Against Multifaceted Graph Heterogeneity via Asymmetric Federated Prompt Learning [Scalable Data Science]

Zhuoning Guo
zguo772@connect.hkust-gz.edu.cn
The Hong Kong University of Science
and Technology (Guangzhou)

Ruiqian Han
rhan464@connect.hkust-gz.edu.cn
The Hong Kong University of Science
and Technology (Guangzhou)

Hao Liu
liuh@ust.hk
The Hong Kong University of Science
and Technology (Guangzhou)
The Hong Kong University of Science
and Technology

ABSTRACT

Federated Graph Learning (FGL) aims to collaboratively and privately optimize graph models on divergent data for different tasks. A critical challenge in FGL is to enable effective yet efficient federated optimization against multifaceted graph heterogeneity to enhance mutual performance. However, existing FGL works primarily address graph data heterogeneity and perform incapable of graph task heterogeneity. To address the challenge, we propose a Federated Graph Prompt Learning (FedGPL) framework to efficiently enable prompt-based asymmetric graph knowledge transfer between multifaceted heterogeneous federated participants. Generally, we establish a split federated framework to preserve universal and domain-specific graph knowledge, respectively. Moreover, we develop two algorithms to eliminate task and data heterogeneity for advanced federated knowledge preservation. First, a Hierarchical Directed Transfer Aggregator (HiDTA) delivers cross-task beneficial knowledge that is hierarchically distilled according to the directional transferability. Second, a Virtual Prompt Graph (VPG) adaptively generates graph structures to enhance data utility by distinguishing dominant subgraphs and neutralizing redundant ones. We conduct theoretical analyses and extensive experiments to demonstrate the significant accuracy and efficiency effectiveness of FedGPL against multifaceted graph heterogeneity compared to state-of-the-art baselines on large-scale federated graph datasets.

1 INTRODUCTION

Federated Graph Learning (FGL) has become an attractive research direction for distributed Graph Neural Network (GNN) optimization on isolated graph data without explicit information exposure [20].

Specifically, in FGL, several participants maintain their private graphs collected from different domains, and they aim to collaboratively train GNNs that can provide beneficial knowledge of raw graph data for particular downstream tasks, respectively.

Recent FGL studies have achieved significant progress mainly in learning more effective graph representation to enhance downstream task prediction [11]. For example, FedSage+ [43] is proposed based on a standard FGL pipeline where subgraphs from different silos are independently distributed, and improve the node classification via missing edge generation. Despite these advancements, the difficulty of graph data heterogeneity remains paramount. Graphs inherently differ in terms of node and edge types, structural configurations, and other characteristics. Traditional graph learning methodologies often operate under the assumption of homogeneity, which can lead to biased or suboptimal models when applied to heterogeneous graph domains. Addressing graph data heterogeneity is crucial for FGL, as it enables the model to capture diverse intrinsic graph characteristics and enhances overall performance. Therefore, FedStar [29] investigates the data heterogeneity of graph features and shared knowledge, which attempts to achieve outperformance in a Non-Independent and Non-Identically Distributed (Non-IID) issue setting. In addition, FedLIT [38] detects link-type heterogeneity to decrease the harmful structure to allow more beneficial message passing. Generally, these works designed effective methods to tackle the graph data heterogeneity in terms of features and structures, where they demand participants to train their models for a consistent downstream task.

However, while attempts have been made to address data heterogeneity, the critical issue of task heterogeneity remains unexplored. FGL participants originate from various sectors, each with its own set of tasks and data characteristics. For instance, healthcare providers may focus on patient data analysis, while financial institutions might prioritize fraud detection. Addressing task heterogeneity allows FGL to adapt to these varied needs, enhancing its applicability and relevance across different applications. Existing federated algorithms, which typically assume a uniform model structure across clients, cannot accommodate the need for diverse architectures in graph learning [28]. This limitation complicates parameters and knowledge sharing between clients working on different tasks, potentially causing inferior effectiveness and efficiency in collaborative optimization. To effectively optimize models against multifaceted heterogeneity, we necessitate a system not only to preserve common knowledge adaptable to various tasks but

also to generalize across divergent graph data distributions. Therefore, the core challenge of this work is to simultaneously overcome the multifaceted heterogeneity for more effective and efficient FGL.

To address the challenge, we propose a **Federated Graph Prompt Learning (FedGPL)** framework to federally fine-tune graph models on heterogeneous tasks and data via an efficient prompt-based asymmetric knowledge transfer. Overall, we establish a split framework to simultaneously achieve a universal graph representing and personalized graph prompting to preserve global and local knowledge, respectively. Then, we design tailored algorithms to disentangle and address task and data heterogeneity. On the server side, we develop a Hierarchical Directed Transfer Aggregator (HiDTA) to extract and share asymmetrically beneficial knowledge among task-heterogeneous participants via personalized federated aggregation. On the client side, we devise a lightweight prompting module, called Virtual Prompt Graph (VPG), to adaptively generate augmented graph data by distilling more dominant information with minor data heterogeneity. We provide theoretical analyses that demonstrate the effectiveness in reducing task and data heterogeneity, as well as the significant reduction of memory and communication costs. Extensive experiments validate that FedGPL outperforms baseline methods across three levels of tasks against multifaceted graph heterogeneity on five datasets. Notably, we evaluate FedGPL in a typical large-scale FGL system consisting of 1 million node data. Our method achieves $5.3\times \sim 6.0\times$ GPU memory efficiency, $2.1\times \sim 3.7\times$ communication efficiency, and $1.3\times \sim 1.9\times$ training time efficiency. The efficiency superiority demonstrates its scalability for massive FGL participants and large-scale graph data.

Our main contributions can be concluded as: (1) To our knowledge, our work is the first to study both task and data heterogeneity in FGL, addressing gaps overlooked by previous research. (2) We propose a federated graph prompt learning framework to effectively enable federated optimization of personalized models among task- and data- heterogeneous participants. (3) We develop an aggregation algorithm to deliver task-specific knowledge based on a transferability-aware hierarchy to asymmetrically enhance model performance. Moreover, we devise a virtual graph prompt to jointly highlight dominant graph structures and alleviate extensive data heterogeneity. (4) We theoretically analyze the mitigation of multifaceted heterogeneity by FedGPL. Besides, we conduct extensive experiments to prove the accuracy and efficiency superiority of FedGPL against FGL with multifaceted heterogeneity on federated graph datasets with million nodes.

2 PRELIMINARIES

2.1 Notations and Problem Statement

2.1.1 Graph notations.
Graph is a common data structure representing a set of objects and their relationships. We denote a graph as $G = (V, E, X)$. Here $V = \{v_1, v_2, \cdots, v_{N_n}\}$ is the node set with N_n nodes. $E = \{e_1, e_2, \cdots, e_{N_e}\}$ is the edge set with N_e edges that $e_i = (v_{s_i}, v_{t_i})$ is an undirected edge from node v_{s_i} to node v_{t_i}. $X \in \mathbb{R}^{N_n \times d_0}$ is the node feature matrix that i-th row vector $x_i \in \mathbb{R}^{d_0}$ is the feature of v_i, where d_0 is the node feature dimension. For graph learning, researchers usually treat three levels (*i.e.*, node-level, edge-level, and graph-level) of tasks in different pipelines [27]. We formulate three levels of tasks in Appendix A.1.

2.1.2 Multifaceted heterogeneity.
This paper investigates an FGL scenario consisting of multifaceted heterogeneity in terms of task and data heterogeneity, which is defined below.

DEFINITION 1. **Task Heterogeneity.** *In FGL, task heterogeneity arises when participants aim to optimize models for different downstream tasks, including node-, edge-, and graph- level tasks. Therefore, task heterogeneity will be reflected by the difference of model structures and parameters. Specifically, we measure the heterogeneity $\Delta_T^{a,b}(\theta_a, \theta_b) \in [0, 1]$ between a-th and b-th tasks as*

$$
\Delta_T^{a,b}(\theta_a, \theta_b) = \begin{cases} 1, & t_a \neq t_b, \\ \frac{1 - \exp(-\|\theta_a - \theta_b\|_2)}{1 + \exp(-\|\theta_a - \theta_b\|_2)}, & t_a = t_b, \end{cases} \tag{1}
$$

where θ_a, θ_b denotes the model parameters for the a -th and b -th tasks, $\overline{\theta}$ means the average values of θ.

