Toward Graph Minimally-Supervised Learning

Kaize Ding
kaize.ding@asu.edu

A Proposal Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Arizona State University
Spring 2022

Abstract

Graph-structured data, ranging from social networks to financial transaction networks, from citation networks to gene regulatory networks, have been widely used for modeling a myriad of real-world systems. As a prevailing model architecture to model graph-structured data, graph neural networks (GNNs) has drawn much attention in both academic and industrial communities in the past decades. Despite their success in different graph learning tasks, existing methods usually rely on learning from "big" data, requiring a large amount of labeled data for model training. However, it is common that real-world graphs are associated with "small" labeled data as data annotation and labeling on graphs is always time and resource-consuming. Therefore, it is imperative to investigate graph machine learning (GML) with minimal human supervision for the low-resource settings where limited or even no labeled data is available.

In this proposal, we will focus on the state-of-the-art techniques of *Graph Minimally-Supervised Learning*, in particular a series of weakly-supervised learning, few-shot learning, and self-supervised learning methods on graph-structured data as well as their real-world applications. The objectives of this tutorial are to: (1) formally categorize the problems in graph minimally-supervised learning and discuss the challenges under different learning scenarios; (2) comprehensively review the existing and recent advances of graph minimally-supervised learning; and (3) elucidate open questions and future research directions. This tutorial introduces major topics within minimally-supervised learning and offers a guide to a new frontier of graph learning. We believe this tutorial is beneficial to researchers and practitioners, allowing them to collaborate on graph learning.

To tackle these challenges, existing research has focused either on the textual content or on tracing user engagements on how fake news propagate. User engagements over information such as news articles, including posting about, commenting on or recommending the news on social media, contain abundant rich information. Since social media data is big, incomplete, noisy, unstructured, with abundant social relations, solely relying on user engagements can be sensitive to noisy user feedback. To alleviate the problem of limited labeled data, it is important to combine contents and this new (but weak) type of information as supervision signals, i.e., *weak social supervision*, to advance fake news detection.

In this proposal, we investigate learning with weak social supervision for understanding disinformation. In particular, we use fake news as an example to study how to effectively derive and exploit the weak supervision for learning with little labeled data. We propose novel frameworks that can learn with weak social supervision to detect fake news more effectively and with explainability, and discuss principles for early detection and cross-domain detection of fake news with weak social supervision.

Contents

1 Introduction

Recent years have witnessed a rapid growth in our ability to generate and gather data from numerous platforms in the online world and various sensors in the physical world. Graphs serve as a common language for modeling a plethora of structured and relational systems, such as social networks, knowledge graphs, and academic graphs, where entities are denoted as nodes while their relations are denoted as edges. More recently, graph learning algorithms, especially those based on graph neural networks (GNNs) [47, 57] have received much research attention due to their significant impacts in addressing real-world problems. To harness the inherent structure among data, significant methodological advances have been made in graph learning, which have produced promising results in applications from diverse domains, ranging from cybersecurity [64] to natural language processing [14].

In general, existing graph learning algorithms focus on the setting where abundant human-annotated examples can be accessed during training. This assumption is often infeasible since collecting such auxiliary knowledge is laborious and requires intensive domain-knowledge, especially when considering the heterogeneity of graph-structured data [59, 15]. As such, it is challenging yet imperative to study graph learning under different low-resource settings with limited or no labeled training data. In particular, three fundamental problems have drawn increasing research attention in the field of graph minimally-supervised learning: (1) *Graph Weakly-Supervised Learning* (Graph WSL), which focuses on learning effective GNNs for a specific down-stream task using either incomplete, or indirect, or inaccurate supervision signals; (2) *Graph Few-Shot Learning* (Graph FSL), whose goal is to handle unseen tasks (from novel label space) when only few labeled instances are available; and (3) *Graph Self-Supervised Learning* (Graph SSL), which aims to either pre-train task-agnostic GNNs or enhance GNNs on specific down-stream tasks without any semantic annotations. To address each of the aforementioned fundamental problems, I conducted a series of research to *push forward the performance boundary of graph machine learning (GML) models with different kinds of low-cost supervision signals*. In the meantime, I also investigated how to effectively adopt minimally-supervised GML algorithms to advance applications in different domains, such as cybersecurity, natural language processing, and recommendation. My Ph.D. research aims to *understand, characterize, and gain actionable insights from massive yet scarcely-labeled data (with a special focus on graph-structured data) by developing effective and efficient AI algorithms, in order to further benefit high-impact real-world applications.*

In this proposal, we attempt to learn with weak social supervision to understand and detect disinformation and fake news more effectively, with explainability, at an early stage, and across domains. The main contributions are:

- Studying novel problems of understanding disinformation such as fake news on social media;
- Providing principled approaches to learn with weak social supervision guided by social theories for multi-faceted social media data;
- Proposing novel frameworks to detect fake news with challenging scenarios including effective, explainable, early, and cross-domain fake news detection; and
- Conducting experiments on real-world datasets to demonstrate the effectiveness of the proposed frameworks.

2 Related Work

2.1 Graph Neural Networks Driven by the momentous success of deep learning, recently, a mass of efforts have been devoted to developing deep neural networks for graph-structured data [7, 6, 12, 54]. As one of the pioneer works, GNN [45] was introduced to learn node representations by propagating neighbors' information via recurrent neural architecture. Based on the graph spectral theory, a series of graph convolutional networks (GCNs) have emerged and demonstrated superior learning performance by designing different graph convolutional layers. Among them, the first prominent research on GCNs called Spectral CNN [5], which extends the convolution operation in the spectral domain for network representation learning. Since then, increasing research advances on graph convolutional networks [26, 9, 23] are presented as its extensions. In addition to spectral graph convolution models, graph neural networks that follow neighborhood aggregation schemes are also extensively investigated. Instead of training individual embeddings for each node, those methods learn a set of *aggregator functions* to aggregate features from a node's local neighborhood. GraphSAGE [22] learns a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. Similarly, Graph Attention Networks (GATs) [52] incorporate trainable attention weights to specify fine-grained weights on neighbors when aggregating neighborhood information of a node. Furthermore, Graph Isomorphism Network (GIN) [58] extends this idea with arbitrary aggregation functions on multi-sets, and is proven to be as theoretically powerful as the Weisfeiler-Lehman (WL) graph isomorphism test. Nevertheless, all the existing GNN models focus on semi-supervised node classification. The inability to handle unseen classes with severely limited samples, is one of the major challenges for the current GNNs.

For real-world graph learning tasks, the amount of gold-labeled samples is usually quite limited due to the expensive labeling cost. To improve the GNN model performance on the node classes with only few labeled nodes, graph few-shot learning [62, 15, 55] and cross-network transfer learning [59, 17] have been proposed to transfer the knowledge from other auxiliary data source(s). Nonetheless, for the problem of *few-shot semi-supervised* node classification, such auxiliary datasets are commonly not allowed to use. As another line of related work, Li et al. [29] combined GCNs and self-training to expand supervision signals, while M3S [49] advances this idea by utilizing the clustering method to eliminate the inaccurate pseudo labels. However, those methods cannot directly address the oversmoothing issue and may suffer from inaccurate pseudo labels. By conducting meta-learning on top of a decoupled design, our approach Meta-PN achieves superior performance on few-shot semi-supervised node classification.

2.2 Few-shot Learning Few-shot learning (FSL) aims to solve new tasks with a limited number of examples, based on the knowledge obtained from previous experiences. Generally, existing FSL models fall into two broad categories: (1) *optimization-based approaches*, which focus on learning the optimization of model parameters given the gradients on few-shot examples [42, 19, 31, 37]. One example is the LSTM-based meta-learner [42], which aims to learn efficient parameter updating rules for training a neural classifier. MAML [19] learns the parameter initialization that is suitable for different FSL tasks and is compatible with any model trained with gradient descent. Meta-SGD [31] goes further in meta-learning by arguing to learn the weights initialization, gradient update direction and learning rate within a single step. SNAIL [37] is another model which combines temporal convolution and soft attention to learn an optimal learning strategy. However, this line of work usually suffers from the computational cost of fine-tuning. (2) *metric-based approaches*, which try to learn generalizable

matching metrics between query and support set across different tasks [53, 48, 43, 50, 33]. For instance, Matching Networks [53] learn a weighted nearest-neighbor classifier with attention networks. Prototypical Network [48] computes the prototype of each class by taking the mean vector of support examples and classifies query instances by calculating their Euclidean distances. An extension of Prototypical Networks proposed by Ren et al. [43] considers both labeled and unlabeled data for few-shot learning. Relation Network [50] trains an auxiliary network to learn a non-linear metric between each query and the support set. Recently, few-shot learning on graphs has received increasing research attention [62, 4]. However, those methods treat support examples equally, rendering the model unstable to noises or outliers [10]. In this paper, we learn a robust and powerful few-shot learning model by considering the individual importance of labeled support examples.

