Ada-GNN: Adapting to Local Patterns for Improving Graph Neural Networks*

Zihan Luo[1,2,3], Jianxun Lian[4], Hong Huang[1,2,3,†], Hai Jin[1,2,3], Xing Xie[4]
[1]National Engineering Research Center for Big Data Technology and System, Wuhan, China
[2]Service Computing Technology and Systems Laboratory, Wuhan, China
[3]Huazhong University of Science and Technology, Wuhan, China
[4]Microsoft Research Asia, Beijing, China
{luozhhh,honghuang,hjin}@hust.edu.cn,{jianxun.lian,xingx}@microsoft.com

ABSTRACT

Graph Neural Networks (GNNs) have demonstrated strong power in mining various graph-structure data. Since real-world graphs are usually on a large scale, training scalable GNNs has become one of the research trends in recent years. Existing methods only produce one single model to serve all nodes. However, different nodes may exhibit various properties thus require diverse models, especially when the graph is large. Forcing all nodes to share a unified model will decrease the model's expressiveness. What is worse, some small groups' patterns are prone to be ignored by the model due to their minority, making these nodes unpredictable and even some raising potential unfairness problems.

In this paper, we propose a model-agnostic framework Ada-GNN that provides personalized GNN models for specific sets of nodes. Intuitively, it is desirable that every node has its own model. But considering the efficiency and scalability of the framework, we generate specific GNN models at the subgroup-level rather than individual node-level. To be specific, Ada-GNN first splits the original graph into several non-overlapped subgroups and tags each node with its subgroup label. After that, a meta adapter is proposed to adapt a base GNN model to each subgroup rapidly. To better facilitate the global-to-local knowledge adaption, we design a feature enhancement module that captures the distinctions among different subgroups to improve the Ada-GNN's performance. Ada-GNN is model-agnostic and can be equipped to almost all existing scalable GNN based methods such as GraphSAGE, ClusterGCN, SIGN, and SAGN. We conduct extensive experiments with six popular scalable GNN as base methods on two large-scale datasets, and the results consistently demonstrate the generality and superiority of Ada-GNN.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Learning latent representations**; • **Information systems** → **Personalization**.

KEYWORDS

Graph Neural Networks; Local Adaption; Meta-learning

ACM Reference Format:
Zihan Luo, Jianxun Lian, Hong Huang, Hai Jin, Xing Xie. 2022. Ada-GNN: Adapting to Local Patterns for Improving Graph Neural Networks*. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3488560.3498460

1 INTRODUCTION

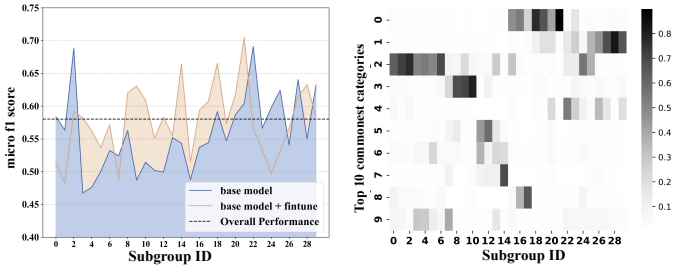
In recent years, *Graph Neural Networks* (GNNs) have become increasingly popular due to their versatile abilities in learning graph structure data and widespread applications such as social networks [11] [23], recommendation systems [6] [27], and bioinformatics [29] [14]. Considering that the real-world graphs are usually on a large scale, to facilitate the success of GNNs applications, a lot of efforts have been devoted to designing effective strategies to train GNNs efficiently on large-scale graphs. For example, GraphSAGE [4] performs neighborhood sampling to control the number of neighbors to be aggregated; ClusterGCN [2] first partitions the original graph into non-overlapped subgraphs with METIS [8], then performs graph convolutions on each subgraph. By eliminating the bias during subgraph sampling, GraphSaint [26] obtains a better way for subgraphs generation. On the other hand, another line of methods, such as SGC [21], SIGN [15], and SAGN [17], decouple the GNN into two parts: pre-processing and post-classification, and the latter can easily apply mini-batch training with the pre-processing output.

Although the above methods have achieved good performance on large-scale graphs, they only focus on using a unified model for representing all nodes in the graph while ignoring the diversity among nodes. In fact, some researchers have achieved some progress on the personalized model for each node, such as Policy-GNN [10], which designs specific aggregation policy for reflecting the diversity among nodes. However, since the web-scaled graphs in the real world, such as Facebook and Twitter, typically have $10^8 \sim 10^9$ nodes [15], it is almost infeasible to perform a node-level personalization on large-scale graphs due to the limitation of memory and efficiency, which leads us to consider this problem at a node-group-level (subgroup-level). For large-scale graphs, it is natural for nodes to form local communities as subgraphs, representing some common local patterns or sharing some semantic meanings. For example, in an academic network, researchers working in the same research area can form a subgraph; in the traffic network, nearby POIs can form a subgraph. Different

* This work is supported by National Natural Science Foundation of China (No. 62172174).
† Hong Huang is the corresponding author.

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498460



(a) Subgroup-level Performance (b) Subgroup Label Distribution

Figure 1: A study of subgroup diversity on the Arxiv1M dataset. (a) We train a SIGN model on the whole dataset (a.k.a the base model), then evaluate the model's performance on each subgroup separately (the blue curve). The performance is not even. We further finetune the base model with each subgroup's data respectively, and evaluate the dedicated model (the orange curve), we find that some subgroups improve but the others drop; (b) The label distribution in different subgroups are diverse.

subgraphs may demonstrate different patterns, so that one unified model may not be good enough to represent the subtle properties encoded in different subgraphs.

To better illustrate the diversity among subgroups, we take a one-million graph (Arxiv1M) for example. We divide the original graph into 30 subgroups by METIS [8] algorithm. After that, we conduct separate performance tests on each subgroup using SIGN [15] trained on the whole graph in advance. The results are reported in Figure 1(a). Besides the model performance, we also print the distributions for top 10 labels in each subgraph in Figure 1(b). It can be seen that the performance of the unified model on the whole graph is uneven on each subgroup, and different subgroups have different subgroup-specific data distributions. Moreover, we fine-tune the unified SIGN model with the local instances from each subgroup, the result is also shown in Figure 1(a). We find that simply applying the pretrain-then-finetune paradigm cannot achieve consistent improvement in general, which indicates its insufficiency in capturing different data distribution and motivates us to design a new method for model personalization at the subgroup-level.

However, it is non-trivial to design a subgroup-level personalized model for large graphs. The challenges mainly include: 1)*How to effectively capture and represent the distinctions among different subgroups (which we call local distinction)?* It is obvious that a model cannot perform personalized operations without perceiving the difference among subgroups; 2) *How to ensure the useful common knowledge among subgroups (which we call global coherence) is preserved after we eventually acquire personalized local models?* Although the distinctions play important roles in personalization, considering the fact that all subgroups are generated from the same large graph, the common characteristics of subgroups should not be ignored during personalized model learning. The performance drop of the pretrain-then-finetune model in some subgroups in Figure 1(a) exactly supports this assumption.