DEFINITION 2. **Data Heterogeneity.** *In FGL, data heterogeneity appears when participants keep divergent graph data generated from their domains. The data heterogeneity between graphs is reflected in the difference of their structures and features. Graph representations can incorporate graph structures and features in a high-dimensional vector. We utilize a popular graph readout function [39] to produce graph representations as $H = \{h_1, \cdots, h_n\} = \mathcal{G}(V, E, X)$ and $h_G = \tilde{H} = \frac{1}{n} \sum_n^{i=1} h_i$, where n is the number of nodes of G and $\mathcal{G}$ is a trained GNN to represent nodes. Hence, we measure the data heterogeneity $\Delta_D^{i,j}(h_{G_i}, h_{G_j}) \in [0, 1)$ between graphs G_i and G_j according to the distance of their representations h_{G_i} and h_{G_j} as*

$$
\Delta_D^{i,j}(h_{G_i}, h_{G_j}) = \frac{1 - \exp(\overline{(h_{G_i} - h_{G_j})^2})}{1 + \exp(\overline{(h_{G_i} - h_{G_j})^2})}, \tag{2}
$$

where $\overline{(h_{G_i} - h_{G_j})^2}$ is the average value among d dimensions of $(h_{G_i} - h_{G_j})^2$.

2.1.3 Problem formulation.

PROBLEM 1. **Federated Graph Learning against Multifaceted Heterogeneity.** *An FGL system consists of a server S and N clients $\{C_1, C_2, \cdots, C_N\}$ corresponding to each participant. Each client aims to learn a model $\mathcal{F}_i(\cdot)$ on its dataset D_i, where $t_i \in \{node, edge, graph\}$ and $1 \leq i \leq N$. Models and datasets may be heterogeneous among clients. We aim to optimize the parameters of all models to minimize the average loss as*

$$
\{\theta_1, \cdots, \theta_N\} = \arg\min \sum_N^{i=1} \mathcal{L}_i(\mathcal{F}_i(D_i; \theta_i)), \tag{3}
$$

where $\mathcal{L}_i(\cdot)$ denotes the loss function of the prediction model $\mathcal{F}_i(\cdot)$ parameterized with θ_i on dataset D_i.

2.1.4 Threat model.
We assume all participants are semi-honest, strictly following the Federated Learning (FL) protocol, and only knowing about the knowledge they already keep or receive from others during interactions. We will clarify the objectives of participants, including clients and server: The client aims to obtain the structure and parameters of GNN from the server; the server aims to see the private data of the client.
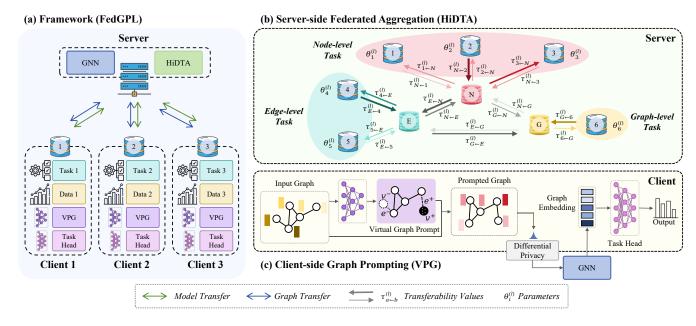
(a) Framework (FedGPL)

(b) Server-side Federated Aggregation (HiDTA)

(c) Client-side Graph Prompting (VPG)

Model Transfer ⟷ *Graph Transfer* ⟷ $\tau_{a \leftarrow b}^{(l)}$ *Transferability Values* $\theta_i^{(l)}$ *Parameters*

Figure 1: An overview of Federated Graph Prompt Learning (FedGPL) framework.

2.2 Graph Prompt Learning

We formulates the pipeline of Graph Prompt Learning (GPL), which can enable effective optimization for downstream tasks to inherit knowledge of Graph Neural Network (GNN) and fine-tune on specific domains.

We first follow the existing work [27] to reformulate node-level and edge-level tasks as graph-level tasks to train three tasks in a consistent workflow illustrated in Appendix A.2. Then, GNN aims to produce representations by encoding input graphs. Existing works pre-train GNNs on large-scale graph data for efficient fine-tuning and knowledge reusing [25, 37, 45]. In this work, we leverage pre-trained GNNs in GPL for better performance. A GNN $\mathcal{G}$ reads an input graph $G = (V, E, X)$ and outputs representations H as $\mathcal{G}(G) = H \in \mathbb{R}^{N_n \times d}$, where d is the dimension.

Specifically, we can insert a prompt into a graph from a source domain, *i.e.*, *source graph*, to build a prompted graph, *i.e.*, *target graph*. GPL follows three steps [27]: *(1) Graph prompting*: A graph prompting function $f_p(\cdot|\cdot)$ is leveraged to transform a source graph G into a target graph $\tilde{G}$ as $\tilde{G} = f_p(G|P_G)$ where P_G is a specific prompt for G. *(2) Graph representing*: The GNN $\mathcal{G}$ produces a representation H of the target graph as $H = \mathcal{G}(\tilde{G})$. *(3) Task prediction*: A task head $f_h(\cdot)$ inputs H to predict the label y of the downstream task as $y = f_h(H)$. Usually, the task head is learnable for more accurate downstream prediction. For example, the task head can be a linear classifier [27].

3 METHODOLOGY

The sec will illustrate the framework, algorithms, and training. Specifically, we first propose the FedGPL framework to splitly enable privacy-preserving and personalized graph prompt tuning. Then, we develop two algorithms, including a hierarchical directed transfer aggregator and a virtual prompt graph, to eliminate task

and data heterogeneity, respectively. Last, we present the training workflow based on algorithms tailored for the FedGPL framework.

3.1 Framework

For an FGL system with both task and data heterogeneity, learning a parameter-shared model for all participants as traditional federated learning is not applicable. Because it may cause poor generalization ability on different tasks and divergent data. Along with this idea, we propose a framework, namely **Federated Graph Prompt Learning (FedGPL)**, to splitly preserve global and local knowledge by prompt-based privacy-preserving fine-tuning, illustrated in Figure 1. In light of Split Federated Learning [32], the common and domain-specific graph knowledge can be captured separately, allowing global generalizability and local personalization.

(a) Architecture. Overall, FedGPL consists of a server and several clients corresponding to participants. First, we follow the GPL formulated in Section 2.2 to unify learning procedure of three levels of tasks. Second, we deploy a trainable GNN in the server to deliver the graph representing ability for heterogeneous clients. The GNN contains versatile graph knowledge, which can be efficiently fine-tuned to fit multiple domains in a federated system. Third, a task head and a graph prompt are personally optimized within each client for domain-specific understanding of its task and data.

(b) Collaborative learning procedure. Based on the GPL pipeline, we describe how to activate forward- and backward- propagation with collaboration between the server and clients via communication. The communication messages between modules include four elements: featured prompted graphs and their representation in forward-propagation, and gradients of the task head and the graph prompt in back-propagation. On the one hand, the forward-propagation procedure follows: (1) A client prompts an input graph and sends it to the server. (2) The server generates the representation of the received prompted graph by GNN and then sends it back

to the client. (3) The client takes the prompted graph representation as input of the task head for prediction. On the other hand, the back-propagation procedure follows: (1) A client computes gradients of the task head and sends them to the server. (2) The server computes gradients of the prompted graph and sends it back to the client. (3) The client computes the gradients of the graph prompt. After two-sided propagation, GPL modules can be optimized.

(c) Differentially privacy preservation. The communication messages remain privacy leakage risks. Specifically, the server may obtain the raw features of prompted graphs from clients by attacks such as DLG [46]. Therefore, to further protect the client-side data privacy, we utilize a Differential Privacy (DP) technique, Laplacian noise, to add it to embeddings and gradients as

$$f(X; \epsilon) = X + \eta, \eta \sim \frac{1}{\lambda}\exp(-\frac{|x|}{\lambda}), \lambda = \frac{\mathcal{V}(X)}{\epsilon}, \quad (4)$$

where X is the embedding or gradient to be protected, $\mathcal{V}(X)$ determines the variation of the protected data, and ϵ is the privacy scale parameter.

In summary, based on the split strategy and DP techniques, we build a privacy-preserving federated framework to allow GPL to prevent private data from being leaked or inferred. Besides, state-of-the-art pre-trained GNNs such as GROVER [25] and DVMP [45] have approximately 100 million parameters. Our split strategy makes clients without heavy memory burden to store GNNs, especially when applying FedGPL among resource-constrained devices [4].

3.2 Algorithms

The existing federated algorithms [14, 19] developed on FedAvg [23] can only be applied in traditional non-IID settings. However, they still fail in FGL with task and data heterogeneity. Because these algorithms strictly require a consistent model structure across clients. The requirement is infeasible in a task-heterogeneous federation that is more general in FGL. How to tackle the joint heterogeneity in this scenario for more effective federated optimization remains a significant difficulty.