2.2.1 Weakly-Supervised Learning

2.2.2 Self-Supervised Learning

3 Graph Few-Shot Learning

Following the commonly used notations, in this paper, we use calligraphic fonts, bold lowercase letters, and bold uppercase letters to denote sets (e.g., $\mathcal{G}$), vectors(e.g., $\mathbf{x}$), and matrices (e.g., $\mathbf{X}$), respectively. The i^{th} row of a matrix $\mathbf{X}$ is denoted by $\mathbf{x}_i$, and the transpose of a matrix $\mathbf{X}$ is represented as $\mathbf{X}^{\mathrm{T}}$. We summarize the main notations used throughout the paper in Table **??**. For the other special notations, we will illustrate them in the corresponding sections.

Formally, an attributed network can be represented as $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ denotes 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 associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^{1\times d}$ and $\mathbf{X} = [\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_n] \in \mathbb{R}^{n\times d}$ denotes all the node features. Thus, more generally, the attributed network can be represented as $G = (\mathbf{A}, \mathbf{X})$, where $\mathbf{A} = \{0, 1\}^{n\times n}$ is an adjacency matrix representing the network structure. Specifically, $\mathbf{A}_{i,j} = 1$ indicates that there is an edge between node v_i and node v_j; otherwise, $\mathbf{A}_{i,j} = 0$. The studied problem can be formulated as follows:

As existing FSL models are not tailored for graph-structured data, it is infeasible to apply them to solve the studied problem directly. In this section, we present the details about the proposed Graph Prototypical Networks (GPN) for few-shot node classification on attributed networks. Specifically, our framework is designed and built to address three challenging research questions: (1) How to perform *meta-learning* on attributed networks (*non-i.i.d.* data) for extracting the meta-knowledge? (2) How to learn expressive node representations from the input attributed network by considering both the node attributes and topological structure? and (3) How to identify the informativeness of each labeled node for learning robust and discriminative class representations?

An overview of the proposed Graph Prototypical Networks (GPN) is provided in Figure 1. In Section 4.1, we introduce the backbone training mechanism of the proposed model. In Section 4.2 and 4.3, we introduce how we design the two essential modules in GPN. Then we discuss how to perform few-shot node classification using the proposed framework in Section 4.4. Last, we present the complexity analysis in Section 4.5.

3.1 Graph Prototypical Networks

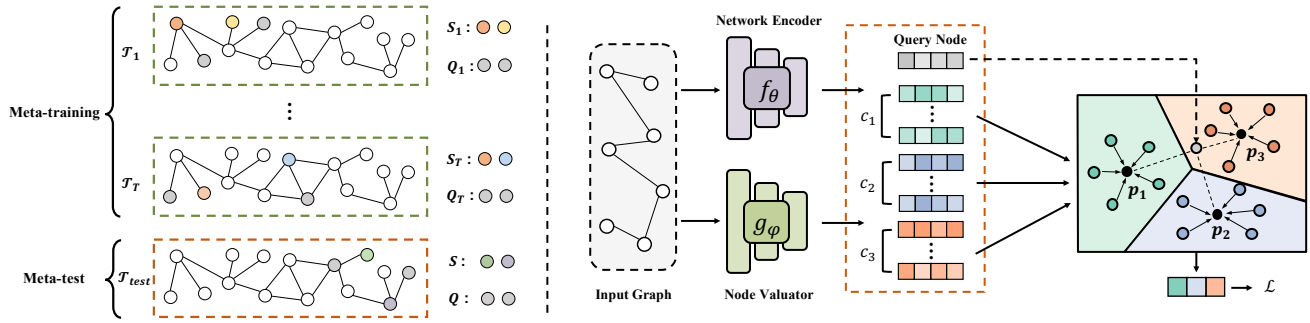


Figure 1: (Left) Episodic training on attributed networks. In each episode, we create a semi-supervised few-shot node classification task by random sampling; (Right) The architecture of the proposed framework Graph Prototypical Networks (GPN).

3.1.1 Episodic Training on Attributed Networks Our approach is a meta-learning framework which follows the prevailing episodic training paradigm [53]. Specifically, GPN learns over diverse *meta-training* tasks in a large number of episodes rather than only on the target *meta-test* task. The key idea of episodic training is to mimic the real test environment by sampling nodes from C_{train}. The consistency between training and test environment alleviates the distribution gap and improves model generalization capability. Specifically, in each episode, we construct a N-way K-shot meta-training task:

$$\begin{aligned} \mathcal{S}_t &= \{(v_1, y_1), (v_2, y_2), ..., (v_{N\times K}, y_{N\times K})\}, \\ \mathcal{Q}_t &= \{(v_1^*, y_1^*), (v_2^*, y_2^*), ..., (v_{N\times M}^*, y_{N\times M}^*)\}, \\ \mathcal{T}_t &= \{\mathcal{S}_t, \mathcal{Q}_t\}, \end{aligned} \tag{3.1}$$

where both the support set $\mathcal{S}_t$ and query set $\mathcal{Q}_t$ of the meta-training task $\mathcal{T}_t$ are sampled from C_{train}. The support set $\mathcal{S}_t$ contains K nodes from each class, while the query set $\mathcal{Q}_t$ includes M query nodes sampled from the remainder of each of the N classes.

The whole training process is based on a set of T meta-training tasks $\mathcal{T}_{train} = \{\mathcal{T}_t\}_{t=1}^{T}$. The model is trained to minimize the loss of its predictions for the query set $\mathcal{Q}_t$ in each meta-training task $\mathcal{T}_t$, and goes episode by episode until convergence. In this way, the model gradually collects meta-knowledge across those meta-training tasks and then can be naturally generalized to the meta-test task $\mathcal{T}_{test} = \{\mathcal{S}, \mathcal{Q}\}$ with unseen classes C_{test}.

Different from conventional episodic training that constructs a pool of supervised *meta-training* tasks [20], in each episode, we sample N-way K-shot labeled nodes and mask the rest as unlabeled nodes. In this way, we can create a semi-supervised *meta-training* task with the partially labeled attributed network. By considering both labeled and unlabeled data and their dependencies, we are able to learn more expressive node representations for few-shot node classification during the *meta-learning* process.

3.1.2 Network Representation Learning In order to learn expressive node representations from an attributed network, we develop a *network encoder* to capture the data heterogeneity. Specifically, the *network encoder* possesses a GNN backbone, which converts each node to a low-dimensional latent representation. In general, GNNs follow the neighborhood aggregation scheme, and compute the node representations by recursively aggregating and compressing node features from local neighborhoods.

Briefly, a GNN layer can be defined as:

$$\begin{aligned}\mathbf{h}_i^l &= \textsc{Combine}^l\Big(\mathbf{h}_i^{l-1}, \mathbf{h}_{\mathcal{N}_i}^l\Big),\\ \mathbf{h}_{\mathcal{N}_i}^l &= \textsc{Aggregate}^l\Big(\{\mathbf{h}_j^{l-1} | \forall j \in \mathcal{N}_i \cup v_i\}\Big),\end{aligned} \tag{3.2}$$

where $\mathbf{h}_i^l$ is the node representation of node i at layer l and $\mathcal{N}_i$ is the set of neighboring nodes of v_i. Combine and Aggregate are two key functions of GNNs and have a series of possible implementations [26, 22, 52].

By stacking multiple GNN layers in the *network encoder*, the learned node representations are able to capture the long-range node dependencies in the network:

$$\begin{aligned}\mathbf{H}^1 &= \text{GNN}^1(\mathbf{A}, \mathbf{X}),\\ &\dots\\ \mathbf{Z} &= \text{GNN}^L(\mathbf{A}, \mathbf{H}^{L-1}),\end{aligned} \tag{3.3}$$

where $\mathbf{Z}$ is the learned node representations from the *network encoder*. For simplicity, we will use $f_{\boldsymbol{\theta}}(\cdot)$ to denote the *network encoder* with L GNN layers.