To address these challenges, in this paper, we propose a model-agnostic framework, **Ada-GNN**, to generate different models for different subgraphs after considering both the global coherence and local distinction. Ada-GNN is inspired by the framework of *Model-Agnostic Meta-Learning* (MAML) [3][12], whose goal is to train a base model from a variety of tasks, which can be adapted rapidly to serve for a new task with only a small number of task-specific training instances. Specifically, firstly we use a graph partition algorithm like METIS to divide the whole graph into multiple non-overlapped subgraphs, and tag each node with its corresponding subgraph ID as the group-wise label. Each subgraph can be regarded as a task, since it includes a set of nodes as instances. Next, we design a meta adapter module to learn a good global model from all subgroups and adapt to local models with a few instances in a subgraph. The global-to-local mode can help Ada-GNN both preserve global coherence and learn local distinction. At last, in case that in a normal MAML-like framework, local patterns may not be effectively reflected from a few support instances, we further propose a feature enhancement module to enhance raw features with group-wise signals, so that Ada-GNN can learn the distinctions among subgroups more easily.

The contributions of this work are summarized as follows:

- Instead of using one unified model to learn representations for all nodes on a large graph, we propose a model-agnostic framework Ada-GNN for almost all scalable GNNs to improve their performance by generating personalized models at the subgroup-level. To the best of our knowledge, this is the first time to take group-wise personalized model into consideration on large-scale graphs.
- We propose two core components in Ada-GNN, including a feature enhancement module and a meta adapter learner. The feature enhancement module can generate more informative features to capture the distinctions among subgroups, while the meta adapter learner has the ability to adapt a global model rapidly to a local model according to local training instances.
- We conduct comprehensive evaluations on two large-scale datasets with various base GNN models. The results demonstrate that Ada-GNN can effectively adapt to local patterns and thus improve all GNNs' performance significantly, and moreover, the improvement is consistently across different subgroups.

2 RELATED WORK

2.1 Graph Neural Networks

Many of the real-world data is organized as graph structures, such as social, bioinformatics, traffic, and citation networks. Recently, GNNs have attracted increasing attention in both academia and industry, due to their advantages in graph structure learning. For example, GCN [9] utilizes a simple convolution to aggregate information from node's neighbors, which provides efficient message passing on graphs. In contrast, GAT [20] uses an attention mechanism to aggregate information from neighbor nodes in a weighted way. Motivated by the Weisfeiler Lehman graph isomorphism test, GIN [24] uses a graph isomorphism network architecture to enhance the performance of GNNs. In order to stack more GNN layers and aggregate information from higher-order neighborhoods, some researchers try to use skip-connections to increase GNN models' depth. For example, aiming at addressing the problems of over smoothness and vanishing gradient, DeepGCN [13] borrows

the ideas from ResNet [5] and DenseNet [19] through residual connections and dense connections to help models go deeper. Wu et al. [22] offers a more comprehensive survey on existing graph neural networks.

2.2 Scalable GNNs

How to train GNNs on large graphs (such as a social network with millions of nodes) becomes one bottleneck, because a gradient-based update for one node involves a large number of neighboring nodes, which poses challenges on both computational cost and memory space. To address the challenge, a lot of works have been proposed to improve the time and memory efficiency of GNNs, which can be categorized into three groups.

1) Sampling Based Methods: As an efficient and effective way to alleviate the "neighbor explosion" problem in large graphs, neighbor sampling methods received wide research focus. GraphSAGE [4] randomly samples several neighbors for each node, which greatly alleviates the memory cost during neighborhood aggregation. Slightly different from GraphSAGE, PinSAGE [25] uses random walk to calculate the importance of neighbors for weighted sampling. Through sampling the receptive domain of each layer with importance, FastGCN [1] ensures that the important nodes will have a great chance to be sampled during aggregation.

2) Subgraph Based Methods: Subgraph based methods mainly aim at restricting the neighborhood search field through generating multiple subgraphs, and then reduce the computation cost. For example, inspired by mini-batch SGD, ClusterGCN [2] uses non-overlapped partition methods to split the original graph into several subgraphs and then trains GCN in each subgraph with less time and memory consumption. GraphSAINT [26] also tries to restrict nodes' receptive fields through generating subgraphs with the process of correcting bias and variance during sampling subgraphs.

3) Decoupling Based Methods: A lot of work tries to decouple the GNN model into graph pre-processing and post classification to reduce the computation complexity. By simply using linear layers as the post classifiers, SGC [21] achieves competitive performance with much less time consumption after removing intermediate non-linear activation. Inspired by the inception module in computer vision, SIGN [15] makes each hop's aggregation pass through a *multi-layer perceptrons* (MLP), and then concatenates the encoded representation as the input for a post MLP classifier. Based on SIGN, the concatenation operation is replaced by attention mechanism in SAGN [18] to further enhance the expressiveness of post classifier, and label propagation modules are added to improve the performance of model.

3 METHODOLOGY

3.1 Problem Formulation

To stay focus, in this paper we study the node classification problem, but the method can be easily extended to other graph applications such as link prediction.

A graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$, which consists of $N = |\mathcal{V}|$ vertices. $\mathbf{A} \in \{0,1\}^{N\times N}$ is the adjacency matrix, with $\mathbf{A}_{ij}$ equal to 1 if there is an edge between node i and node j, and 0 otherwise. $\mathbf{X} \in \mathbb{R}^{N\times F}$ denotes the F-dimensional attribute vector for each node in $\mathcal{V}$. Let Y denote the set of node labels, and each node i has one unique label $y_i \in Y$.

Generally, in node classification tasks, only part of nodes are associated with labels during training, denoted as $\mathcal{V}_L$. The goal is to learn a graph-based mapping function $f_\theta : \{v_i, \mathcal{G}\} \mapsto \{1, 2, ..., |Y|\}$ based on $\mathcal{V}_L$ as the training set, so that by leveraging the graph signal, f_θ can map each unlabeled node v_i in $\mathcal{V}$ to one label.

3.2 Framework Overview

Traditionally, GNN models such as SGC, ClusterGCN, and SIGN generate one single model f_θ after training, then use it to make predictions for all nodes. When the graph is large, it is intuitive that subgraphs located in different regions on the graph may have their own special patterns. For example, the traffic map of Beijing is different from that of Shanghai to a great extent, although they are both parts of the traffic map of China. Thus, the final model's expressiveness will be compromised if we force all the nodes in the big graph to share one unified model.