Therefore, we propose an aggregating and a graph prompting algorithms to eliminate damage from graph task and data heterogeneity by a privacy-preserving and collaborative optimization. Specifically, we first devise a federated aggregation algorithm in the server side to transfer asymmetrically beneficial knowledge between different tasks. Moreover, we equip each client with an graph prompt to retain useful local knowledge with less data divergence.

3.2.1 Server-side federated aggregation.
Previous methods such as FedAvg mostly aggregate local models based on an assumption that the heterogeneity between clients affects symmetrically. However, cross-task knowledge transfer is asymmetric, which means that the effectiveness of transferring a task to another may differ from the inverse. In fact, measuring the optimal directed knowledge transfer and activating it with learning benefits are both non-trivial problems in task-heterogeneous FGL. Therefore, we develop the **Hierarchical Directed Transfer Aggregator (HiDTA)** to federally enhance prediction performance of task-heterogeneous clients via directional knowledge transfer that is hierarchically determined by a proposed metric for transferring availability.

(1) Transferability Definition: a metric for knowledge transfer availability. First, we define a metric, *i.e.*, transferability, to model the directed heterogeneity between clients in Definition 3. For a source client and a target client from their corresponding clients, transferability reflects *how much benefit of the target client if it uses the source client's model on the target client's data.*

DEFINITION 3. **Transferability.** *We define the transferability* $\tau_{a \leftarrow b}$ *from model a to model b as*

$$\tau_{a \leftarrow b}^{(l)} = \frac{\overrightarrow{\theta_a^{(l)\prime} - \theta_a^{(l)}} \cdot \overrightarrow{\theta_b^{(l)\prime} - \theta_a^{(l)}}}{\|\overrightarrow{\theta_a^{(l)\prime} - \theta_a^{(l)}}\|}, \quad (5)$$

where θ_a is the parameters of a model, $\theta^{(l)}$ is the parameters at l-th optimization step, $\theta^{(l)\prime}$ is the estimated optimal parameters computed on local data for $\theta^{(l)}$ at the next step.

Transferability fits the property that knowledge transferring is often asymmetric, *i.e.*, $\tau_{i \leftarrow j} \neq \tau_{j \leftarrow i}$, which means that it is more convincing to measure the transferability by a directed value.

(2) Hierarchical transferability evaluation. Next, we propose to evaluate pairwise transferability to understand the task heterogeneity based on the decoupled hierarchy of clients. Specifically, we construct task-models to identify task-specific knowledge by fused perception on corresponding models. For each task, we measure the multi-dimensional distances of parameters between client-models and task-models. Based on task-models and calculated distances, we extrapolate the transferability for any two client-models by

$$\tau_{a \leftarrow b}^{(l)} = (1 - \sigma(\|\theta_a^{(l)} - \theta_{p_a}^{(l)}\|)) \cdot (1 - \sigma(\|\theta_b^{(l)} - \theta_{p_b}^{(l)}\|)) \cdot \tau_{p_a \leftarrow p_b}^{(l)}, \quad (6)$$

where a and b are two client-models, p_a and p_b are a and b's task-models, and $\sigma(\cdot)$ is a normalization function (*e.g.*, Sigmoid$(\cdot)$ or ReLU$(\cdot)$).

(3) Asymmetric knowledge transfer via model aggregation. Last, we weighted aggregate models to transfer knowledge from source clients to target clients based on normalized transferability values. The model parameters of a client i can be compute as

$$\theta_i^{(l+1)} = \frac{\sum_N^{j=1} (\tau_{i \leftarrow j}^{(l)} \cdot \theta_j^{(l)})}{\sum_N^{j=1} \tau_{i \leftarrow j}^{(l)}}, \quad (7)$$

where $\theta_i^{(l+1)} = (\theta_{p,i}^{(l+1)}, \theta_{h,i}^{(l+1)})$ is the parameter set of $f_p(\cdot)$ and $f_h(\cdot|\cdot)$ of client i, and N is the number of participating clients.

In short, HiDTA assists positive knowledge transfer across clients and impedes harmful attacks aroused by task heterogeneity, which are both extracted based on the hierarchical transferability assessment. The success of tackling task heterogeneity will be theoretically analyzed in Section 4.1.

3.2.2 Client-side graph prompting.
Existing works have proposed to construct universal graph prompts for any level of tasks by incorporating task-specific properties [7, 27, 47]. However, there is an underexplored conflict between distilling specific knowledge and eliminating corresponding data heterogeneity. On the one hand, prompting with more task-specific information on graphs brings more severe data heterogeneity. On another hand, extracting more unique patterns requires more distinguished prompted features. Besides, existing graph prompting strategies remain inefficient due

to heavy models carrying prior knowledge, which is unsuitable for large-scale FGL scenarios. Therefore, we devise **Virtual Prompt Graph (VPG)**, to model task-specific knowledge as well as minimize data heterogeneity by adaptively indicating intellective graph structures.

(1) Prompting module construction. First, we insert a positive structure to capture distinctive information in the particular domain. Second, we construct a negative structure to neutralize noisy nodes and edges in terms of relatively useful graph structures. Simultaneously, VPG can assist the GNN to generate more effective yet less heterogeneous graph representations for tailored clients. Then, we define VPG in Definition 4, which implies how we can transform a *source graph*, *i.e.*, the original graph, into a *target graph*, *i.e.*, the prompted graph, to be input into the GNN.

Definition 4. **VPG.** *For a graph $G = (V, E, X)$, we denote a VPG as $\hat{G} = ((\hat{V}^+, \hat{V}^-), (\hat{E}^+, \hat{E}^-), (\hat{X}^+, \hat{X}^-))$. It is satisfied that $\hat{V}^+ \cap \hat{V}^- = \emptyset$, where $\hat{V}^+ = \{v_{N_n+1}^+, v_{N_n+2}^+, \cdots\}$ is the prompted node set and $\hat{V}^- = \{v_{i_1}^-, v_{i_2}^-, \cdots\}, v_{i_1}, v_{i_2}, \cdots \in V$ is the prompted anti-node set. And it is satisfied that $\hat{E}^+ \cap \hat{E}^- = \emptyset$, where $\hat{E}^+ = \{e_{N_e+1}^+, e_{N_e+2}^+, \cdots\}$ is the prompted edge set and $\hat{E}^- = \{e_{j_1}^-, e_{j_2}^-, \cdots\}, e_{j_1}, e_{j_2}, \cdots \in E$ is the prompted anti-edge set.*

(2) Learning for adaptive prompting on graphs. We define VPG-based prompt function as $\tilde{G} = f_p(G|\hat{G})$, where $f_p(\cdot|\cdot)$ follows $\tilde{V} = V \cup \hat{V}^+ \setminus \hat{V}^-, \tilde{E} = E \cup \hat{E}^+ \setminus \hat{E}^-, \tilde{X} = \{x_i | v_i \in \tilde{V}\}$. Then, we propose a generative algorithm to build specific VPGs. Specifically, we construct a candidate set of k' nodes, whose features are predefined (default) or learnable. We connect them to the source graph if v_i and v_j satisfies $\text{Sigmoid}(x_i \cdot x_j) < \gamma$, where γ is a threshold. After that, we have $G' = (V', E', X')$. We generate node significance as $w_n' = X' \cdot p'/||p'||$, where $p' \in \mathbb{R}^d$ is a learnable vector. We also generate edge significance between i-th and j-th node as $w_{e,i,j}' = ||w_{n,i}' - w_{n,j}'||$. The node features will be also processed by $x_i = x_i \cdot \tanh(w_{n,i}')$. Next, we determine the node significance borderline $\bar{w}_n'$ as the k_n-th highest significance score among V', and the edge significance borderline $\bar{w}_e'$ in a similar way. Note that $k_n = \alpha_n \cdot |V|$ and $k_e = \alpha_e \cdot |E|$ and we manually decide α_n and α_e. Last, we can create a VPG $\hat{G}$ by constructing the prompted (anti-) node set and (anti-) edge set, by letting $\hat{V}^+ = V_{w_{n,i}' \geq \bar{w}_n'}' \setminus V$, $\hat{V}^- = V_{w_{n,i}' < \bar{w}_n'}, \hat{E}^+ = E_{w_{e,i,j}' \geq \bar{w}_e', w_{n,i}' \geq \bar{w}_n', w_{n,j}' \geq \bar{w}_n'}' \setminus E$, and $\hat{E}^- = E_{w_{e,i,j}' < \bar{w}_e' | w_{n,i}' < \bar{w}_n' | w_{n,j}' < \bar{w}_n'}$.

In conclusion, VPG indicates useful and useless graph structures to simplify local graph data to reduce data heterogeneity with better representation production, which will be discussed in Section 4.2.