Prototype Computation. With the learned node representations from the *network encoder*, next, we aim to compute the representation of each class with the labeled nodes from the support set. We follow the idea of Prototypical Networks [48], which encourages nodes of each class cluster around a specific prototype representation. Formally, the class prototypes can be computed by:

$$\mathbf{p}_c = \textsc{Proto}\Big(\{\mathbf{z}_i | \forall i \in \mathcal{S}_c\}\Big), \tag{3.4}$$

where $\mathcal{S}_c$ denotes the set of labeled examples from class c and Proto is the prototype computation function. For instance, in the vanilla Prototypical Networks [48], the prototype of each class is computed by taking the average of all embedded nodes belonging to that class:

$$\mathbf{p}_c = \frac{1}{|\mathcal{S}_c|} \sum_{i \in \mathcal{S}_c} \mathbf{z}_i. \tag{3.5}$$

3.1.3 Node Importance Valuation Despite its simpleness, directly taking the mean vectors of the embedded support instances as prototypes may not provide promising results for our problem. It not only neglects the fact that each node has a different significance in a network, but also makes the FSL model highly noise-sensitive since labeled data is severely limited [60]. Therefore, refining those class prototypes becomes especially essential for building a robust and effective FSL model.

To identify the informativeness of each labeled node, we adopt a view that the importance of a node is highly correlated with its neighbors' importance [40]. Accordingly, we design a GNN-based *node valuator* $g_\phi(\cdot)$ (as shown in Figure 3) to estimate node importance scores through a *score aggregation layer*, which can be defined as follows:

$$s_i^l = \sum_{j \in \mathcal{N}_i \cup v_i} \alpha_{ij}^l s_j^{l-1}, \tag{3.6}$$

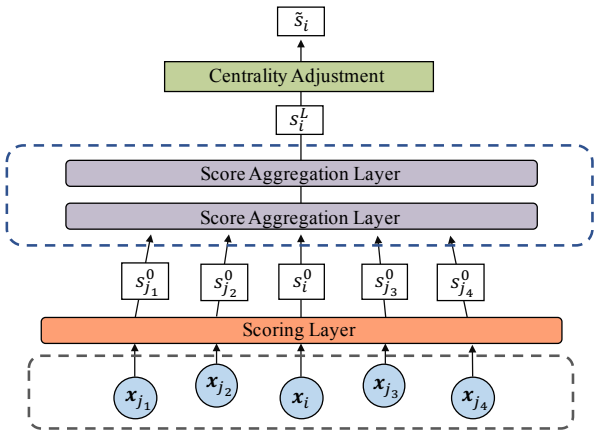


Figure 2: Architecture of the node valuator.

where s_i^l is the importance score of node v_i in the l-th layer ($l = 1, \ldots, L$). α_{ij}^l is the attention weight between nodes v_i and v_j, we compute it via a shared attention mechanism:

$$\alpha_{ij}^l = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^{\mathrm{T}}[s_i^{l-1} \| s_j^{l-1}]))}{\sum_{k \in \mathcal{N}_i \cup v_i} \exp(\text{LeakyReLU}(\mathbf{a}^{\mathrm{T}}[s_j^{l-1} \| s_k^{l-1}]))}, \tag{3.7}$$

where $\|$ is a concatenation operator and $\mathbf{a}$ is a weight vector.

To compute the initial importance score s_i^0, we employ a *scoring layer* to compress the node features. Our *scoring layer* is a feed-forward layer with tanh non-linearity. Specifically, the initial score of node v_i is computed by:

$$s_i^0 = \tanh(\mathbf{w}_s^{\mathrm{T}} \mathbf{x}_i + b_s) \tag{3.8}$$

where $\mathbf{w}_s \in \mathbb{R}^d$ is a learnable weight vector and $b_s \in \mathbb{R}^1$ is the bias.

Centrality Adjustment. As suggested in previous research on node importance estimation [39, 40] , the importance of a node positively correlates with its centrality in the graph. Given that the in-degree $\deg(i)$ of node v_i is a common proxy for its centrality and popularity, we define the initial centrality $C(i)$ of node v_i as:

$$C(i) = \log(\deg(i) + \epsilon), \tag{3.9}$$

where ϵ is a small constant. To compute the final importance score, we apply centrality adjustment to the estimated score s_i^L from the last layer, and apply a sigmoid non-linearity as follows:

$$\tilde{s}_i = \text{sigmoid}(C(i) \cdot s_i^L). \tag{3.10}$$

3.1.4 Few-shot Node Classification After we compute the importance score of each support node, we first normalize those scores using the softmax function:

$$\beta_i = \frac{\exp(\tilde{s}_i)}{\sum_{k \in \mathcal{S}_c} \exp(\tilde{s}_k)}, \tag{3.11}$$

where β_i represents the normalized weight of each support node v_i, then the refined prototypes can be directly computed by:

$$\mathbf{p}_c = \sum_{i \in \mathcal{S}_c} \beta_i \mathbf{z}_i. \tag{3.12}$$

Table 1: Statistics of the evaluation datasets.

Datasets	# nodes	# edges	# attributes	# Train/Valid/Test
Amazon-Clothing	24,919	91,680	9,034	40/17/20
Amazon-Electronics	42,318	43,556	8,669	90/37/40
DBLP	40,672	288,270	7,202	80/27/30
Reddit	232,965	11,606,919	602	16/10/15

As such, our model can adjust the cluster locations to better represent the examples in both the support and unlabeled sets. These learned prototypes define a predictor for the class label of a query node v_i^*, which assigns a probability over each class c based on the distances between the query node v_i^* and each prototype:

$$p(c|v_i^*) = \frac{\exp(-d(\mathbf{z}_i^*, \mathbf{p}_c))}{\sum_{c'} \exp(-d(\mathbf{z}_i^*, \mathbf{p}_{c'}))}, \tag{3.13}$$

where $d(\cdot)$ is a distance metric function. Commonly, squared Euclidean distance is a simple and effective choice [48].

Under the episodic training framework, the objective of each meta-training task is to minimize the classification loss between the predictions of the query set and the ground-truth. Specifically, the training loss can be defined as the average negative log-likelihood probability of assigning correct class labels:

$$\mathcal{L} = -\frac{1}{N \times M} \sum_{i=1}^{N \times M} \log p(y_i^*|v_i^*). \tag{3.14}$$

By minimizing the above loss function, GPN is able to learn a generic classifier for a specific meta-training task. Training episodes are formed by randomly selecting a subset of classes from the auxiliary class set C_{train}, then choosing a subset of nodes within each class to act as the support set and a subset of the remainder to serve as query set. After training on a considerable number of meta-training tasks, its generalization performance will be measured on the test episodes, which contain nodes sampled from C_{test} instead of C_{train}. For each test episode, we use the predictor produced by our GPN for the provided support set $\mathcal{S}$ to classify each query node in $\mathcal{Q}$ into the most likely class: $\hat{y}_i^* = \text{argmax}_c p(c|v_i^*)$.

3.2 Performance Evaluation

3.2.1 Evaluation Settings

Evaluation Datasets. Due to the fact that few-shot node classification on graph-structured data remains an under-studied problem, it is worth mentioning that the existing benchmark datasets (e.g., Cora, Pubmed) for conventional node classification problem are not suitable for evaluating FSL models. The main reason is that FSL models usually need to be tested on many different classification tasks, while those datasets only contain limited node classes. To extensively evaluate the model performance on few-shot node classification, in our experiments, we adopt four public datasets with plenty of node classes and their statistics of can be found in Table 1. **Amazon-Clothing** and **Amazon-Electronics** [35] are two product networks built with the products in "Clothing, Shoes and Jewelry" and "Electronics" on Amazon respectively. In these networks, each product is considered as a node and its description is used to construct the node attributes. We use the substitutable and complementary

relationship to create links between products. The class label is defined as the low-level product category. **DBLP** [51] is a citation network where each node represents a paper, and the links are the citation relations among different papers. The paper abstracts are used to construct node attributes. The class label of a node is defined as the paper venue. **Reddit** [22] is a post-to-post graph constructed with data sampled from Reddit, which is used to evaluate the performance of our model on large-scale attributed networks. In this large-scale attributed network, posts are represented by nodes and two posts are connected if they are commented by the same user. Each post is labeled with it a community ID.