To address the problem, we propose Ada-GNN, which is a model-agnostic framework, to empower a GNN model with the ability of adapting to local patterns. An overall illustration of Ada-GNN is shown in Fig. 2. There are three main components in Ada-GNN, including node tagging, meta adapter, and feature enhancement module. Given an original graph $\mathcal{G}$, we first apply a graph partition method to generate M non-overlapped subgroups $\{\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_M\}$, then each node will be tagged with the ID of its belonging subgroup. This step is called node tagging. After that, to make subgroup-level patterns easier to be captured, we propose a feature enhancement module to generate additional subgroup features which record the distinctions among subgroups. Finally, inspired by the *Model-Agnostic Meta-Learning* (MAML) [3][12] framework, we propose an adaptive learner, called meta adapter, to ensure that a global GNN model can rapidly adapt to local patterns in each subgroup and finish the coarse-to-fine model transition. In addition, we design an fairness controller to alleviate the subgroup-wise unfairness problem in Ada-GNN. In the following sections, we will introduce these components of Ada-GNN in detail.

3.3 Node Tagging & Base GNN Model

Firstly, we employ a graph partition algorithm to divide the original graph into multiple disjoint subgraphs. How to partition the graph is not the research focus of this paper, and here we choose METIS [8] after considering its high efficiency and effectiveness in graph partition. After that, instead of making each subgraph a separated and independent graph, here we only tag each node with its corresponding subgroup ID, while leaving the original graph structure unchanged, which is different from the operations in ClusterGCN [2]. The advantages are two-fold: 1) preserving the original graph structure makes the proposed framework compatible with all kinds of base GNN models, because various graph operations like graph partition [2][26] and inception-like [15] propagation are still applicable; 2) node tagging does not cause any information loss due to the edges across subgraphs are not removed.

To fully represent the information of a node v_i, a GNN model aggregates the features from its neighbor nodes and fuses neighbor

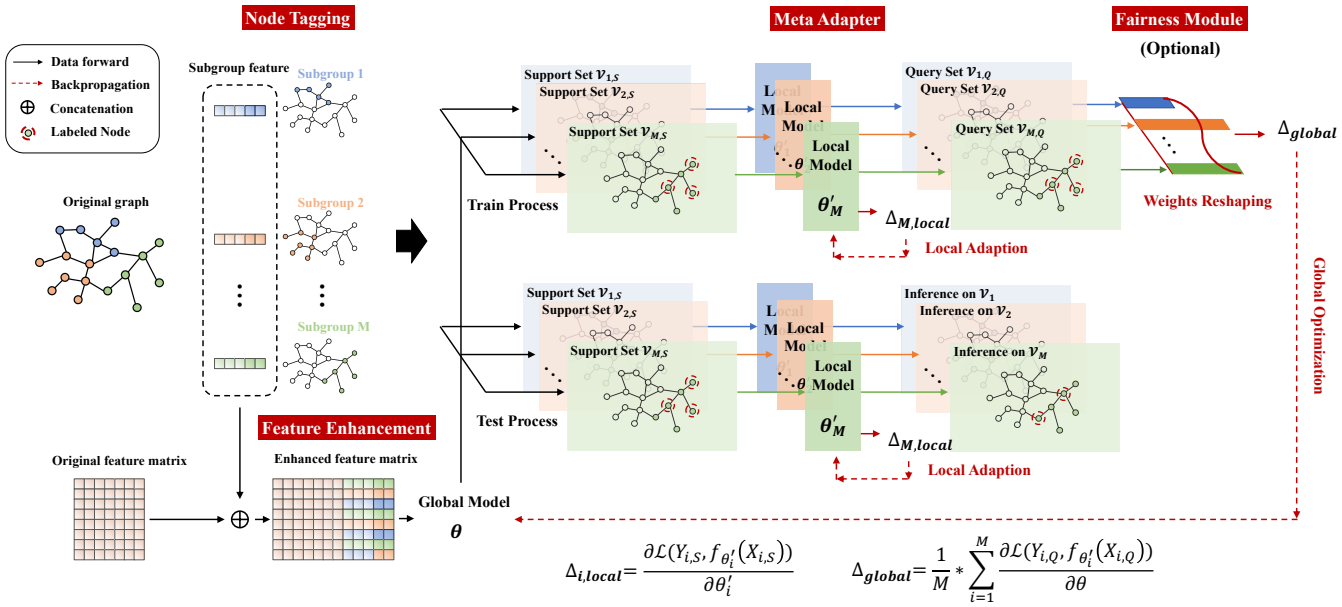


Figure 2: An overview of the Ada-GNN framework

information with its own. This aggregation can be repeated multiple times so that high-order neighborhood information can be captured. Ada-GNN is model-agnostic, but for better illustration, we take SAGN [17] as an example. SAGN aggregates neighbors' messages without feature transformation and non-linear activation:

$$\bar{\mathbf{X}}^{(k)} = \bar{\mathbf{A}}\bar{\mathbf{X}}^{(k-1)} \tag{1}$$

where $\bar{\mathbf{A}}$ is a transition matrix derived from $\mathbf{A}$ through a prepossessing step such as the row-stochastic random walk. $\bar{\mathbf{X}}^{(k)}$ is the k-hop smoothed node feature matrix. $\bar{\mathbf{X}}^{(0)}$ is the original feature matrix (which stores the node attributes). To obtain the $k-$hop representation $\mathbf{H}^{(k)}$, SAGN applies a MLP operation:

$$\mathbf{H}^{(k)} = MLP^{(k)}(\bar{\mathbf{X}}^{(k)}) \tag{2}$$

Multi-hop's neighborhood information is merged via an attention pooling:

$$\mathbf{H}_{att} = \sum_{k=0}^{L} \alpha^{(k)}\mathbf{H}^{(k)} \tag{3}$$

$$\alpha^{(k)} = softmax_k(LeakyReLU([\mathbf{H}^{(k)}||\mathbf{H}^{(0)}] \cdot \mathbf{W}_a)) \tag{4}$$

where $||$ means the concatenation operation and $\alpha^{(k)}$ is the attention score of k-hop neighborhood's information. SAGN's final representation of nodes is:

$$\mathbf{H}_f = MLP(\mathbf{H}_{att} + \mathbf{X}\mathbf{W}_r) \tag{5}$$

To fulfill the personalization for different subgroups, possible solutions include (1) finding a most suitable base GNN model for each subgroup, e.g., GraphSAGE for subgroup #1, SAGN for subgroup #2; (2) using one base model, but searching for a set of most suitable hyper-parameters for each subgroup, e.g., different hop depth k for different subgroups; (3) using one base model and one set of hyper-parameters, but adapting to different parameters for each subgroup, e.g., different subgroups have different attention parameters $\mathbf{W}_a$ and final merging parameters $\mathbf{W}_r$. We argue that method (3) is the best choice as a prior study in personalized GNN models, and the reasons are three-fold. Firstly, by sharing model backbone and hyper-parameters among subgroups, the structure of the GNN model is universal so that common knowledge is easy to share across subgroups. Secondly, the flexibility of changing model structure requires some complex techniques such as AutoML, which will increase the computational cost of the framework. Thirdly, theoretically, it is feasible to acquire subgroup personalization via adapting to different model parameters. For instance, if subgroup #1 mainly relies on self-information when neighbors' message is noisy, the final local model will adjust the parameter $\mathbf{W}_r$ in Eq.(5) to emphasize the influence of $\mathbf{X}$ for subgroup #1.