3.3 Training

We present the training workflow of FedGPL. First, we allow clients to independently optimize their kept modules including prompting and task head modules. Second, the server-side GNN can be optionally optimized. It can be pre-trained or randomly initialized. After that, its parameters will be fine-tuned or frozen during optimization. In this work, we choose to fine-tune a pre-trained GNN by default, because pre-training and fine-tuning GNNs can efficiently utilize common graph knowledge and fit on specific domains.

Detailedly, we take the node classification task as an example in Appendix A.4. Specifically, we first initialize the client-side parameters. For each training round, we enable each client in parallel to individually train their modules. The client cooperates with the server for split and privatized forward- and back- propagation. Then the server globally calculates transferability between clients and coordinates them to locally update parameters. The training process will terminate after parameters in clients are converged.

4 THEORETICAL ANALYSIS

This part investigates FedGPL about (1) the reduction of task heterogeneity, (2) the elimination of data heterogeneity, and (3) the complexity of memory and communication.

4.1 Task Heterogeneity Analysis

We analyze the effect of our aggregation algorithm, HiDTA, on task heterogeneity by quantifying the heterogeneity values based on Definition 1. We theoretically propose Theorem 1 to demonstrate the success of HiDTA in eliminating task heterogeneity, which is proved in Appendix A.5.

Theorem 1. *For a -th and b -th clients in FedGPL, we denote their estimated optimized parameters at l-th step as $\theta_a^{(l+1)'}, \theta_b^{(l+1)'}$ (aggregated) or $\theta_a^{(l)'}, \theta_b^{(l)'}$ (non-aggregated). It is satisfied that*

$$\Delta_T^{a,b}(\theta_a^{(l+1)'}, \theta_b^{(l+1)'}) \leq \Delta_T^{a,b}(\theta_a^{(l)'}, \theta_b^{(l)'}), \tag{8}$$

when their bidirectional transferability are positive. Hence, the task heterogeneity can be reduced by HiDTA.

4.2 Data Heterogeneity Analysis

We quantify data heterogeneity influenced by VPG based on Definition 2. We compute the theoretical values before and after prompting and propose Theorem 2 which can be proved by Appendix A.6.

Theorem 2. *For the a -th and b -th clients in FedGPL, graph representations from two clients are $h_{\tilde{G}_a}, h_{\tilde{G}_b}$ and $h_{\tilde{G}_a}, h_{\tilde{G}_b}$ for with and without prompting methods, respectively. And their graph data following uniform distributions of $h_{G_a} \sim U^a[0, \eta^a]$ and $h_{G_b} \sim U^b[0, \eta^b]$, respectively. We ensure*

$$\mathbb{E}_{\begin{cases} h_{\tilde{G}_a} \sim \tilde{U}^a \\ h_{\tilde{G}_b} \sim \tilde{U}^b \end{cases}} [\Delta_D^{i,j}(h_{\tilde{G}_a}, h_{\tilde{G}_b})] \leq \mathbb{E}_{\begin{cases} h_{G_a} \sim U^a \\ h_{G_b} \sim U^b \end{cases}} [\Delta_D^{i,j}(h_{G_a}, h_{G_b})], \tag{9}$$

when $\alpha_n^a \approx \alpha_n^b$. Thus, the data heterogeneity can be reduced by VPG.

4.3 Efficiency Analysis

4.3.1 Memory efficiency. We assume the input graph has N nodes and M edges, the GNN has L layers with the maximum dimension of a layer being h, and the task head has l layers. Here we compute the theoretical storage complexity. The client has to store the whole GNN in the usual federated prompting setting. Hence, we take a popular GNN variant, GAT [34], as an example of the GNN, the storage complexity will be $O(LKh^2 + LKh + Kh)$. By applying our split strategy, each client only needs to store the prompt and a task head layer, where the storage complexity is $O(h + lh)$.

Table 1: ACC (%) and F1 (%) of different federated algorithms and graph prompting methods. We fine-tune all GPL modules including a pre-trained GraphTransformer [40], graph prompts, and task heads.

Federated Method	Prompt Method	Cora		CiteSeer		DBLP		Photo		Physics	
		ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Local	GPF	85.02	84.37	84.60	84.18	81.03	80.70	76.85	76.46	88.29	88.49
	ProG	86.45	85.91	85.72	85.85	82.20	82.01	71.47	70.47	86.73	86.98
	SUPT	85.66	85.23	87.00	86.88	80.57	80.52	73.31	72.87	89.69	89.56
FedAvg	GPF	85.20	84.76	85.36	85.22	81.87	81.88	80.73	80.52	89.75	88.83
	ProG	86.70	86.13	86.49	86.05	82.43	82.21	72.21	71.76	89.05	89.27
	SUPT	85.74	85.23	87.17	84.05	83.07	82.52	81.70	81.44	89.79	88.60
FedProx	GPF	80.52	79.74	83.20	82.69	80.57	79.84	82.93	82.79	87.60	87.19
	ProG	84.04	83.79	84.39	83.87	81.86	81.55	72.45	72.46	89.01	88.57
	SUPT	81.81	80.59	83.47	83.82	80.79	80.80	80.26	80.64	89.21	88.21
SCAFFOLD	GPF	77.08	76.39	75.79	74.87	78.57	77.63	74.43	74.37	85.20	84.85
	ProG	73.90	73.25	74.80	74.57	76.99	76.25	72.36	71.72	85.60	85.27
	SUPT	75.38	70.47	74.91	73.73	75.80	75.71	75.65	75.86	85.53	84.49
FedGPL		**88.34**	**88.07**	**87.60**	**87.72**	**83.93**	**83.27**	**86.75**	**86.03**	**90.91**	**90.61**

4.3.2 Communication efficiency. The communication includes two parts in FedGPL that are between clients and the server (depending on the graph and embedding sizes), and between clients and the server (depending on the prompt and task head sizes). Theoretically, VPG appends k' candidate nodes into the input graph, then score by a learnable vector p', the final prompted graph contains k_n nodes, as the candidate nodes are pre-defined, the parameter complexity is $O(h)$, if we set the candidate nodes as trainable, the parameter complexity is $O(k'h + h)$. Thus, VPG contains minimum d parameters. In addition, we involve two baseline GPL methods for comparison, including GPF (parameters: $O(nh)$) and ProG (parameters: $O(kh)$).

5 EXPERIMENTS

We conduct experiments to evaluate our methods by answering the following questions in particular: **Q1**: How effective is the overall prediction performance of FedGPL and each module? **Q2**: How useful is FedGPL against task and data heterogeneity, respectively? **Q3**: How efficient is our method in terms of memory, communication, and time on large-scale data? Our codes, data, and appendix are open in https://anonymous.4open.science/r/FedGPL/.

5.1 Experimental Setup

5.1.1 Datasets. We conduct experiments on five datasets including Cora [3], CiteSeer [3], DBLP [2], Photo [26], and Physics [26]. We follow [27] for constructing datasets of node-, edge- and graph- level tasks. Each data sample is a graph with about 300 nodes. Usually, we can generate more than 400 samples for each client. We deploy multiple clients (3 by default) for each level of task in the following experiments, all clients participate in each round of training, and each client conducts local training once per round. The detailed statistic of the original graph datasets is listed in Appendix A.7.

5.1.2 Baselines. We combine the state-of-the-art GPL and FL methods to construct competitive baselines. First, we select three GPL methods available for all three levels of graph tasks, including (1) ProG [27], which insert an extra graph as a prompt into the input graph, (2) GPF [7], which inserts extra vectors as prompts
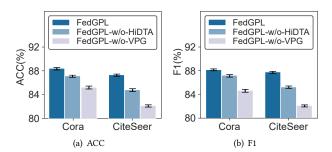
into the nodes' feature, and (3) SUPT [17], which prompt features in the subgraph. Second, we select three FL methods, including FedAvg [23], FedProx [19], and SCAFFOLD [14] to aggregate corresponding GPL modules.

5.1.3 Evaluation metrics. We use accuracy (ACC) and F1 Score (F1) to evaluate the prediction performance of tested methods on the node classification task.

5.2 RQ1: Federated Prompt Tuning Performance

5.2.1 Overall results. Here we compare FedGPL with baselines by averaging the performance of clients. Their performance is evaluated according to prediction ACC and F1 on node classification tasks on five datasets. First, we train all modules in these methods, including the pre-trained GNN, graph prompts, and task heads. As shown in Table 1, FedGPL consistently outperforms all other algorithms, achieving an improvement range of 2.37% to 16.07% over state-of-the-art methods by jointly incorporating HiDTA and VPG to tackle multifaceted heterogeneity.