Compared Methods. In the experiments, we compare the proposed model GPN with related baseline methods: **DeepWalk** [41] learns node embeddings from a stream of truncated vanilla random walks on the input graph, and **node2vec** [21] extends it with biased random walks to explore diverse neighborhoods. **GCN** [26] learns latent node representations based on the first-order approximation of spectral graph convolutions. **SGC** [56] eliminates the non-linearity between GCN layers and folding the convolution functions into a linear transformation. Prototypical Network (**PN**) [48] is one of the widely used few-shot learning methods for image classification. **MAML** [19] is an optimization-based meta-learning method, which tries to learn a better model initialization from a series of meta-training tasks. By using a GNN base model, **Meta-GNN** [62] extends MAML to graph data.

3.2.2 General Comparisons For each dataset, we evaluate the performance of all the algorithms on four few-shot node classification tasks, i.e., 5-way-3-shot, 5-way-5-shot, 10-way-3-shot, and 10-way-5-shot. We set the query size as same as the support size in our experiments. We adopt two widely used metrics Accuracy (ACC) and Micro-F1 (F1) to evaluate performance. Each model is evaluated on 50 *meta-test* tasks and each *meta-test* task is randomly sampled from test node classes. We repeat the process 10 times and the averaged results are presented in Table 3. Higher values are better for all metrics. From the comprehensive views, we make the following observations:

- A general observation is that our approach GPN achieves the best performance on all the few-shot tasks. For example, on the Amazon-Clothing dataset, GPN outperforms the best performing baseline Meta-GNN by 5.9% (ACC) under the 10-way-3-shot task. The improvements are even more substantial on the larger dataset Reddit. This result verifies that GPN is a powerful and reliable model to tackle the problem of few-shot node classification on attributed networks.

- Overall, DeepWalk and node2vec largely fall behind other methods on few-shot node classification tasks. Those random walk-based methods need to train a supervised classifier (e.g., Logistic Regression) with learned node representations, which typically rely on a large number of labeled data for good performance. Similarly, GNN-based methods are unable to obtain competitive results on the few-shot node classification problem. Conventional GNN models are developed for semi-supervised node classification, and could be easily overfitted with only a small number of labeled instances.

- Despite the success of MAML and PN on few-shot image classification, however, both of them perform poorly on our tasks. The main reason is that those methods cannot capture the dependency between nodes for learning expressive node representations, rendering unsatisfactory performance on few-shot node classification tasks.

- By integrating the idea of meta-learning into graph neural networks, Meta-GNN is able to achieve considerable improvements over other baseline methods on few-shot node classification in most

Table 2: Averaged few-shot node classification results on four datasets w.r.t ACC and F1 (%).

Methods	Amazon-Clothing								Amazon-Electronics							
	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	36.7	36.3	46.5	46.6	21.3	19.1	35.3	32.9	23.5	22.2	26.1	25.7	14.7	12.9	16.0	14.7
node2vec	36.2	35.8	41.9	40.7	17.5	15.1	32.6	30.2	25.5	23.7	27.1	24.3	15.1	13.1	17.7	15.5
GCN	54.3	51.4	59.3	56.6	41.3	37.5	44.8	40.3	53.8	49.8	59.6	55.3	42.3	38.4	47.4	48.3
SGC	56.8	55.2	62.2	61.5	43.1	41.6	46.3	44.7	54.6	53.4	60.8	59.4	43.2	41.5	50.0	47.6
PN	53.7	53.6	63.5	63.7	41.5	41.9	44.8	46.2	53.5	55.6	59.7	61.5	39.9	40.0	45.0	44.8
MAML	55.2	54.5	66.1	67.8	45.6	43.3	46.8	45.6	53.3	52.1	59.0	58.3	37.4	36.1	43.4	41.3
Meta-GNN	74.1	73.6	77.3	77.5	61.4	59.7	64.2	62.9	63.2	61.5	67.9	66.8	58.2	55.8	60.8	60.1
GPN	**75.4**	**74.7**	**78.6**	**79.0**	**65.0**	**66.1**	**67.7**	**68.9**	**64.6**	**62.8**	**70.9**	**70.6**	**60.3**	**60.7**	**62.4**	**63.7**

Methods	DBLP								Reddit							
	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	44.7	43.1	62.4	60.4	33.8	30.8	45.1	43.0	26.7	26.1	30.1	29.7	17.6	17.1	18.8	18.6
node2vec	40.7	38.5	58.6	57.2	31.5	27.8	41.2	39.6	27.1	25.6	31.2	29.8	19.8	18.6	23.4	22.6
GCN	59.6	54.9	68.3	66.0	43.9	39.0	51.2	47.6	38.8	38.1	45.5	44.1	29.0	27.0	35.7	32.4
SGC	57.3	54.7	65.0	62.1	40.2	36.8	50.3	46.4	44.4	42.1	46.8	42.5	29.7	26.8	31.6	27.7
PN	37.2	36.7	43.4	44.3	26.2	26.0	32.6	32.8	34.6	33.3	37.6	36.4	19.8	18.0	23.3	21.4
MAML	39.7	39.7	45.5	43.7	30.8	25.3	34.7	31.2	29.1	26.8	31.1	29.7	15.2	12.2	17.9	15.6
Meta-GNN	70.9	70.3	78.2	78.2	60.7	60.4	68.1	67.2	60.8	58.3	62.7	61.2	44.9	42.1	51.5	47.1
GPN	**74.5**	**73.9**	**80.1**	**79.8**	**62.6**	**62.6**	**69.0**	**69.4**	**65.5**	**66.2**	**68.4**	**69.0**	**53.4**	**55.8**	**57.7**	**59.2**

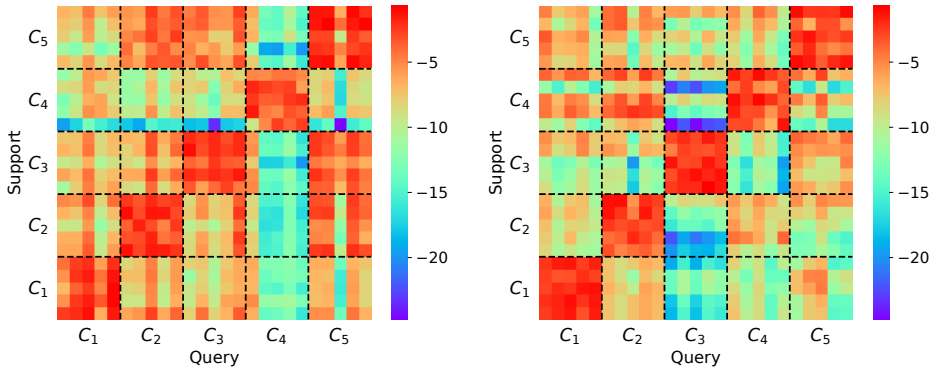
cases. However, it is worth noting that its performance suffers a catastrophic decline on the Reddit dataset. One reasonable explanation is that optimization-based FSL approaches require extensive fine-tuning efforts for the target task, especially on those large-scale datasets.

3.2.3 Case Study Figure 3 shows the similarity matrix learned by the best performing baseline Meta-GNN and our approach on the DBLP dataset, with the same network encoder in a 5-way 5-shot task. Here we use the negative Euclidean distance as the similarity metric. Specifically, each cell consists of 5×5 grids illustrating the divergence between two classes, as well as the intra-class similarities. To better visualize the results, for GPN, we use the weighted embedding of each support node instead of computing the class prototype. From the figure, we can observe that GPN can better capture the similarities between the support nodes and query nodes from a same class, which validates the robustness and effectiveness of our approach.

4 Graph Weakly-Supervised Learning

Despite their promising results, existing GNNs developed for *semi-supervised* node classification predominantly assume that the provided gold-labeled nodes are relatively abundant. This assumption is often impractical as data labeling requires intensive domain knowledge, especially when considering the heterogeneity of graph-structured data [59, 15]. When only few labeled nodes per class are available, how to improve the expressive power of Graph ML models for tackling the *few-shot semi-supervised* node classification problem remains understudied and meanwhile requires urgent research efforts.