3.4 Meta Adapter

Inspired by the global-to-local learning framework in MAML [3][12], which aims at training a good global initialization θ that can help local models rapidly adapt to new tasks, we design a meta adapter to generate personalized models for different subgroups, while sharing common global knowledge from the original graph. The overall process is illustrated in the *Meta Adapter* module in Figure 2. Meta adapter first initializes the global model with random parameters θ. There are two kinds of parameters updates in meta adapter, named local adaption and global optimization. Each subgroup is regarded as one task, with training instances separated into support set and query set. Support set is used for local adaption. In other words, with copying θ as initialization for a local model θ_i', meta adapter trains the local model on support set to match the i^{th} subgroup's pattern. After local updates, the quality of local models is evaluated on the local query sets, and the gradients derived from query sets will guide the updating direction for the global model, which is called global optimization. Through this step, the meta adapter achieves

the ultimate goal: to generate a good global model, which can be adapted rapidly to different subgroups based on corresponding task-specific support sets. As for the inference stage, the meta adapter only has the local adaption step, because we do not have query labels for global optimization. More Specifically, meta adapter has the following two processes:

Train Process. Algorithm 1 in the appendix shows the detailed training process of Ada-GNN. Firstly, the parameters of the global model (line 1) are randomly initialized as θ. Then, based on a partitioning algorithm, M subgroups are generated for node tagging (line 2), denoted as $\{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_M\}$. After that, the meta adapter enters the phrase of subgroup-aware model training (lines 4-11). In each subgroup $\mathcal{V}_i$, training instances (which are labeled nodes in the training set) constitute support set $\mathcal{V}_{i,S}$ and query set $\mathcal{V}_{i,Q}$. On the support set, by minimizing the following loss function:

$$\mathcal{L}_i^S(\theta, \mathcal{G}) = \frac{1}{|\mathcal{V}_{i,S}|} \sum_{j \in \mathcal{V}_{i,S}} \mathcal{L}(y_j, \hat{y}_j(\theta)) \tag{6}$$

which is also named as train loss, we can get the adapted local parameters θ_i':

$$\theta_i' = \theta - \gamma_1 \frac{\partial \mathcal{L}_i^S(\theta, \mathcal{G})}{\partial \theta} \tag{7}$$

where y_j, $\hat{y}_j$ represent the true label and predicted label for node j, respectively. $\mathcal{L}$ is the entropy loss function in most cases. γ_1 is the local learning rate. We hope that through the local adaption step, errors in the query set $\mathcal{V}_{i,Q}$ can be minimized and local distinctions can be learned. To this end, we calculate the evaluation loss in query set:

$$\hat{\mathcal{L}}_i = \mathcal{L}_i^Q(\theta_i', \mathcal{G}) = \frac{1}{|\mathcal{V}_{i,Q}|} \sum_{j \in \mathcal{V}_{i,Q}} \mathcal{L}(y_j, \hat{y}_j(\theta_i')) \tag{8}$$

and use it to update the global model with following functions:

$$\mathcal{L}_{meta} = \sum_{i=1}^{M} \hat{\mathcal{L}}_i \tag{9}$$

$$\theta = \theta - \gamma_2 \frac{\partial \mathcal{L}_{meta}}{\partial \theta} \tag{10}$$

where $\mathcal{V}_{i,Q}$ denotes the query set in subgroup $\mathcal{V}_i$, M denotes the subgroup number, $\hat{\mathcal{L}}_i$ denotes the evaluation loss of subgroup $\mathcal{V}_i$, and γ_2 is the global learning rate. Through accumulating multiple subgroups' evaluation loss and applying it to the global parameters once (line 11 and 13), θ would be forced to learn the global coherence among all subgroups. Normally, the train nodes and validation nodes in each subgroup should represent support set and query set, respectively. However, because we do not want to use validation sets during training, which might cause an unfair comparison with other scalable GNNs, we re-use the train nodes in each subgroup as query set as well. Thanks to the global-to-local mode, meta adapter could successfully find desirable parameters θ which could adapt to an optimal space with only K steps for all subgroups. The algorithm will repeat the above procedures until θ is stable.

Test Process. The test process of Ada-GNN is very similar to the train process. The only difference is that there is no global optimization for global model θ anymore. After local adaption, each local model will be directly evaluated through the test nodes in the corresponding subgroup. Note that, there is no need for meta adapter to store all local models' parameters after training. Instead, with a trained global initialization, the global model θ can rapidly adapts to θ_i for subgroup i via corresponding support sets $\mathcal{V}_{i,S}$. In other words, given the trained global parameters θ, Ada-GNN could rapidly generate all local models (lines 3-6) and perform an evaluation on each subgroup (line 8). The pseudo codes of the test process are shown in Algorithm 2 in the appendix.

3.5 Feature Enhancement Module

Essentially, the classical MAML framework relies on the distribution of <feature, label> mapping to capture the special patterns in a local task in an implicit manner. However, when the support set is small (which is the common case for a MAML setting), the local pattern cannot be effectively reflected. We argue that to facilitate the local model adaption process, it will be helpful to provide informative group-level signals to describe the local patterns in an explicit manner. These signals should be easy to be obtained, informative to distinguish subgroups, and can directly influence the model prediction [16].

To this end, we propose a feature enhancement module that extracts and appends group-level features to node attributes for better distinction among subgroups. The group-level features include 1) normalized labels distribution; 2) one-hot encoding subgroup ID. The two auxiliary information will be appended to the original node features. Due to the limited number of classes and subgroups, feature enhancement module only adds little computation and memory costs while yielding much better performance. As a result, the new feature vector for node u in subgroup $\mathcal{V}_i$ is derived as:

$$\bar{\mathbf{x}}_u = [\mathbf{x}_u, \mathbf{a}_i] \tag{11}$$

where $[\cdot, \cdot]$ indicates the concatenation operation, $\mathbf{a}_i$ denotes the group-level features for subgroup $\mathcal{V}_i$, which can be represented as:

$$\mathbf{a}_i = [distribution_i, ID_i] \tag{12}$$

where $distribution_i$ denotes the label distribution in subgroup $\mathcal{V}_i$, and ID_i denotes the one-hot encoded ID index of subgroup $\mathcal{V}_i$. After feature enhancement, the meta adapter can perceive and adapt to local patterns better.

3.6 Group-wise Fairness Discussion

The classical supervised learning paradigm aims to minimize the overall loss in Eq. 9, which indicates an optimal accuracy performance in the global view. To achieve this, the model will often converge to a tricky state where for some groups the performance is extremely high and for some other groups the performance is extremely low. This is one type of *unfairness* problems, which means that some groups' performance is sacrificed too much to pursue an overall metrics. From the view of responsible *Artificial Intelligence* (AI), we hope the performance of different groups should not be biased severely. A natural merit of Ada-GNN is that fairness level can be easily controlled in the framework.