5.2.2 Ablation study. To evaluate the effectiveness of FedGPL modules, we conduct an ablation study with two variations. *FedGPL-w/o-HiDTA* refers to a variant that excludes the HiDTA module, while *FedGPL-w/o-VPG* denotes another variant without the VPG module. Figure 2 shows significant performance drops when HiDTA or

Figure 2: Ablation study.

Table 2: ACC (%) of federated algorithms under different task combinations on Cora. Tasks in the federation are `blue`. N, E, and G: node-, edge-, and graph-level task. N+E: federation between node- and edge- level tasks. Global: the average of three tasks.

Method	N+E				N+G				E+G			
	N	E	G	Global	N	E	G	Global	N	E	G	Global
FedAvg	79.02	89.34	91.02	86.46	79.09	90.35	90.02	86.49	79.19	90.13	91.14	86.82
FedProx	73.96	89.58	91.02	84.85	79.15	90.35	91.11	86.87	79.19	91.52	91.14	87.28
SCAFFOLD	71.21	89.67	91.02	83.97	72.53	90.35	80.33	81.07	79.19	90.21	80.68	83.36
HiDTA	79.73	90.89	91.02	87.21	80.44	90.35	**92.56**	87.78	79.19	**92.13**	**92.75**	88.02
	80.80	**91.76**	**92.12**	**88.34**	**80.80**	**91.76**	92.12	**88.34**	**80.80**	91.76	92.12	**88.34**

Table 3: Performance of different federated algorithms against data heterogeneity. Minor α means higher data heterogeneity.

Federated Method	$\alpha = 0.9$				$\alpha = 0.6$				$\alpha = 0.3$				$\alpha = 0.1$			
	Cora		CiteSeer		Cora		CiteSeer		Cora		CiteSeer		Cora		CiteSeer	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
FedAvg	86.93	86.40	86.93	86.42	87.77	87.02	86.28	85.87	85.00	84.59	83.87	83.62	83.43	83.39	82.13	81.92
FedProx	85.00	84.45	85.71	85.57	84.07	83.63	85.77	85.29	82.43	82.28	83.09	82.90	81.90	81.35	81.57	81.20
SCAFFOLD	71.67	71.87	73.07	72.59	74.17	73.60	73.07	72.60	71.65	71.90	73.12	73.07	72.87	72.28	71.94	70.64
FedGPL	**88.23**	**87.83**	**87.33**	**87.39**	**87.90**	**87.42**	**86.93**	**86.69**	**85.07**	**84.07**	**82.77**	**82.60**	**83.93**	**83.72**	**82.29**	**82.23**

VPG is removed, confirming their effectiveness in our framework. Specifically, the ACC of FedGPL-w/o-HiDTA decreases by 1.04% on Cora, and by 1.55% on CiteSeer. We explain that HiDTA facilitates adaptive aggregation that enhances cross-client knowledge sharing, resulting in performance improvements compared to isolated training. Additionally, ACC of FedGPL-w/o-VPG decreases by 3.31% on Cora, and 3.62% on CiteSeer, indicating that VPG allows transformed input graphs to be well understood by the GNN. We also test the statistical significance of prediction performance by varying random seeds. For example, for FedGPL, ACC is 0.8817 ± 0.1650 and F1 is 0.8772 ± 0.1672 on Cora.

5.3 RQ2: Effectiveness of Tackling Graph Task and Data Heterogeneity

We evaluate the effectiveness against heterogeneous graph tasks and data under the FedGPL framework according to the prediction ACC tested on varied settings.

5.3.1 Task heterogeneity. We construct three settings with heterogeneous tasks, which only involve two tasks in federated optimization and another one is trained locally. We present the results in Table 2. On the one hand, compared with baseline aggregation algorithms, HiDTA performs better under multiple task-heterogeneous scenarios. For example, the ACC of HiDTA improves at least 1.63% in node-level tasks and 1.59% in graph-level tasks when they are federally optimized. On the other hand, HiDTA achieves a higher average ACC with a federation among three tasks than two tasks. For example, the three-task federation gains 1.10% enhancement compared with a federation between node-level tasks and edge-level tasks. Besides, a special case is that edge-level tasks and graph-level tasks perform slightly better in their federation than the ones involving node-level tasks. On the contrary, node-level tasks obtain

1.97% improvement through participation. We observe a positive trade-off between tasks that although two levels of tasks drop their ACC, they significantly enhance node-level ones to reach a better overall ACC. The finding shows that HiDTA optimizes models by centering the global performance.

5.3.2 Data heterogeneity. We vary the degree of data heterogeneity by following an existing work [41] to simulate non-IID scenarios with a α-Dirichlet distribution. As α decreases, clients' data becomes more non-IID. We report results in Table 3. When graph data tend to be divergent, the ACC is reduced. In comparison, FedGPL outperforms baselines under different non-IID degrees. For example, on Cora, FedGPL improves over 1.35%, 0.38%, 0.35% F1 of baselines when $\alpha = 0.9, 0.6$, and 0.3, respectively. We conclude that VPG can eliminate extensive graph data heterogeneity by highlighting important graph structures while reducing useless subgraphs.

5.4 RQ3: Efficiency Evaluation

5.4.1 Memory efficiency. We calculate the number of stored parameters to evaluate the client-side memory burden. Specifically, our default setting demands 81, 600 parameters in each client. Our method requires each client only to store 800 parameters, reducing 99.02% memory space. In practice, Table 4 shows that FedGPL achieves $5.3\times \sim 6.0\times$ GPU memory efficiency in the training environment compared to previous work, which implies the potential to adopt FedGPL on resource-constrained settings.

5.4.2 Communication efficiency. Here we compute and compare the communication cost of four methods, including (1) FedAvg+GPF, (2) FedAvg+ProG, (3) FedGPL, and (4) FedGPL*: a variant of FedGPL incorporating a VPG with learnable V^+ features. Experimental results are listed in Table 5. Specifically, FedGPL and FedGPL* reduce over 70.83% communication costs compared with FedAvg+GPF, and

Table 4: Training time (200 epochs) and memory cost (GPU memory usage per client) on the Physics dataset (∼ 120,000 nodes per client) with 9 clients.

Federated Method	Prompt Method	Time (min)	Memory (G)
FedAvg	GPF	9.52	2.29
	ProG	10.83	2.37
	SUPT	10.11	2.52
FedProx	GPF	9.84	2.32
	ProG	11.21	2.38
	SUPT	10.62	2.54
SCAFFOLD	GPF	11.81	2.39
	ProG	14.61	2.57
	SUPT	12.63	2.48
FedGPL		**7.51**	**0.43**

Table 5: Evaluation of communication cost. $f_p(\cdot|\cdot)$: prompting function; $\tilde{G}$: prompted graph; Comm.: communication.

	$f_p(\cdot|\cdot)$ Size	$\tilde{G}$ Size	Comm. Cost
FedAvg+GPF	20,000	20,000	81,600
FedAvg+ProG	1,000	21,000	45,600
FedGPL	**100**	**10,000**	**21,800**
FedGPL*	1,100	**10,000**	23,800

47.80% compared with FedAvg+ProG. The significant reduction is attributed to the smaller sizes of $f_p(\cdot|\cdot)$ and $\tilde{G}$. For $f_p(\cdot|\cdot)$, FedGPL requires only 0.5% and 10% parameters of GPF and ProG. The size of FedGPL's $\tilde{G}$ is 50% and 47.62% of FedAvg+GPF and FedAvg+ProG. In conclusion, VPG behaves 2.1× ∼ 3.7× more communication-efficiently than baselines. Therefore, FedGPL is scalable for massive FGL participants with lightweight transmission between the server and clients.

5.4.3 Time efficiency. We demonstrate the time efficiency for our method in Table 4. Specifically, FedGPL can decrease the training time cost from 21.1% to 51.4% compared to baseline methods. The significant improvement in training time efficiency is attributed to reduced training parameters and communication messages. Thus, FedGPL is suitable for large-scale datasets with millions of nodes and edges in resource-constrained real-world FGL scenarios.

6 RELATED WORKS

Here we review works of two related research fields, including federated graph learning, and graph prompt learning.

6.1 Federated Graph Learning

Federated Learning (FL) [23] is a distributed machine learning algorithm that allows multiple clients to collaborate without leakage of their data. Studies have focused on various issues such as privacy-protection [33, 36, 44], data heterogeneity [14, 18, 19], model heterogeneity [13, 30], communication costs [1, 5, 16]. The integration of

FL with graph learning, called Federated Graph Learning (FGL), has provided promising solutions to many real-world problems [3, 8, 42]. Researchers study FGL to allow institutions to collaboratively learn models while keeping local graph data private [11]. Existing FGL works propose various GNN modules to improve downstream prediction effectiveness. For example, FedSage+ [43] generates local nodes' neighbor features to offset the ignorance of cross-client edges due to subgraph-level privacy preservation, which improves predicting ability with significant extra training costs. Furthermore, recent works investigate to tackle the non-IID issue in FGL. FedStar [29] shares common structural knowledge across heterogeneous clients by a personalized module. FedLIT [38] automatically discovers link-type heterogeneity to alleviate the useless structure to enable more effective message passing. FGGP [35] addresses non-IID graphs by decoupling structure and attribute domain shift based on prototype learning. However, they overlook task-heterogeneous FGL, a more general and difficult setting. Moreover, existing frameworks fail to adapt the scenario with heterogeneous downstream tasks. Our work proposes a novel framework tailored for task- and data- heterogeneous FGL to fill this gap.