However, it is a non-trivial and challenging task mainly because of two reasons: (i) ***oversmoothing***



(a) **Meta-GNN** (b) **GPN**

Figure 3: Similarity matrix on DBLP dataset (5-way 5-shot).

and overfitting. In general, most of the existing GNNs are designed with shallow architecture with restricted receptive fields, thereby restricting the efficient propagation of label information [29]. In order to propagate the label signals more broadly, larger receptive fields of GNNs, i.e., the number of layers, are particularly desirable [28]. Due to the entanglement of representation transformation and propagation in each layer, GNNs will face the oversmoothing issue when increasing the model depth [34], which in turn renders the learned node representations inseparable. In the meantime, when training with few labeled nodes, an over-parametric deep GNN model tends to overfit and goes timber easily; (ii) ***no auxiliary knowledge.*** Though previous works proposed for graph few-shot learning [15] or cross-network transfer learning [59] also focus on related low-resource scenarios, their key enabler lies in transferring knowledge from either label-rich node classes or other similar networks. Nonetheless, such auxiliary knowledge is commonly not accessible, making those methods practically infeasible to be applied to few-shot semi-supervised learning. As suggested by previous research, pseudo-labeling [29, 49, 16] is commonly beneficial to solve semi-supervised learning, whereas inaccurate pseudo labels may instead lead to abysmal failure. Hence, how to infer accurate pseudo labels on unlabeled nodes plays a pivotal role to solve the studied research problem.

4.1 Meta Propagation Networks

4.1.1 Architecture Overview For solving the problem of few-shot semi-supervised node classification, we propose a new framework Meta Label Propagation (Meta-PN), which is built with two simple neural networks, i.e., *adaptive label propagator* and *feature-label transformer*. By decoupling the propagation and transformation steps with two independent networks, such a design inherently allows large receptive fields without suffering performance deterioration. Upon our proposed meta-learning algorithm, the meta learner – *adaptive label propagator* learns to adjust its propagation strategy for inferring accurate pseudo labels on unlabeled nodes, by using the feedback from the target model. Meanwhile, the target model – *feature-label transformer* assimilates both the structure and feature knowledge from pseudo-labeled nodes, therefore addressing the challenges behind few-shot semi-supervised learning. Specifically, we introduce the architecture details as follows:

Adaptive Label Propagator (Meta Learner). In order to enable broader propagation of label signals, we propose to adopt the idea of label propagation (LP) [63] to encode informative local and

global structural information. Similar to the message-passing scheme adopted by many GNNs, label propagation follows the principle of Homophily [36] that indicates two connected nodes tend to be similar (share same labels). Specifically, the objective of LP is to find a prediction matrix $\hat{\mathbf{Y}} \in \mathbb{R}^{n \times c}$ that agrees with the label matrix $\mathbf{Y}$ while being smooth on the graph such that nearby vertices have similar soft labels [61]. Generally, the solution can be approximated via the iteration as follows:

$$\hat{\mathbf{Y}} = \mathbf{Y}^{(K)}, \mathbf{Y}^{(k+1)} = \mathbf{T}\mathbf{Y}^{(k)}, \tag{4.15}$$

where $\mathbf{Y}^{(0)} = \mathbf{Y}$ and K denotes the number of power iteration (propagation) steps. The transition matrix is denoted by $\mathbf{T}$, which can be set as any form of normalized adjacency matrix (e.g., $\tilde{\mathbf{A}}_{sym}$). After K iterations of label propagation, the predicted soft label matrix $\hat{\mathbf{Y}}$ can capture the prior knowledge of neighborhood label distribution up to K hops away.

In practice, various propagation schemes can be adopted for LP, such as the Personalized PageRank [28] where $\mathbf{Y}^{(k+1)} = (1-\alpha)\mathbf{T}\mathbf{Y}^{(k)} + \alpha\mathbf{Y}^{(0)}$. With appropriate teleport probability α, the smoothed labels can avoid losing the focus on local neighborhood even using infinitely many propagation steps [28]. However, most of the existing LP algorithms cannot adaptively balance the label information from different neighborhoods for each node, which largely restricts the model expressive power when learning with complex real-world graphs.

To counter this issue, we build an *adaptive label propagator* $g_{\boldsymbol{\phi}}(\cdot)$ parameterized with $\boldsymbol{\phi}$, which is able to adjust the contribution of different propagation steps for computing the smoothed label vector of one node. Specifically, the propagation strategy can be formulated as:

$$\hat{\mathbf{Y}}_{i,:} = \sum_{k=0}^{K} \gamma_{ik} \mathbf{Y}_{i,:}^{(k)}, \mathbf{Y}^{(k+1)} = \mathbf{T}\mathbf{Y}^{(k)}, \tag{4.16}$$

where γ_{ik} denotes the influence from k-hop neighborhood for node v_i and can be computed by the attention mechanism:

$$\gamma_{ik} = \frac{\exp\left(\mathbf{a}^{\mathrm{T}}\mathrm{ReLU}\left(\mathbf{W}\mathbf{Y}_{i,:}^{(k)}\right)\right)}{\sum_{k'=0}^{K}\exp\left(\mathbf{a}^{\mathrm{T}}\mathrm{ReLU}\left(\mathbf{W}\mathbf{Y}_{i,:}^{(k')}\right)\right)}, \tag{4.17}$$

where $\mathbf{a} \in \mathbb{R}^c$ is the attention vector and $\mathbf{W} \in \mathbb{R}^{c \times c}$ is a weight matrix. By setting the attention vector and weight matrix as learnable parameters, the *adaptive label propagator* acquire the capability of adjusting its propagation strategy for each node and the final smoothed labels can capture rich structure information of the input graph.

Feature-label Transformer (Target Model). After encoding the structure knowledge into the smoothed label matrix $\hat{\mathbf{Y}}$, we then build a *feature-label transformer* $f_{\boldsymbol{\theta}}(\cdot)$ that transforms node features to node label, in order to further capture feature-based graph information. For each node v_i, the *feature-label transformer* parameterized with $\boldsymbol{\theta}$ takes the node feature vector $\mathbf{X}_{i,:}$ as input and predicts its node label $\mathbf{P}_{i,:}$ by:

$$\mathbf{P}_{i,:} = f_{\boldsymbol{\theta}}(\mathbf{X}_{i,:}), \tag{4.18}$$

where $f_{\boldsymbol{\theta}}(\cdot)$ is a multi-layer perceptron (MLP) followed by a softmax function.

In order to learn the target model , i.e., *feature-label transformer*, we take the soft pseudo labels computed by the *adaptive label propagator* as "ground-truth". Ideally, if the generated pseudo labels are of high quality, they can be used to augment the insufficient labeled nodes to avoid overfitting

and improve the model generalization ability [29]. In the meantime, high-quality pseudo-labeled data not only encodes the feature patterns of unlabeled nodes, but also carries informative local and global structure knowledge, which enables the target model to leverage larger receptive fields without suffering from performance degradation. As a result, the *feature-label transformer* can achieve excellent performance on the problem of few-shot semi-supervised node classification.

4.1.2 Learning to Propagate. One key challenge of our approach lies in how to learn a better label propagation strategy for generating pseudo labels on unlabeled nodes. If the pseudo labels are inaccurate, the target model may easily overfit to mislabeled nodes and encounter severe performance degradation [44]. This issue is also known as the problem of confirmation bias in pseudo-labeling [1]. While inferring accurate pseudo labels by recursively selecting a subset of samples, re-training the prediction model will be too expensive and unstable. Hence, without linking the two networks in a principled way, it is almost infeasible to enforce the *adaptive label propagator* to efficiently infer meaningful label propagation strategy for improving the performance of the *feature-label transformer*.

In this work, we propose to tackle this problem through a unified meta-learning algorithm, allowing the model to infer accurate pseudo labels for unlabeled nodes and learn a better target model. In a sense, if the generated pseudo labels are of high quality, their data utility should align with the gold-labeled nodes. Accordingly, we can derive the following meta-learning objective: *optimal pseudo labels generated by meta-learner should maximize target model's performance (minimize the classification loss) on the gold-labeled training nodes*. For each *meta label propagation* task, the goal is to generate pseudo labels for a batch of unlabeled nodes using the feedback of the target model (i.e., *feature-label transformer*). By optimizing the *adaptive label propagator* on a meta-level, it can adjust the label propagation strategy to generate informative pseudo-labeled data.