Specifically, we slightly modify Eq. 9 to mitigate the potential unfairness problem and let the model spare more effort on subgroups that have poor performance:

$$\mathcal{L}_{meta}^* = \sum_{i=1}^{M} w_i \hat{\mathcal{L}}_i \tag{13}$$

Table 1: Dataset Statistics

Dataset	Vertices	Edges	Class	Train/Val/Test
Arxiv1M	1,546,782	13,701,428	172	0.71 / 0.17 / 0.12
Amazon2M	2,449,029	61,859,140	47	0.70 / 0.20 / 0.10

where w_i represents the weight for subgroup $\mathcal{V}_i$, which is determined by the accuracy of the subgroup on the validation set with softmax as modulation:

$$z_i = \frac{1/mic_i}{\sum_{i=1}^{M} 1/mic_i} \tag{14}$$

$$w_i = \frac{exp((z_i - max(z))/\tau)}{\sum_{i=1}^{M} exp((z_i - max(z))/\tau)} \tag{15}$$

where mic_i is the micro f1 score of subgroup $\mathcal{V}_i$, and τ is temperature parameter. With this fairness module, the weights of groups with poor performance will be dynamically lifted during the training process. The fairness module is optional in Ada-GNN, it depends on the trade-off between the global accuracy and group-wise fairness concern.

4 EXPERIMENTS

4.1 Experimental Settings

We evaluate Ada-GNN and other baselines on a server with Intel(R) Xeon(R) CPU E5-2680 and GPU Tesla V100 with 32GB Memory. Our experiment environment is Ubuntu 18.04 with CUDA 11.4. All methods are implemented using Python 3.7 with PyTorch 1.8.0 and *Deep Graph Library* (DGL) 0.6 [28]. To be fair, we report the average results after repeating each method five times.

4.1.1 ***Datasets.*** We perform experiments on two large-scale graph datasets. The basic statistics are shown in Table 1.

- **Arxiv1M**: This is a homogeneous network composed of a subset of papers that are published in arXiv. All paper nodes are indexed by ogbn-papers100M in advance [7]. Each node represents an arXiv paper, and each directed edge represents the citation relationship between two paper nodes. By averaging the embeddings of words in the title and abstract of the paper, a 128-dimensional feature vector is attached to each paper node. In total, there are 172 classes, which represent 172 arXiv subject areas.
- **Amazon2M**: An undirected item-item graph in which vertices are products sold by Amazon and edges represent whether two items are purchased by the same customer [7]. Each product node comes with a 100-dimensional feature vector which is generated from the product's description. In total, 47 product categories are used as node labels.

4.1.2 ***Baselines.*** Intuitively, the node diversity phenomenon is more severe on large graphs, we are more interested to see how Ada-GNN can improve those base GNN model which are scalable. Thus, we choose six different GNN methods, including **GraphSAGE** [4], **SGC** [21], **ClusterGCN** [2], **GraphSAINT** [26], **SIGN** [15], and **SAGN** [17]. They represent different types of scalable structures: GraphSAGE samples a fixed number of neighbors for each node to avoid the neighborhood explosion problem; SGC reduces model complexity by eliminating non-linear functions between GCN layers and transforming nonlinear GCNs into simple linear models; ClusterGCN is a framework for training large-scale GNN via partitioning the origin graph into small disjoint subgraphs; GraphSAINT explicitly considers the deviation caused by subgraph sampling on GCN calculation, which can ensure that the aggregation process of nodes after sampling is unbiased and the variance caused by sampling is as small as possible; SIGN passes each hop's neighborhood aggregation through a MLP encoder and then concatenates the encoded representation as the input for a post MLP classifier; SAGN leverages attention module to attentively merge multi-hop's neighborhood message, and uses a *Self-Label-Enhance* (SLE) training mechanism to improve the model performance.

4.1.3 ***Implementation Details.*** For all baselines, the hyper-parameters are set to be the same with the corresponding paper's settings, and in order to provide a fair comparison, all hidden dimensions are set to 128. We use the open-sourced codes released by the authors and DGL official forum for comparison with Ada-GNN. As for Ada-GNN, we use METIS algorithm to perform node tagging and generate subgroups. Due to the limited number of training samples, too much subgroups might cause insufficient training for each local model and huge memory cost. Therefore, unlike the large partition size in ClusterGCN, we set subgroups' number M as 5 in our work. Besides, to make Ada-GNN adapt more rapidly, the adaption step K should not be large, and is set to 5 except in the hyper-parameter study section. Both local learning rate γ_1 and global learning rate γ_2 are set to 0.005. Other hyper-parameters, such as weight decay rate and dropout rate, are all set to be the same with the settings in corresponding local models. The detailed hyper-parameters analysis in Ada-GNN will be further discussed in subsection 4.6.

4.2 Overall Evaluation of Ada-GNN

We evaluate the node classification performance of our model – Ada-GNN and other different models on Amazon2M and Arxiv1M. To demonstrate that Ada-GNN is a model-agnostic framework and can be applied to different base GNN models, we choose GraphSAGE, SGC, ClusterGCN, GraphSAINT, SIGN, and SAGN as base GNN models in experiments, which are denoted as Ada-GraphSAGE, Ada-SGC, Ada-ClusterGCN, Ada-GraphSAINT, Ada-SIGN, and Ada-SAGN, respectively. Table 2 reports the overall performance of our proposed model as well as the baseline methods, from which we have following observations:

- Equipped with the Ada-GNN framework, all the base models get consistent improvement in both micro-f1 and macro-f1 performance on two datasets, which demonstrates the necessity of local adaption in big graphs and the effectiveness of our proposed framework in improving various types of GNN models.
- Among the base models, SAGN achieves the best performance, and Ada-SAGN can further improve it on both Arxiv1M and Amazon2M dataset, reaching the state-of-the-art performance. Overall, decoupling based methods (SIGN, SAGN) generally perform better than subgraph based methods (ClusterGCN, GraphSAINT), which is consistent with the conclusions in existing literature [17].