6.2 Graph Prompt Learning

Prompting methods have achieved significant success in Natural Language Processing (NLP). Thus, researchers are exploring applying prompting methods to graph learning tasks. The heterogeneity among different graph learning tasks is more complex than NLP, which brings unique challenges. A main branch of prompts for graph learning is inserting extra feature vectors as prompts into the feature space of nodes and graphs. [7, 10, 21, 31, 47] add trainable feature vectors to the node features, aligning different types of pre-training tasks and downstream tasks. [22] injects heterogeneous feature prompt into the input graph feature space. Other methods design extra graphs as prompts and link them with the input graph. For example, [9, 12, 27] design extra small graphs as prompts and link the prompt graph with the input graph to perform the prompting process. The above works deploy prompting techniques on the features or structures of graph data. In our work, we design VPG for featural and structural prompting, which adds extra knowledge by nodes and edges, and neutralizes noisy structures to reduce the harm from task and data heterogeneity.

7 CONCLUSIONS

This paper investigated the multifaceted heterogeneity of graph task and data in FGL, which is unexplored in previous studies. We first construct a split framework for the separate preservation of global and local knowledge. Moreover, we developed HiDTA, a federated aggregator to hierarchically share asymmetrically beneficial knowledge with task heterogeneity elimination. Next, we devise a graph prompt that is adaptively generated to emphasize informative structures and denoise useless ones to improve representation efficiency. We provide theoretical analysis to prove the successful mitigation of the graph task and data heterogeneity. We also conduct extensive experiments on our approach and baseline methods to show the accuracy and efficiency superiority of FedGPL on large-scale datasets with millions of nodes.

REFERENCES

[1] Omair Rashed Abdulwareth Almanifi, Chee-Onn Chow, Mau-Luen Tham, Joon Huang Chuah, and Jeevan Kanesan. 2023. Communication and computation efficiency in Federated Learning: A survey. *Internet of Things* (2023), 100742.

[2] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815* (2017).

[3] Fahao Chen, Peng Li, Toshiaki Miyazaki, and Celimuge Wu. 2021. Fedgraph: Federated graph learning with intelligent sampling. *IEEE Transactions on Parallel and Distributed Systems* 33, 8 (2021), 1775–1786.

[4] Jiasi Chen and Xukan Ran. 2019. Deep learning with edge computing: A review. *Proc. IEEE* 107, 8 (2019), 1655–1674.

[5] Mingzhe Chen, Nir Shlezinger, H Vincent Poor, Yonina C Eldar, and Shuguang Cui. 2021. Communication-efficient federated learning. *Proceedings of the National Academy of Sciences* 118, 17 (2021), e2024789118.

[6] William Falcon et al. 2019. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning* 3, 6 (2019).

[7] Taoran Fang, Yunchao Zhang, Yang Yang, and Chunping Wang. 2022. Prompt tuning for graph neural networks. *arXiv preprint arXiv:2209.15240* (2022).

[8] Xingbo Fu, Binchi Zhang, Yushun Dong, Chen Chen, and Jundong Li. 2022. Federated graph machine learning: A survey of concepts, techniques, and applications. *ACM SIGKDD Explorations Newsletter* 24, 2 (2022), 32–47.

[9] Qingqing Ge, Zeyuan Zhao, Yiding Liu, Anfeng Cheng, Xiang Li, Shuaiqiang Wang, and Dawei Yin. 2023. Enhancing graph neural networks with structure-based prompt. *arXiv preprint arXiv:2310.17394* (2023).

[10] Chenghua Gong, Xiang Li, Jianxiang Yu, Cheng Yao, Jiaqi Tan, Chengcheng Yu, and Dawei Yin. 2023. Prompt Tuning for Multi-View Graph Contrastive Learning. *arXiv preprint arXiv:2310.10362* (2023).

[11] Zhuoning Guo, Duanyi Yao, Qiang Yang, and Hao Liu. 2024. HiFGL: A Hierarchical Framework for Cross-silo Cross-device Federated Graph Learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 968–979.

[12] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. 2023. PRODIGY: Enabling In-context Learning Over Graphs. *arXiv preprint arXiv:2305.12600* (2023).

[13] Jaehee Jang, Heoneok Ha, Dahuin Jung, and Sungroh Yoon. 2022. FedClassAvg: Local representation learning for personalized federated learning on heterogeneous neural networks. In *Proceedings of the 51st International Conference on Parallel Processing.* 1–10.

[14] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning.* PMLR, 5132–5143.

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

[16] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[17] Junhyun Lee, Wooseong Yang, and Jaewoo Kang. 2024. Subgraph-level Universal Prompt Tuning. *arXiv preprint arXiv:2402.10380* (2024).

[18] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE).* IEEE, 965–978.

[19] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[20] Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, and Han Yu. 2024. Federated graph neural networks: Overview, techniques, and challenges. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[21] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023.* 417–428.

[22] Yihong Ma, Ning Yan, Jiayu Li, Masood Mortazavi, and Nitesh V Chawla. 2023. HetGPT: Harnessing the Power of Prompt Tuning in Pre-Trained Heterogeneous Graph Neural Networks. *arXiv preprint arXiv:2310.15318* (2023).

[23] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics.* PMLR, 1273–1282.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[25] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems* 33 (2020), 12559–12571.

[26] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. arXiv 2018. *arXiv preprint arXiv:1811.05868* (2018).

[27] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining (KDD'23)* (Long Beach, CA, USA). 2120–2131. https://doi.org/10.1145/3580305.3599256

[28] Xiangguo Sun, Jiawen Zhang, Xixi Wu, Hong Cheng, Yun Xiong, and Jia Li. 2023. Graph Prompt Learning: A Comprehensive Survey and Beyond. *arXiv preprint arXiv:2311.16534* (2023).

[29] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. 2023. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 9953–9961.

[30] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. 2022. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, No. 8.* 8432–8440.

[31] Zhen Tan, Ruocheng Guo, Kaize Ding, and Huan Liu. 2023. Virtual Node Tuning for Few-shot Node Classification. *arXiv preprint arXiv:2306.06063* (2023).

[32] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. 2022. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8485–8493.

[33] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security.* 1–11.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[35] Guancheng Wan, Wenke Huang, and Mang Ye. 2024. Federated graph learning under domain shift with generalizable prototypes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 15429–15437.

[36] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.

[37] Jun Xia, Yanqiao Zhu, Yuanqi Du, and Stan Z Li. 2022. A survey of pretraining on graphs: Taxonomy, methods, and applications. *arXiv preprint arXiv:2202.07893* (2022).

[38] Han Xie, Li Xiong, and Carl Yang. 2023. Federated node classification over graphs with latent link-type heterogeneity. In *Proceedings of the ACM Web Conference 2023.* 556–566.

[39] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations.*

[40] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. *Advances in neural information processing systems* 32 (2019).

[41] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning.* PMLR, 7252–7261.

[42] Huanding Zhang, Tao Shen, Fei Wu, Mingyang Yin, Hongxia Yang, and Chao Wu. 2021. Federated graph learning–a position paper. *arXiv preprint arXiv:2105.11099* (2021).

[43] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. *Advances in Neural Information Processing Systems* 34 (2021), 6671–6682.

[44] Xiaojin Zhang, Yan Kang, Kai Chen, Lixin Fan, and Qiang Yang. 2023. Trading Off Privacy, Utility, and Efficiency in Federated Learning. *ACM Transactions on Intelligent Systems and Technology* 14, 6 (2023), 1–32.

[45] Jinhua Zhu, Yingce Xia, Lijun Wu, Shufang Xie, Wengang Zhou, Tao Qin, Houqiang Li, and Tie-Yan Liu. 2023. Dual-view molecular pre-training. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 3615–3627.

[46] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).

[47] Yun Zhu, Jianhao Guo, and Siliang Tang. 2023. SGL-PT: A Strong Graph Learner with Graph Prompt Tuning. *arXiv preprint arXiv:2302.12449* (2023).

A APPENDIX

A.1 Graph Learning Tasks

Here we define three levels of graph learning tasks.