4.1.3 Model Learning via Bi-level Optimization The above meta-learning objective implies a bi-level optimization problem with $\boldsymbol{\phi}$ as the outer-loop parameters and $\boldsymbol{\theta}$ as the inner-loop parameters. This problem shares the same formulation with many meta-learning algorithms that have been proposed for solving different learning tasks such as few-shot learning [19], hyper-parameter optimization [3], and neural architecture search [32]. Specifically, let $\mathcal{L}$ denote the cross-entropy loss for node classification, and this bi-level optimization problem can be formulated as:

$$\begin{aligned} &\text{Outer loop: } \boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{v_i \in \mathcal{V}^L}[\mathcal{L}(f_{\theta^*(\boldsymbol{\phi})}(\mathbf{X}_{i,:}), \mathbf{Y}_{i,:})], \\ &\text{Inner loop: } \boldsymbol{\theta}^*(\boldsymbol{\phi}) = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{v_i \in \mathcal{V}^U}[\mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{X}_{i,:}), g_{\boldsymbol{\phi}}(\mathbf{Y}, \mathbf{A})_{i,:})]. \end{aligned} \tag{4.19}$$

The optimal solution of this bi-level optimization problem can potentially train a highly discriminative *feature-label transformer* with abundant pseudo-labeled data and only a small set of gold-labeled data. However, deriving exact solutions for this bi-level problem is indeed analytically intractable and computationally expensive, owing to the fact that it requires solving for the optimal $\boldsymbol{\theta}^*(\boldsymbol{\phi})$ whenever $\boldsymbol{\phi}$ gets updated. To approximate the optimal solution $\boldsymbol{\theta}^*(\boldsymbol{\phi})$, we propose to take one step of gradient descent update for $\boldsymbol{\theta}$, without solving the inner-loop optimization completely by training until convergence. This way allows the optimization algorithm to alternatively update the parameters of *feature-label transformer* in the inner loop and the parameters of *adaptive label propagator* in the outer loop:

Target Model (Inner-loop) Update. Given a batch of unlabeled nodes from $\mathcal{V}^U$, we update the target model parameters $\boldsymbol{\theta}$ by taking their pseudo labels computed by the *adaptive label propagator* as

ground-truth. For simplicity, we use $J_{\text{pseudo}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ to denote the inner-loop loss computed on a batch of pseudo-labeled nodes. Assuming that parameter $\boldsymbol{\theta}$ is updated using the computed gradient descent on $J_{\text{pseudo}}(\boldsymbol{\theta}, \boldsymbol{\phi})$, with a learning rate $\eta_{\boldsymbol{\theta}}$, then we have:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} J_{\text{pseudo}}(\boldsymbol{\theta}, \boldsymbol{\phi}). \tag{4.20}$$

Meta Learner (Outer-loop) Update. Note that the dependency between $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ allows us to compute the meta-level (outer-loop) loss using the gold-labeled nodes from $\mathcal{V}^L$. We denote this loss by $J_{\text{gold}}(\boldsymbol{\theta}'(\boldsymbol{\phi}))$ for the purpose of simplicity, and back-propagate this loss to compute the gradient for the *feature-label transformer*. Having the gradient, we can update on the backward parameters $\boldsymbol{\phi}$ with learning rate $\eta_{\boldsymbol{\phi}}$:

$$\boldsymbol{\phi}' = \boldsymbol{\phi} - \eta_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\phi}} J_{\text{gold}}(\boldsymbol{\theta}'(\boldsymbol{\phi})). \tag{4.21}$$

To further compute the gradient of $\boldsymbol{\phi}$, we apply chain rule to differentiate $J_{\text{gold}}(\boldsymbol{\theta}'(\boldsymbol{\phi}))$ with respect to $\boldsymbol{\phi}$ via $\boldsymbol{\theta}'$, where $\boldsymbol{\theta}'(\boldsymbol{\phi}) = \boldsymbol{\theta} - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} J_{\text{pseudo}}(\boldsymbol{\theta}, \boldsymbol{\phi})$. The full derivation is delegated to the Appendix **??**. Here, we directly present the final result:

$$\nabla_{\boldsymbol{\phi}} J_{\text{gold}}(\boldsymbol{\theta}'(\boldsymbol{\phi})) \approx -\frac{\eta_{\boldsymbol{\phi}}}{2\epsilon} [\nabla_{\boldsymbol{\phi}} J_{\text{pseudo}}(\boldsymbol{\theta}^+, \boldsymbol{\phi}) - \nabla_{\boldsymbol{\phi}} J_{\text{pseudo}}(\boldsymbol{\theta}^-, \boldsymbol{\phi})], \tag{4.22}$$

where $\boldsymbol{\theta}^{\pm} = \boldsymbol{\theta} \pm \epsilon \nabla_{\boldsymbol{\theta}'} J_{\text{gold}}(\boldsymbol{\theta}'(\boldsymbol{\phi}))$, and ϵ is a small scalar for finite difference approximation.

By alternating the update rules in Eq. (4.20) and Eq. (4.21), we are able to progressively learn the two modules. The complete meta-learning algorithm is shown in Algorithm **??**. Finally, as the *feature-label transformer* only learns from unlabeled data with pseudo labels generated by the *adaptive label propagator*, we can further fine-tune the *feature-label transformer* on labeled data to improve its accuracy. After the model converges, we use the *feature-label transformer* to make final predictions on unlabeled nodes.

4.2 Performance Evaluation

4.2.1 Evaluation Settings

Evaluation Datasets. We conduct experiments on five graph benchmark datasets for semi-supervised node classification to demonstrate the effectiveness of the proposed Meta-PN. The detailed statistics of the datasets are summarized in Table **??**. Specifically, **Cora-ML**, **CiteSeer** [46] and **PubMed** [38] are the three most widely used citation networks. **MS-CS** is a co-authorship network based on the Microsoft Academic Graph [47]. For data splitting, we follow the previous work [28] and split each dataset into training set (i.e., K nodes per class for K-shot task), validation set and test set. In addition, to further evaluate the performance of different methods on large-scale graphs, we further include the **ogbn-arxiv** datasets from Open Graph Benchmark (OGB) [24]. For the ogbn-arxiv dataset, we randomly sample 1.0%, 1.5%, 2.0%, 2.5% nodes from its training splits as labeled data while using the same validation and test splits in OGB Benchmark [24]. Note that for all the datasets, we run each experiment 100 times with multiple random splits and different initializations.

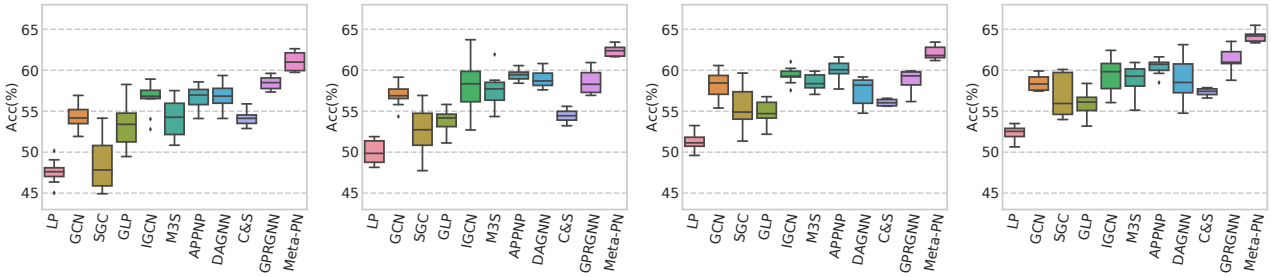
Compared Methods. To corroborate the effectiveness of our approach, three categories of baselines are included in our experiments: (i) *Classical Models*. **MLP**, **LP** (Label Propagation) [61] are two classical models using only feature and structure information, respectively. **GCN** [27] and **SGC** [56] are two representative GNN models. Due to the space limit, we omit some baselines like GAT, GraphSAGE since similar results can be observed; (ii) *Label-efficient GNNs*. **GLP** (Generalized Label Propagation)

Table 3: Test accuracy on few-shot semi-supervised node classification: mean accuracy (%) ± 95% confidence interval.