Table 2: Overall performance (in percentage) comparisons of Ada-GNN with six different base models

Models	Arxiv1M		Amazon2M	
	micro-f1	macro-f1	micro-f1	macro-f1
GraphSAGE	$54.12_{\pm 0.03}$	$32.17_{\pm 0.21}$	$83.01_{\pm 0.04}$	$47.77_{\pm 0.18}$
SGC	$52.73_{\pm 0.17}$	$28.96_{\pm 0.28}$	$74.17_{\pm 0.01}$	$37.88_{\pm 0.47}$
ClusterGCN	$55.49_{\pm 0.02}$	$34.01_{\pm 0.41}$	$85.71_{\pm 0.02}$	$49.87_{\pm 0.27}$
GraphSAINT	$54.60_{\pm 0.02}$	$24.31_{\pm 0.34}$	$87.02_{\pm 0.01}$	$44.40_{\pm 0.22}$
SIGN	$58.20_{\pm 0.03}$	$35.85_{\pm 0.25}$	$85.38_{\pm 0.06}$	$50.34_{\pm 0.43}$
SAGN	$59.42_{\pm 0.02}$	$36.07_{\pm 0.15}$	$87.56_{\pm 0.03}$	$55.21_{\pm 0.13}$
Ada-GraphSAGE	$54.54_{\pm 0.15}$	$33.05_{\pm 0.23}$	$85.47_{\pm 0.13}$	$50.10_{\pm 0.57}$
Ada-SGC	$53.87_{\pm 0.29}$	$32.53_{\pm 0.37}$	$76.15_{\pm 0.01}$	$42.42_{\pm 2.78}$
Ada-ClusterGCN	$58.25_{\pm 0.17}$	$35.33_{\pm 0.31}$	$86.56_{\pm 0.05}$	$51.15_{\pm 1.23}$
Ada-GraphSAINT	$55.90_{\pm 0.21}$	$33.15_{\pm 0.44}$	$87.73_{\pm 0.11}$	$45.67_{\pm 1.31}$
Ada-SIGN	$59.60_{\pm 0.11}$	$35.81_{\pm 0.75}$	$86.79_{\pm 0.14}$	$52.62_{\pm 0.39}$
Ada-SAGN	$\mathbf{59.92}_{\pm 0.12}$	$\mathbf{36.19}_{\pm 0.27}$	$\mathbf{87.84}_{\pm 0.08}$	$\mathbf{55.32}_{\pm 0.13}$

- All models perform better on Amazon2M than on Arxiv1M. The main reason may be that the number of class in Arxiv1M is much more than that in Amazon2M, which makes it be much harder to predict. In addition, the density (i.e., the average degree of nodes) of the Amazon2M graph is much higher than that of the Arxiv1M graph, which yields more sufficient neighbor information, thus bring better performance.

4.3 Subgroup-level Analysis

In order to analyze the influence of Ada-GNN on each subgroup after learning the local patterns, we set the subgroup number to 5 and compare the results between each base model and its corresponding adapted model at subgroup-level. The results are shown in Figure 3 and Figure A1 in the appendix. As illustrated in the figures, for most of the cases, Ada-GNN's improvement is consistent across subgroups with different base models, due to the successful subgroup-level personalization.

One interesting observation is that all models demonstrate the same bias on subgroup-level's performance trend and unfairness problem. For example, on the Arxiv1M dataset, all models follow the trend that subgroup #1 > subgroup #5 > subgroup #2 > subgroup #4 > subgroup #3, which indicates that the main reason for uneven performance is not due to the model's nature, but caused by diversity of data distribution across different subgroups. This phenomenon supports the motivation of enabling subgroup personalization and the following fairness controller. After applying Ada-GNN, base models' performance on all subgroups are improved, which verifies that the overall gain of Ada-GNN in Table 2 does not come from sacrificing one subgroup's performance to remedy another subgroup's performance. Instead, Ada-GNN can help local model to better learn the local patterns of each subgroup, so almost all subgroups get improved.

4.4 Ablation Study

In this section, we investigate the contribution of the meta adapter and the feature enhancement module to Ada-GNN, by removing one of them from Ada-GNN in a time (for example, without the meta adapter or without the feature enhancement) and check how

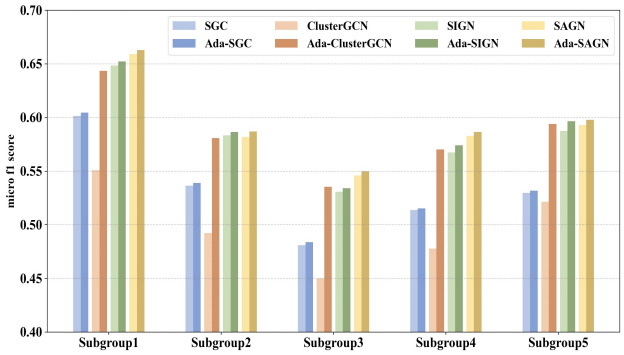


Figure 3: Subgroup-level comparisons between base models and adapted local models on Arxiv1M

the performance is impacted. We further compare Ada-GNN with a simple *pretrain + finetune* paradigm, which regards the base model as a pretrained global model and fine-tunes it with subgroup's samples only before evaluating the subgroup's test samples. From Table 3 we have the following observations:

- The *pretrain + finetune* paradigm cannot bring consistent improvement over the base model, while Ada-GNN can bring consistent improvement over the base model and in most of the cases outperforms *pretrain + finetune* (The only exception is with SGC. However, SGC's best performance is still far behind the other models'). This indicates that learning to keep the global coherence is indispensable during adapting to local distinctions.
- Removing either the meta adapter or the feature enhancement module will lead to a significant performance drop. For example, in Ada-SIGN, when removing the meta adapter, the performance drops from 0.5960 to 0.5936; while removing the feature enhancement module, it drops from 0.5960 to 0.5714.
- By comparing *base model* with w/o adapter, we can see that subgroup-level features indeed carry useful signals which enhance the attributes of nodes, and by comparing w/o adapter with Ada-GNN, we can see that meta adapter is a better way to make use of subgroup-level features.
- Without feature enhancement module, meta adapter can not consistently improve the base model. This matches our expectation, because in the local adaption step, we only use a few supporting instances which may not reflect the unique data distribution of each subgroup well.

4.5 Fairness Analysis

Next, we evaluate the optional module - the fairness controller - to verify if the group-wise variance can be alleviated by adding this module. To be specific, we compare the standard deviations of subgroup-level performance among settings of applying Ada-GNN with (w/) or without (w/o) fairness module, as well as the base model, on the Amazon2M dataset. We follows the experimental setting in Section 4.1 and τ in Eq.15 is set to 0.01. From Table 4 we have these observations:

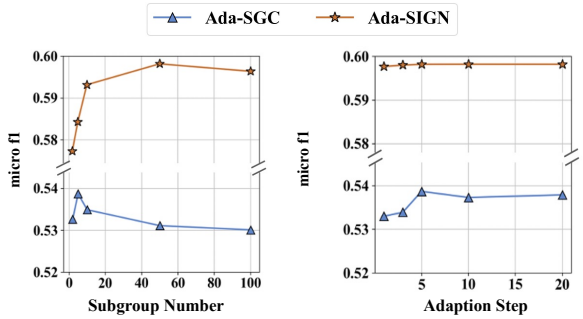
- After applying fairness module, the subgroup-level performance's standard deviation of Ada-GNN is indeed reduced with all base models, which indicates that the

Table 3: An ablation study on removing the meta adapter or/and the feature enhancement module (in micro-f1). Numbers in bold type indicate the best setting for each base model.