(1) Node-level Task. In a graph $G = (V, E, X)$, given a node $v_i \in V$ with label y_i, the node-level task aims to learn a model $\mathcal{F}_{node}(\cdot)$ satisfying $\mathcal{F}_{node}(v_i, G) = y_i$. The node-level dataset is $D_{node} = \{\cdots, ((v_i, G), y_i), \cdots\}$.

(2) Edge-level Task. In a graph $G = (V, E, X)$, given two nodes $v_i, v_j \in V$ with a label $y_{i,j}$, the edge-level task aims to learn a model $\mathcal{F}_{edge}(\cdot)$ satisfying $\mathcal{F}_{edge}(v_i, v_j, G) = y_{i,j}$. The edge-level dataset is $D_{edge} = \{\cdots, ((v_i, v_j, G), y_{i,j}), \cdots\}$.

(3) Graph-level Task. For a graph $G = (V, E, X)$ with a label y, the graph-level task aims to learn a model $\mathcal{F}_{graph}(\cdot)$ satisfying $\mathcal{F}_{graph}(G) = y$. The graph-level dataset is $D_{graph} = \{\cdots, ((G_i), y_i), \cdots\}$.

A.2 Graph Inducing

To align three basic graph learning tasks, we follow the existing work [27] to reformulate node-level and edge-level tasks as graph-level tasks, respectively, by transforming their inputs into induced graphs, a κ-ego network. For node-level tasks, an induced graph includes a node and its neighbors within κ hops. For edge-level tasks, an induced graph includes two nodes and their neighbors within κ hops.

In our experiments, we follow [27] construct datasets. For edge-level tasks, we specifically choose samples where the endpoints have identical labels, setting the label of the edge-level sample to match that of its endpoints. For the graph-level tasks, we select the majority label of the subgraph nodes as the graph-level sample's label.
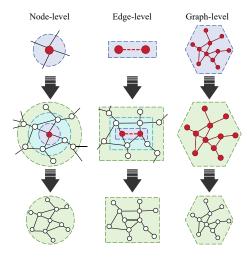

Figure 3: Graph inducing for three levels of tasks.

A.3 Differential Privacy

Here we define the Differential Privacy (DP) [36] for information privatization.

DEFINITION 5. ***Differential Privacy.*** *A mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ that maps domain $\mathcal{M}$ to range $\mathcal{R}$ meets (ε, δ)-differential privacy if it satisfies:*

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S] + \delta \tag{10}$$

where $d, d' \in \mathcal{D}$ are two adjacent inputs, $S \subseteq \mathcal{R}$ are any subset of the mechanism's outputs.

A.4 Training Algorithm

Here we take the node classification task as an example to illustrate the training pipeline in Algorithm 1. We set the GNN as a pre-trained and learnable model, which will be optimized in the server after aggregation.

Algorithm 1: FedGPL training workflow.

Input: A server S with a GNN $\mathcal{G}$, and N clients with $f_p(\cdot)$'s parameters θ_p and $f_h(\cdot)$'s parameters θ_h.

Output: Optimized prompt parameters and head parameters for each client.

1 Initialize each client's parameters $\theta_p^{(0)}$ and $\theta_h^{(0)}$;

2 **for** *n-th training round not converge* **do**

3   **foreach** *i-th client C_i in parallel* **do**

4   C_i prompts all the k-th input graphs $\{G_{i,k}\}$ by $\{\tilde{G}_{i,k}\} = \{f_p(G_{i,k}|\hat{G}_i)\}$;

5   C_i sends differentially privatized $\{\tilde{G}_{i,k}\}$ to S;

6   S computes representations of $\{\tilde{G}_{i,k}\}$ by $\{h_{i,k}\} = \{\mathcal{G}(\tilde{G}_{i,k})\}$;

7   S sends $\{h_{i,k}\}$ to C_i;

8   C_i computes estimated labels $\{\hat{y}_{i,k}\} = \{f_h(h_{i,k})\}$;

9   C_i calculates loss values and enables backward propagation;

10   C_i calculates gradients of $\theta_p^{(n-1)}$ and $\theta_h^{(n-1)}$;

11   C_i optimizes $\theta_p^{(n)}$ and $\theta_h^{(n)}$;

12   **end**

13   S calculates $\{\tau_{i \leftarrow j}^{(n+1)} | i, j \in [1, N]\}$ according to $\theta_i^{(n)} = (\theta_{p,i}^{(n)}, \theta_{h,i}^{(n)})$ as Equation 5 and 6;

14   S optimizes parameters of $\mathcal{G}$;

15   S sends $\{\tau_{i \leftarrow j}^{(n)} | j \in [1, N]\}$ to C_i for $i \in [1, N]$;

16   **foreach** *i-th client C_i in parallel* **do**

17   C_i updates parameters $\theta_p^{(n)}$ and $\theta_h^{(n)}$ as Equation 7;

18   **end**

19 **end**

A.5 Proof of Theorem 1

PROOF. Basically, we suppose that FedGPL consists of two participants as a-th and b-th clients. Their local parameters are $\theta_a^{(l)}$ and $\theta_b^{(l)}$ at l-th step. We can estimate the optimal learning directions by local data as $\theta_a^{(l)\prime}$ and $\theta_b^{(l)\prime}$. ACCording to Definition 3 we can calculate pairwise transferability as $\tau_{a \leftarrow a}, \tau_{a \leftarrow b}, \tau_{b \leftarrow a}, \tau_{b \leftarrow b}$. Moreover, based on Equations 5 and 6, we can calculate their parameters

after federated aggregation as

$$
\begin{aligned}
\theta_a^{(l+1)} &= \frac{\tau_{a\leftarrow a}}{\tau_{a\leftarrow a}+\tau_{a\leftarrow b}}\theta_a^{(l)} + \frac{\tau_{a\leftarrow b}}{\tau_{a\leftarrow a}+\tau_{a\leftarrow b}}\theta_b^{(l)}, \\
\theta_b^{(l+1)} &= \frac{\tau_{b\leftarrow a}}{\tau_{b\leftarrow a}+\tau_{b\leftarrow b}}\theta_a^{(l)} + \frac{\tau_{b\leftarrow b}}{\tau_{b\leftarrow a}+\tau_{b\leftarrow b}}\theta_b^{(l)}.
\end{aligned} \tag{11}
$$

Next, we assume the learning direction is consistent between two steps, thus we can estimate their optimal learning direction at $(l+1)$-th step as

$$
\begin{aligned}
\theta_a^{(l+1)\prime} &= \theta_a^{(l+1)} + \overrightarrow{\theta_a^{(l)\prime} - \theta_a^{(l)}}, \\
\theta_b^{(l+1)\prime} &= \theta_b^{(l+1)} + \overrightarrow{\theta_b^{(l)\prime} - \theta_b^{(l)}}.
\end{aligned} \tag{12}
$$

The 2-norm parameter difference between two tasks with our proposed aggregation algorithm can be calculated as

$$
\begin{aligned}
\|\theta_a^{(l+1)\prime} - \theta_b^{(l+1)\prime}\|_2 = \ & \|(\frac{\omega_a}{1+\omega_a} + \frac{\omega_b}{1+\omega_b})(\|\overrightarrow{\theta_b^{(l)} - \theta_a^{(l)}}\|) \\
& + (\theta_a^{(l+1)} - \theta_b^{(l+1)})\|_2,
\end{aligned} \tag{13}
$$

where $\omega_a = \frac{\overrightarrow{\theta_a^{(l)\prime}-\theta_a^{(l)}}\cdot\overrightarrow{\theta_b^{(l)\prime}-\theta_a^{(l)}}}{\|\overrightarrow{\theta_a^{(l)\prime}-\theta_a^{(l)}}\|_2}$ and $\omega_b = \frac{\overrightarrow{\theta_b^{(l)\prime}-\theta_b^{(l)}}\cdot\overrightarrow{\theta_a^{(l)\prime}-\theta_b^{(l)}}}{\|\overrightarrow{\theta_b^{(l)\prime}-\theta_b^{(l)}}\|_2}$. Obviously, if $\Delta_T^{a,b}(\theta_a^{(l+1)}, \theta_b^{(l+1)}) \leq \Delta_T^{a,b}(\theta_a^{(l)\prime}, \theta_b^{(l)\prime})$ that the task heterogeneity is minor after applying HiDTA, we have

$$
\begin{aligned}
\|\theta_a^{(l+1)\prime} - \theta_b^{(l+1)\prime}\|_2 &\leq \|\theta_a^{(l)} - \theta_b^{(l)\prime}\|_2, \\
(\frac{\omega_a}{1+\omega_a} + \frac{\omega_b}{1+\omega_b})(\|\overrightarrow{\theta_b^{(l)} - \theta_a^{(l)}}\|) &\leq 0, \\
\frac{\omega_a}{1+\omega_a} + \frac{\omega_b}{1+\omega_b} &\leq 0, \\
\frac{1}{\omega_a} + \frac{1}{\omega_b} + 2 &\geq 0, \\
\frac{\|\overrightarrow{\theta_a^{(l)\prime}-\theta_a^{(l)}}\|}{\tau_{a\leftarrow b}} + \frac{\|\overrightarrow{\theta_b^{(l)\prime}-\theta_b^{(l)}}\|}{\tau_{b\leftarrow a}} + 2 &\geq 0
\end{aligned} \tag{14}
$$

where $\|\overrightarrow{\theta_b^{(l)} - \theta_a^{(l)}}\| \geq 0$. Therefore, when the pairwise transferability values between two clients $\tau_{a\leftarrow b}, \tau_{b\leftarrow a}$ are positive, we have $\Delta_T^{a,b}(\theta_a^{(l+1)\prime}, \theta_b^{(l+1)\prime}) \leq \Delta_T^{a,b}(\theta_a^{(l)\prime}, \theta_b^{(l)\prime})$, which means that the task heterogeneity decrease after aggregation. □