Method	Cora-ML		CiteSeer		PubMed		MS-CS	
	3-shot	5-shot	3-shot	5-shot	3-shot	5-shot	3-shot	5-shot
MLP	41.07 ± 0.76	51.12 ± 0.61	43.34 ± 0.56	44.90 ± 0.60	56.59 ± 0.93	59.90 ± 0.84	70.33 ± 0.37	79.41 ± 0.31
LP	62.07 ± 0.71	68.01 ± 0.62	54.07 ± 0.59	55.73 ± 1.19	58.75 ± 0.89	59.91 ± 0.85	57.96 ± 0.69	62.98 ± 0.61
GCN	48.02 ± 0.89	67.32 ± 1.02	53.60 ± 0.86	62.60 ± 0.58	58.89 ± 0.80	65.77 ± 0.98	69.24 ± 0.94	84.43 ± 0.89
SGC	49.60 ± 0.55	67.24 ± 0.86	57.37 ± 0.98	61.55 ± 0.53	63.37 ± 0.93	64.93 ± 0.81	72.11 ± 0.76	87.51 ± 0.27
GLP	65.57 ± 0.26	71.26 ± 0.31	65.76 ± 0.49	71.36 ± 0.18	65.34 ± 0.54	65.26 ± 0.29	86.10 ± 0.21	86.94 ± 0.23
IGCN	66.60 ± 0.29	72.50 ± 0.20	67.47 ± 0.29	72.92 ± 0.10	62.28 ± 0.23	65.19 ± 0.13	85.83 ± 0.06	87.01 ± 0.05
M3S	64.66 ± 0.31	69.64 ± 0.18	65.12 ± 0.20	68.18 ± 0.18	63.40 ± 0.32	68.85 ± 0.26	84.96 ± 0.18	86.83 ± 0.29
APPNP	72.39 ± 0.98	78.32 ± 0.58	67.55 ± 0.77	71.08 ± 0.61	70.52 ± 0.62	74.24 ± 0.87	86.65 ± 0.42	90.13 ± 0.86
DAGNN	71.86 ± 0.75	77.20 ± 0.69	66.62 ± 0.27	70.55 ± 0.12	71.22 ± 0.82	73.91 ± 0.71	86.32 ± 0.57	90.30 ± 0.66
C&S	68.93 ± 0.68	73.37 ± 0.24	63.02 ± 0.72	64.72 ± 0.53	70.51 ± 0.57	73.22 ± 0.57	85.86 ± 0.45	87.99 ± 0.24
GPR-GNN	70.98 ± 0.84	75.18 ± 0.52	64.32 ± 0.81	65.28 ± 0.52	71.03 ± 0.73	74.08 ± 0.65	86.12 ± 0.37	90.29 ± 0.38
Meta-PN	**74.94 ± 0.25**	**79.88 ± 0.15**	**70.48 ± 0.34**	**74.14 ± 0.50**	**73.25 ± 0.77**	**77.78 ± 0.92**	**88.99 ± 0.29**	**91.31 ± 0.22**

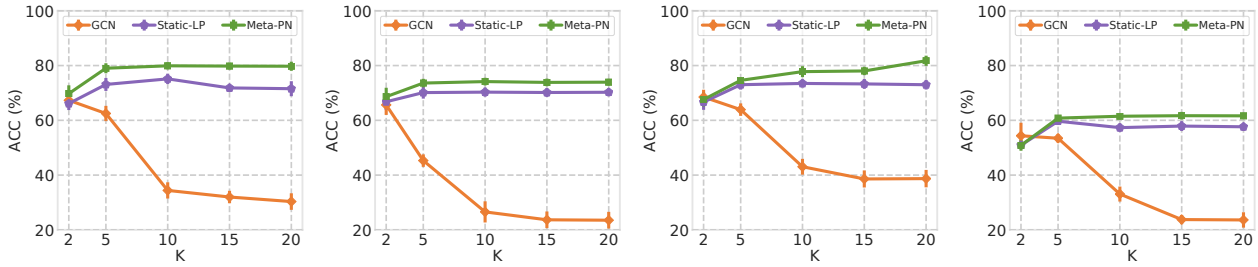
and **IGCN** (Improved GCN) [30] are two models combine label propagation and GCN from a unifying graph filtering perspective. **M3S** [49] is a multi-stage self-training framework, which incorporates self-supervised learning to improve the model performance with few labeled nodes; (iii) *Deep GNNs*. **APPNP** [28] decouples prediction and propagation with performing personalized propagation of neural predictions, while **DAGNN** [34] adaptively incorporate information from large receptive fields. **C&S** [25] is an effective model that combines label propagation and simple neural networks. **GPR-GNN** [8] addresses the limitation of APPNP on different types of graphs with adaptive propagation weights.

4.2.2 Few-shot Semi-supervised Evaluation First, we evaluate the proposed approach Meta-PN and all the baseline methods on few-shot semi-supervised node classification, which aims to predict the missing node labels with only a few labeled nodes. The average test accuracies under the few-shot setting (i.e., 3-shot and 5-shot) can be found in Table 3. From the reported results, we can clearly see that Meta-PN significantly outperforms all the baseline methods on each dataset based on paired t-tests with $p < 0.05$. Specifically, we elaborate our in-depth observations and analysis as follows: (i) without abundant labeled data, classical models including vanilla GNNs only obtain very poor classification accuracy under different evaluation entries; (ii) overall the label-efficient GNNs outperform classical GNNs, but still cannot achieve satisfying results. One major reason is that those methods cannot handle the oversmoothing issue since they are incapable of explicitly leveraging the knowledge from large receptive fields; (iii) by enabling better propagation of label signals, deep GNNs have stronger performance than both the classical models and label-efficient GNNs, which again demonstrates the necessity of addressing the oversmoothing issue for solving the few-shot semi-supervised learning problem. However, existing deep GNNs are not specifically developed to tackle the data sparsity issue, thus their performance still falls behind Meta-PN by a noticeable margin on different datasets when only very few labels are available. This observation proves that Meta-PN is able to address the overfitting and oversmoothing issues when labeled data is extremely sparse by combining the power of large receptive fields and pseudo labels.



(a) label ratio = 1.0% (b) label ratio = 1.5% (c) label ratio = 2.0% (d) label ratio = 2.5%

Figure 4: Comparison results on ogbn-arxiv w.r.t different size of training labels.



(a) Cora-ML (b) CiteSeer (c) PubMed (d) Ogbn-arxiv

Figure 5: Few-shot (i.e., 5-shot or 1.0% label ratio) evaluation on different datasets w.r.t. propagation steps (K).

4.2.3 Evaluation on Open Graph Benchmark (OGB). Real-world graphs commonly have a larger size and more node classes than many toy graphs, leading to the collected graphs having noisy structures and complex properties. To further illustrate the effectiveness of our approach on large-scale real-world graphs, we adopt the widely used ogbn-arxiv dataset and compare all the methods under the few-shot setting (i.e., from 1% to 2.5% label ratio). We summarize their performance for few-shot semi-supervised node classification on ogbn-arxiv in Figure 4 by changing the ratio of training labels, in which we omit MLP as its test accuracy is much lower than the other methods. We can observe that Meta-PN can significantly outperform all the baseline models under different few-shot environments. Compared to the other baseline methods, the performance of Meta-PN is relatively stable when we decrease the ratio of training labels, which demonstrates the robustness of Meta-PN in handling noisy and complex real-world graphs. Remarkably, our approach can achieve close performance to the vanilla GCN on ogbn-arxiv with much fewer labeled nodes (2.5% vs. 54%).

4.2.4 Parameter & Ablation Analysis. To demonstrate the effects of using different propagation steps and the importance of the meta-leaned label propagation strategy for Meta-PN, we compare our approach with two baselines under the 5-shot (or 1.0% label ratio for ogbn-arxiv) semi-supervised setting with varying number of propagation steps. Specifically, *GCN* learns the node representation with the standard message-passing scheme while *Static-LP* representing the variant of Meta-PN that uses fixed teleport probabilities instead of meta-learned ones. The evaluation results are shown in Figure 5. As we can observe from the figure, *GCN* can achieve very close performance with the other

two methods when the number of propagation steps is relatively small. While if we largely increase the number of propagation steps, the performance of *GCN* breaks down due to the oversmoothing issue. Empowered by the idea of label propagation, *Static-LP* can largely alleviate the oversmoothing issue and significantly outperform *GCN*. This verifies that larger propagation steps or receptive fields are necessary for improving the performance of GNN when labeled data is extremely limited. In the meantime, *Static-LP* still falls behind Meta-PN, mainly because of the infeasibility of balancing the importance of different receptive fields. On the contrary, Meta-PN is able to address this issue by inferring optimal pseudo labels on unlabeled nodes with our meta-learning algorithm. Its performance becomes stable when $K \geq 10$, indicating that Meta-PN can obtain good performance considering both efficiency and effectiveness with a moderate number of propagation steps (e.g., $K = 10$).

