guan, Yonghao Liu, Xiaoyue Feng, and Ximing Li. 2021. Paper-publication Prediction with Graph Neural Networks. In *Proceedings of the ACM International Conference on Information & Knowledge Management*.

[15] David Ha, Andrew M. Dai, and Quoc V. Le. 2017. HyperNetworks. In *International Conference on Learning Representations*.

[16] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*.

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*.

[18] Kexin Huang and Marinka Zitnik. 2020. Graph meta learning via local subgraphs. In *Advances in Neural Information Processing Systems*.

[19] Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[20] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. In *International Conference on Learning Representations*.

[21] Siavash Khodadadeh, Ladislau Bölöni, and Mubarak Shah. 2019. Unsupervised meta-learning for few-shot image classification. In *Advances in Neural Information Processing Systems*.

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

[23] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*.

[24] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*.

[25] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835* (2017).

[26] Meng Liu, Zhengyang Wang, and Shuiwang Ji. 2020. Non-local graph neural networks. *arXiv preprint arXiv:2005.14612* (2020).

[27] Yonghao Liu, Renchu Guan, Fausto Giunchigliaz, Yanchun Liang, and Xiaoyue Feng. 2021. Deep Diffusion Graph Neural Network for Text Classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

[28] Zemin Liu, Yuan Fang, Chenghao Liu, and Steven CH Hoi. 2021. Relative and absolute location embedding for few-shot node classification on graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[29] Zemin Liu, Wentao Zhang, Yuan Fang, Xinming Zhang, and Steven CH Hoi. 2020. Towards locality-aware meta-learning of tail node embeddings on networks. In *Proceedings of the International Conference on Information & Knowledge Management*.

[30] Ning Ma, Jiajun Bu, Jieyu Yang, Zhen Zhang, Chengwei Yao, Zhi Yu, Sheng Zhou, and Xifeng Yan. 2020. Adaptive-Step Graph Meta-Learner for Few-Shot Graph Classification. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.

[31] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming graph neural networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[32] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[33] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[34] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*.

[35] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[37] Guo-Jun Qi, Charu Aggarwal, Qi Tian, Heng Ji, and Thomas Huang. 2011. Exploring context and content links in social media: A latent space method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2011).

[38] Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*.

[39] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*.

[40] Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*.

[41] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[42] Qiuling Suo, Jingyuan Chou, Weida Zhong, and Aidong Zhang. 2020. Tadanet: Task-adaptive network for graph-enriched meta-learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[43] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *International Conference Learning Representations*.

[45] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*.

[46] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J. Lim. 2019. Multimodal Model-Agnostic Meta-Learning via Task-Aware Modulation. In *Advances in Neural Information Processing Systems*.

[47] Ning Wang, Minnan Luo, Kaize Ding, Lingling Zhang, Jundong Li, and Qinghua Zheng. 2020. Graph Few-shot Learning with Attribute Matching. In *Proceedings of ACM International Conference on Information & Knowledge Management*. 1545–1554.

[48] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference*.

[49] Zhihao Wen, Yuan Fang, and Zemin Liu. 2021. Meta-Inductive Node Classification across Graphs. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[50] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International Conference on Machine Learning*.
[51] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. [n. d.]. Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks. In *International Conference on Machine Learning*.
[52] Huaxiu Yao, Chuxu Zhang, Ying Wei, Meng Jiang, Suhang Wang, Junzhou Huang, Nitesh Chawla, and Zhenhui Li. 2020. Graph few-shot learning via knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
[53] Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *International Conference on Machine Learning*.
[54] Zhaoquan Yuan, Jitao Sang, Yan Liu, and Changsheng Xu. 2013. Latent feature learning in social media network. In *Proceedings of the ACM International Conference on Multimedia*.
[55] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. 2020. DeepEMD: Few-Shot Image Classification With Differentiable Earth Mover's Distance and Structured Classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pttern Recognition*.
[56] Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. 2020. Few-shot knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
[57] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. 2019. Meta-gnn: On few-shot node classification in graph meta-learning. In *International Conference on Information and Knowledge Management*.
[58] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*.
[59] Linchao Zhu and Yi Yang. 2018. Compound memory networks for few-shot video classification. In *European Conference on Computer Vision*.