Model	Arxiv1M					Amazon2M				
	base model	pretrain+ finetune	w/o adapter	w/o feature	Ada-GNN	base model	pretrain+ finetune	w/o adapter	w/o feature	Ada-GNN
Ada-SGC	0.5273	**0.5412**	0.5329	0.5336	0.5387	0.7417	**0.7852**	0.7601	0.7479	0.7615
Ada-ClusterGCN	0.5549	0.5615	0.5806	0.5816	**0.5825**	0.8571	0.8602	0.8606	0.8576	**0.8656**
Ada-SIGN	0.5820	0.5840	0.5936	0.5714	**0.5960**	0.8538	0.8529	0.8644	0.8559	**0.8679**
Ada-SAGN	0.5942	0.5927	0.5977	0.5960	**0.5992**	0.8756	0.8736	0.8772	0.8764	**0.8784**

Table 4: Comparison on overall performance and standard deviations among base models, Ada-GNNs (Ada), and Ada-GNNs with fairness module (Ada$_{Fair}$)

Model	SAGE		SGC		SAINT		SIGN	
	mic	std	mic	std	mic	std	mic	std
base	83.01	2.20	74.28	2.48	87.02	2.90	85.38	1.70
Ada	85.47	2.54	76.10	2.35	87.73	3.01	86.79	2.41
Ada$_{Fair}$	85.00	**1.72**	76.00	**1.41**	87.21	**1.48**	85.84	**1.60**



(a) Influence of subgroup number M on Ada-GNN

(b) Influence of adaption steps K on Ada-GNN

Figure 4: The influence of subgroup numbers M (a) and adaption steps K (b) over Arxiv1M. For conciseness we only demonstrate two base models: SGC and SIGN.

performance of all subgroups becomes much more even with the help of fairness module.

- The overall performance of Ada-GNN will slightly decrease after applying fairness module. The main reason might be that the fairness module forces meta adapter to focus more on inferior subgroups, and this leads to less improvement on subgroups which are easy to learn. Thus, currently it is trade-off between the optimal accuracy and decent fairness. We leave the problem of fulfilling fairness while keeping optimal accuracy as a future work. Note that, although fairness module will hurt the performance of Ada-GNN to some extent, there are still obvious improvement compared with base GNN models.

4.6 Hyper-parameters Analysis

We analyze two most important hyper-parameters of Ada-GNN: the number of subgroups M during node tagging, and the number of adaption steps K in meta adapter.

4.6.1 ***The number of subgroups*** M. We show how the number of subgroups M during node tagging would influence Ada-GNN. To study the impact of the subgroup numbers M in our model, we investigate the performance of Ada-SGC, Ada-SIGN by varying the values of M in [2, 5, 10, 50, 100]. As shown in Figure 4(a), with the increase of M, there is a downward trend on performance over Arxiv1M, which indicates that a too large subgroups number might cause insufficient training samples, and harm the model performance. On Amazon2M, similar results can be shown in Figure A2(a) in the appendix. We also find an interesting observation that in Ada-SGC and Ada-SIGN, the performance increases at first and then drops slowly. We hypothesize that Ada-SGC might benefit from the personalization when we adjust M from 2 to 5, while Ada-SIGN's performance get consistent improvement due to better personalization when we increase M from 2 to 50.

4.6.2 ***The number of adaption steps*** K. In this part, we test the impact of the adaption steps K on model performance. In expectation, a small number of adaption steps are enough to capture the subgroup pattern, and further increasing the adaption step number will not further improve the performance but cause more computational time. To verify this, we vary the adaption steps K in [1, 3 ,5, 10, 20]. As shown in Figure 4(b), in Arixv1M, with the increase of K, performance of Ada-SGC and Ada-SIGN both increase at first until reaching a converged status with slight fluctuation. For conciseness, the results on Amazon2M are provided in Figure A2(b) in the appendix. Note that, in some cases, Ada-GNNs even could achieve optimal performance with $K = 1$. The result matches our expectation, which indicates that Ada-GNN could rapidly adapt to local models with limited adaption steps with a trained global initialization. Usually, a setting of $K = 5$ could be near-optimal.

5 CONCLUSION

In this paper, we propose a model-agnostic framework Ada-GNN, which learns personalized models for different subgroups by considering both their distinctions and similarities. We adopt the concept of global-to-local in MAML for designing our meta adapter, which provides Ada-GNN with the capability of rapidly adapting to local models. Moreover, node tagging and feature enhancement module are designed for splitting different subgroups and capturing the distinctions among them, respectively. At last, we discuss the issue of fairness in Ada-GNN and propose fairness module as an optional solution. Extensive experimental results on two large-scale datasets with six different base GNN models demonstrate that Ada-GNN can consistently improve various GNN models.

REFERENCES

[1] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018*. OpenReview.net. https://openreview.net/forum?id=rytstxWAW

[2] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 257–266. https://doi.org/10.1145/3292500.3330925

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1126–1135. http://proceedings.mlr.press/v70/finn17a.html

[4] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 770–778. https://doi.org/10.1109/CVPR.2016.90

[6] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 639–648. https://doi.org/10.1145/3397271.3401063

[7] 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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html

[8] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.* 20, 1 (1998), 359–392. https://doi.org/10.1137/S1064827595287997

[9] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[10] Kwei-Herng Lai, Daochen Zha, Kaixiong Zhou, and Xia Hu. 2020. Policy-GNN: Aggregation Optimization for Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2020, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 461–471. https://doi.org/10.1145/3394486.3403088

[11] Dong-Ho Lee, Yu-Ri Kim, Hyeong-Jun Kim, Seung-Myun Park, and Yu-Jun Yang. 2019. Fake News Detection Using Deep Learning. *J. Inf. Process. Syst.* 15, 5 (2019), 1119–1130. http://www.jips-k.org:80/q.jips?cp=pp&pn=707

[12] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 1073–1082. https://doi.org/10.1145/3292500.3330859

[13] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs Go As Deep As CNNs?. In *Proceedings of 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 9266–9275. https://doi.org/10.1109/ICCV.2019.00936

[14] Gainza Pablo, Sverrisson Freyr, Monti Frederico, Rodola Emanuele, Boscaini Davide, Bronstein Michael, and Correia Bruno. 2019. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods* 17, 4 (2019), 1–9.