5 Future Work and Plan

This proposal defines a new research problem of learning with weak social supervision for understanding disinformation, and shows its potential and significance, but only touches upon the tip of the iceberg of this fertile research area. There are many extensions and work that are worth further explorations such early fake news detection and cross-domain fake news detection. We summarize the tasks to be finished and the plan in the near future.

5.1 Graph Self-Supervised Learning Research has shown that fake news spreads farther, faster, deeper, and more widely than true news [**?**]. Widespread fake news can erode the public trust in government and professional journalism and lead to adverse real-life events. Thus, a timely detection of fake news on social media is critical to cultivate a healthy news ecosystem.

It presents unique challenges and opportunities for early detection of fake news. First, fake news is diverse in terms of topics, content, publishing method and media platforms, and sophisticated linguistic styles geared to emulate true news. Consequently, training machine learning models on such sophisticated content requires *large-scale annotated fake news data* that is egregiously difficult to obtain. Second, it is important to detect fake news early. Most of the research on fake news detection rely on signals that require a long time to aggregate, making them unsuitable for *early detection*. Third, the evolving nature of fake news makes it essential to analyze it with signals from multiple sources to better understand the context. A system solely relying on social networks and user engagements can be easily influenced by biased user feedback, whereas relying only on the content misses the rich auxiliary information from the available sources.

5.2 Applications for Graph Anomaly Detection Real-world graphs are commonly contaminated with a small portion of nodes, namely, anomalies, whose patterns significantly deviate from the vast majority of nodes [13, 11]. For instance, in a citation network that represents citation relations between papers, there are some research papers with a few spurious references (i.e., edges) which do not comply with the content of the papers [2]; In a social network that represents friendship of users, there may exist camouflaged users who randomly follow different users, rendering properties like homophily not applicable to this type of relationships [18]. As the existence of even few abnormal instances could cause extremely detrimental effects, the problem of graph anomaly detection has received much attention in industry and academy alike. Due to the fact that labeling anomalies is highly labor-intensive and takes specialized domain-knowledge, existing methods are predominately developed in an unsupervised manner. However, the anomalies identified by unsupervised methods may turn out to

be data noises or uninteresting data instances due to the lack of prior knowledge on the anomalies of interest. Hence, we propose to transfer the knowledge of anomalies from auxiliary graphs to the target graph to enable accurate anomaly detection without abundant labeled anomalies.

5.3 Time Schedule The above proposed tasks are expected to be finished in a 6-month (03/2022 - 08/2022) period. A specific schedule is planned as follows.

- 03/2022 - 04/2022: prepare for comprehensive exam and proposal defense
- 04/2022 - 06/2022: to investigate graph self-supervised learning and graph anomaly detection;
- 06/2022 - 08/2022: to write up the thesis and prepare for the dissertation defense.

References

[1] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *IJCNN*, 2020.

[2] S. Bandyopadhyay, N. Lokesh, and M. N. Murty. Outlier aware network embedding for attributed networks. In *AAAI*, 2019.

[3] A. G. Baydin, R. Cornish, D. M. Rubio, M. Schmidt, and F. Wood. Online learning rate adaptation with hypergradient descent. In *ICLR*, 2018.

[4] A. J. Bose, A. Jain, P. Molino, and W. L. Hamilton. Meta-graph: Few shot link prediction via meta learning. *arXiv preprint arXiv:1912.09867*, 2019.

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

[6] S. Cao, W. Lu, and Q. Xu. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[7] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[8] E. Chien, J. Peng, P. Li, and O. Milenkovic. Adaptive universal generalized pagerank graph neural network. In *ICLR*, 2021.

[9] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2016.

[10] S. Deng, N. Zhang, J. Kang, Y. Zhang, W. Zhang, and H. Chen. Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. In *Proceedings of the International Conference on Web Search and Data Mining*, 2020.

[11] K. Ding, J. Li, N. Agarwal, and H. Liu. Inductive anomaly detection on attributed networks. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020.

[12] K. Ding, J. Li, R. Bhanushali, and H. Liu. Deep anomaly detection on attributed networks. In *Proceedings of the SIAM International Conference on Data Mining*, 2019.

[13] K. Ding, J. Li, and H. Liu. Interactive anomaly detection on attributed networks. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, 2019.

[14] K. Ding, J. Wang, J. Li, D. Li, and H. Liu. Be more with less: Hypergraph attention networks for inductive text classification. In *EMNLP*, 2020.

[15] K. Ding, J. Wang, J. Li, K. Shu, C. Liu, and H. Liu. Graph prototypical networks for few-shot learning on attributed networks. In *CIKM*, 2020.
[16] K. Ding, Z. Xu, H. Tong, and H. Liu. Data augmentation for deep graph learning: A survey. *arXiv preprint arXiv:2202.08235*, 2022.
[17] K. Ding, Q. Zhou, H. Tong, and H. Liu. Few-shot network anomaly detection via cross-network meta-learning. In *TheWebConf*, 2021.
[18] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *CIKM*, 2020.
[19] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*, 2017.
[20] V. Garcia and J. Bruna. Few-shot learning with graph neural networks. *Proceedings of the International Conference on Learning Representations*, 2018.
[21] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016.
[22] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2017.
[23] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.
[24] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.
[25] Q. Huang, H. He, A. Singh, S.-N. Lim, and A. R. Benson. Combining label propagation and simple models out-performs graph neural networks. In *ICLR*, 2021.
[26] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2016.
[27] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *NeurIPS*, 2017.
[28] J. Klicpera, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.
[29] Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.
[30] Q. Li, X.-M. Wu, H. Liu, X. Zhang, and Z. Guan. Label efficient semi-supervised learning via graph filtering. In *CVPR*, 2019.
[31] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.
[32] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. In *ICLR*, 2018.
[33] L. Liu, T. Zhou, G. Long, J. Jiang, and C. Zhang. Learning to propagate for graph meta-learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2019.
[34] M. Liu, H. Gao, and S. Ji. Towards deeper graph neural networks. In *KDD*, 2020.
[35] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.
[36] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 2001.
[37] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *Proceedings of the International Conference on Learning Representations*, 2018.
[38] G. Namata, B. London, L. Getoor, B. Huang, and U. EDU. Query-driven active surveying for collective classification. In *Workshop on MLG*, 2012.

[39] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[40] N. Park, A. Kan, X. L. Dong, T. Zhao, and C. Faloutsos. Estimating node importance in knowledge graphs using graph neural networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[41] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[42] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *Proceedings of the International Conference on Learning Representations*, 2017.

[43] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In *Proceedings of the International Conference on Learning Representations*, 2018.

[44] M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

[45] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 2009.

[46] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008.

[47] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[48] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2017.

[49] K. Sun, Z. Lin, and Z. Zhu. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *AAAI*, 2020.

[50] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[51] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[52] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*, 2018.

[53] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2016.

[54] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee. Next-item recommendation with sequential hypergraphs. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[55] N. Wang, M. Luo, K. Ding, L. Zhang, J. Li, and Q. Zheng. Graph few-shot learning with attribute matching. In *CIKM*, 2020.

[56] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger. Simplifying graph convolutional networks. *Proceedings of the International Conference on Machine Learning*, 2019.

[57] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *TNNLS*, 2020.

[58] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations*, 2019.

[59] H. Yao, C. Zhang, Y. Wei, M. Jiang, S. Wang, J. Huang, N. Chawla, and Z. Li. Graph few-shot learning via knowledge transfer. In *AAAI*, 2020.

[60] J. Zhang, C. Zhao, B. Ni, M. Xu, and X. Yang. Variational few-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[61] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NeurIPS*, 2004.

[62] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng. Meta-gnn: On few-shot node classification in graph meta-learning. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2019.

[63] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. *Technical Report*, 2002.

[64] D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In *KDD*, 2018.