[15] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael M. Bronstein, and Federico Monti. 2020. SIGN: Scalable Inception Graph Neural Networks. *CoRR* abs/2004.11198 (2020). arXiv:2004.11198 https://arxiv.org/abs/2004.11198

[16] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, and Xiaoqiang Zhu. 2021. One Model to Serve All: Star Topology Adaptive Recommender for Multi-Domain CTR Prediction. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management, CIKM 2021, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.). ACM, 4104–4113. https://doi.org/10.1145/3459637.3481941

[17] Chuxiong Sun and Guoshi Wu. 2021. Scalable and Adaptive Graph Neural Networks with Self-Label-Enhanced training. *CoRR* abs/2104.09376 (2021). arXiv:2104.09376 https://arxiv.org/abs/2104.09376

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

[19] Naoya Takahashi and Yuki Mitsufuji. 2021. Densely Connected Multi-Dilated Convolutional Networks for Dense Prediction Tasks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 993–1002. https://openaccess.thecvf.com/content/CVPR2021/html/Takahashi_Densely_Connected_Multi-Dilated_Convolutional_Networks_for_Dense_Prediction_Tasks_CVPR_2021_paper.html

[20] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018*. OpenReview.net. https://openreview.net/forum?id=rJXMpikCZ

[21] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6861–6871. http://proceedings.mlr.press/v97/wu19e.html

[22] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[23] Zhiyuan Wu, Dechang Pi, Junfu Chen, Meng Xie, and Jianjun Cao. 2020. Rumor detection based on propagation graph neural network with attention mechanism. *Expert Syst. Appl.* 158 (2020), 113595. https://doi.org/10.1016/j.eswa.2020.113595

[24] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=ryGs6iA5Km

[25] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 974–983. https://doi.org/10.1145/3219819.3219890

[26] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=BJe8pkHFwS

[27] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, 2017, Halifax, NS, Canada, August 13-17, 2017*. Association for Computing Machinery, New York, NY, USA, 635–644. https://doi.org/10.1145/3097983.3098063

[28] Da Zheng, Minjie Wang, Quan Gan, Zheng Zhang, and George Karypis. 2020. Learning Graph Neural Networks with Deep Graph Library. In *Companion of The 2020 Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Amal El Fallah Seghrouchni, Gita Sukthankar, Tie-Yan Liu, and Maarten van Steen (Eds.). ACM / IW3C2, 305–306. https://doi.org/10.1145/3366424.3383111

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

A APPENDIX

A.1 Ada-GNN Algorithms

Algorithm 1 Ada-GNN Train Process

Input: Graph G, Hyper-parameters: local learning rate γ_1, global learning rate γ_2, subgroup nums M and local adaption steps K.
Output: Global model parameters θ

Randomly initialize global model parameters as θ.
Split G into M subgroups, denoted as $\{\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_M\}$.
while not converged **do**
  for each subgroup $\mathcal{V}_i$ **do**
    Initialize the local model parameters θ_i' with θ.
    for each adaption step **in** K **do**
      Compute the train loss $\mathcal{L}_i(\theta_i')$ through support set $\mathcal{V}_{i,S}$ in subgroup $\mathcal{V}_i$.
      Local updates: $\theta_i' = \theta_i' - \gamma_1 \frac{\partial \mathcal{L}_i(\theta_i')}{\partial \theta_i'}$.
    end for
    Compute evaluation loss $\hat{\mathcal{L}}_i$ through query set $\mathcal{V}_{i,Q}$ in subgroup $\mathcal{V}_i$.
    $\mathcal{L}_{meta}$ += $\hat{\mathcal{L}}_i$
  end for
  Global updates: $\theta = \theta - \gamma_2 \frac{\partial \mathcal{L}_{meta}(\theta')}{\partial \theta}$.
end while
return θ

Algorithm 2 Ada-GNN Test Process

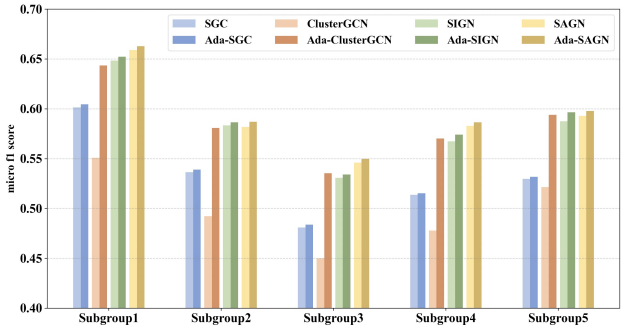
Input: Graph G, Splitted subgroups $\{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_M\}$, Trained global parameters θ, Hyper-parameters: local learning rate γ_1, local adaption steps K.

Initialize global model with trained parameters θ.
for each subgroup $\mathcal{V}_i$ **do**
  Initialize the local model parameters θ_i' with θ.
  for each adaption step **in** K **do**
    Compute the train loss $\mathcal{L}_i(\theta_i')$ through support set $\mathcal{V}_{i,S}$ in subgroup $\mathcal{V}_i$.
    Local updates: $\theta_i' = \theta_i' - \gamma_1 \frac{\partial \mathcal{L}_i(\theta_i')}{\partial \theta_i'}$.
  end for
  Evaluation through test nodes in subgroup $\mathcal{V}_i$ with adapted local model θ_i'.
end for

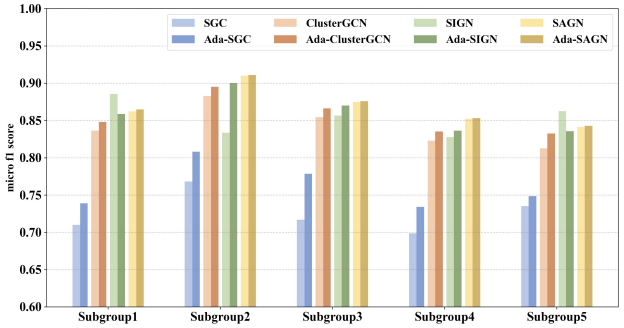
We provide the complete process of Ada-GNN's training process and test process in Algorithm 1 and Algorithm 2 respectively, which are the core components helping Ada-GNN to learn both local and global information simultaneously. Those algorithms are also literally described in Section 3.4 for better understanding.

A.2 Subgroup-level Analysis

As shown in Figure A1, the Ada-GNN brings consistent improvement for almost all base models on the subgroup-level. The only two exceptions are with SIGN on subgroup #1 and subgroup #5 in Amazon2M. The reason may be the insufficient model personalization caused by a small subgroup number. In Section 4.6.1, we further discuss the influence of subgroup number M's influence on Ada-GNNs' performance.

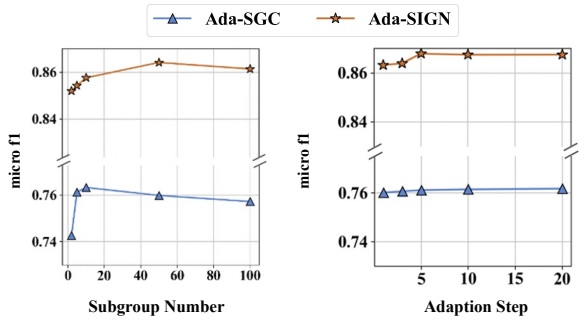


(a) Performance on Arxiv1M



(b) Performance on Amazon2M

Figure A1: Subgroup-level comparisons between base models and adapted local models on Arxiv1M (a) and Amazon2M (b).



(a) Influence of subgroup number M on Ada-GNN

(b) Influence of adaption steps K on Ada-GNN

Figure A2: The influence of subgroup numbers M (a) and adaption steps K (b) over Amazon2M. For conciseness we only demonstrate two base models: SGC and SIGN.

A.3 Hyper-parameters Analysis

We conduct hyper-parameters experiments on Arxiv1M and Amazon2M to analyze the impact of subgroup number M and adaption step K. Here we provide the results on Amazon2M in Figure A2. Readers can find detailed experimental analysis in Section 4.6.