A.6 Proof of Theorem 2

PROOF. Given a graph $G(V, E, X)$, we assume that a graph representation computed by a GNN $\mathcal{G}$ follows $h_G \sim U[0, \eta]$. And the GNN is smooth as $h_G = \mathbb{E}_{E\to\emptyset}[\mathcal{G}(V, E, X)]$. The representation of prompted graph $\tilde{G}$ by a VPG $\hat{G} = ((\hat{V}^+, \hat{V}^-), (\hat{E}^+, \hat{E}^-), (\hat{X}^+, \hat{X}^-))$ can estimated as

$$
h_{\tilde{G}} = \frac{|V|\cdot h_G + |\hat{V}^+|\cdot h_{G^+} - |\hat{V}^-|\cdot h_G}{|V| + |\hat{V}^+| - |\hat{V}^-|}, \tag{15}
$$

where $h_{G^+} = \mathcal{G}(\hat{V}^+, \hat{E}^+, \hat{X}^+)$. Next, we assume $\hat{X}^+ \sim X$, $|\hat{V}^+| << |V|$, and $|\hat{V}^+| << |\hat{V}^-|$ in practice, thus we can get the distribution of prompted graphs as

$$
h_{\tilde{G}} \sim \tilde{U}[\eta\alpha_n, \eta], \tag{16}
$$

where $\alpha_n \in [0, 1]$ is a pre-defined percentage parameter of significance score borderline in VPG.

To measure the data heterogeneity between i-th and j-th clients, we denote their graph data following uniform distributions of $h_{G_a} \sim U^a[0, \eta^a]$ and $h_{G_b} \sim U^b[0, \eta^b]$, respectively. We can infer

$h_{\tilde{G}_a} \sim \tilde{U}^a[\eta^a\alpha_n^a, \eta^a]$ and $h_{\tilde{G}_b} \sim \tilde{U}^b[\eta^b\alpha_n^b, \eta^b]$. Then, the expectation of the 2-norm embedding difference with and without graph prompting $\tilde{\delta}$ can be calculated by

$$
\begin{aligned}
\mathbb{E}[\tilde{\delta}] &= \mathbb{E}_{\substack{h_{\tilde{G}_a}\sim\tilde{U}^a \\ h_{\tilde{G}_b}\sim\tilde{U}^b}}[\|h_{\tilde{G}_a} - h_{\tilde{G}_b}\|_2] - \mathbb{E}_{\substack{h_{G_a}\sim U^a \\ h_{G_b}\sim U^b}}[\|h_{G_a} - h_{G_b}\|_2] \\
&= ((\frac{1-\alpha_n^a}{2}\eta^a)^2 + (\frac{1-\alpha_n^b}{2}\eta^b)^2 - 2(\frac{1-\alpha_n^a}{2}\eta^a)(\frac{1-\alpha_n^b}{2}\eta^b)) \\
&\quad - ((\frac{1}{2}\eta^a)^2 + (\frac{1}{2}\eta^b)^2 - 2(\frac{1}{2}\eta^a)(\frac{1}{2}\eta^b)) \\
&= \frac{(1-\alpha_n^a)^2-1}{4}\eta^{a2} + \frac{(1-\alpha_n^b)^2-1}{4}\eta^{b2} - 2(\frac{(1-\alpha_n^a)(1-\alpha_n^b)-1}{4}\eta^a\eta^b).
\end{aligned} \tag{17}
$$

Obviously, when $\alpha_n^a \approx \alpha_n^b$, we ensure

$$
\mathbb{E}[\tilde{\delta}] \approx \frac{(1-\alpha_n^a)^2-1}{4}(\eta_a - \eta_b)^2 \leq 0, \tag{18}
$$

because $(1-\alpha_n^a)^2 - 1 \leq 0$. Subsequently, according to Equation 1 that is monotonically increasing, the data heterogeneity can be reduced after graph prompting as

$$
\mathbb{E}_{\substack{h_{\tilde{G}_a}\sim\tilde{U}^a \\ h_{\tilde{G}_b}\sim\tilde{U}^b}}[\Delta_D^{i,j}(h_{\tilde{G}_a}, h_{\tilde{G}_b})] \leq \mathbb{E}_{\substack{h_{G_a}\sim U^a \\ h_{G_b}\sim U^b}}[\Delta_D^{i,j}(h_{G_a}, h_{G_b})]. \tag{19}
$$

□

A.7 Dataset Statistics

Table 6 are the statistics of graph datasets, including Cora, CiteSeer, DBLP, Photo, and Physics, which are transformed from traditional graph learning datasets for node classification.

Table 6: Statistics of datasets.

Dataset	# Nodes	# Edges	# Features	# Labels
Cora	2,708	5,429	1,433	7
CiteSeer	3,327	9,104	3,703	6
DBLP	17,716	105,734	602	6
Photo	7,650	238,162	745	8
Physics	34,493	495,924	8,415	5

A.8 Implementation Details

For the implementation of federated learning, we deploy 9 clients, and each level of tasks (node-level, edge-level, and graph-level) contains 3 clients, each client has 400 induced graphs on average, which are induced from the raw dataset with $\kappa = 5$, and each client participates in every communication round, during each round, clients undergo a single epoch of training, and the global communication round is set to 50. For the implementation of graph prompting methods, for GPF, we add additional feature vectors into the node features, for ProG, we set the number of tokens as 10. Each client contains its training dataset, prompt, and an answering layer that projects the embedding output by the GNN to the final results. The learning rate is set as 0.1 for all datasets. The results are averaged on all clients. The HiFGL framework is implemented based on PyTorch [24], and PyTorch-Lightning [6] runs on the machine with Intel Xeon Gold 6148 @ 2.40GHz, V100 GPU and 64G memory.

Table 7: Overall performance (ACC (%) and F1 (%)) of different federated algorithms and graph prompting methods in few-shot settings.

Federated Method	Prompt Method	Cora		CiteSeer		DBLP		Photo		Physics	
		ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Local	GPF	80.06	79.84	82.34	81.84	78.46	78.15	74.51	73.14	86.15	86.21
	ProG	84.25	84.31	83.19	83.04	79.84	79.73	69.84	69.47	85.89	85.03
	VPG	85.81	85.10	84.52	84.37	82.31	82.17	77.97	76.84	88.03	88.14
FedAvg	GPF	79.81	79.57	82.79	82.87	79.64	78.84	79.03	78.14	86.87	86.65
	ProG	80.44	80.26	84.54	83.81	80.14	80.20	70.12	70.03	86.84	86.67
	VPG	86.14	85.82	85.01	84.77	80.72	80.07	79.31	79.01	88.72	88.12
FedProx	GPF	77.37	76.41	81.04	80.97	78.07	77.91	80.57	80.19	84.58	84.29
	ProG	78.59	78.14	80.08	80.14	77.48	77.97	70.16	70.07	85.17	85.01
	VPG	84.32	84.23	81.24	80.88	78.14	78.45	84.25	84.12	90.05	90.03
SCAFFOLD	GPF	72.94	72.44	72.48	70.84	77.61	77.31	72.15	70.13	82.14	82.23
	ProG	70.67	68.47	60.41	60.87	75.14	75.10	68.72	68.12	83.17	83.04
	VPG	79.74	79.55	75.85	84.76	77.22	77.07	73.17	72.94	86.14	86.07
HiDTA	GPF	80.68	80.81	82.99	82.83	80.75	80.66	84.10	84.07	87.02	87.06
	ProG	80.22	79.28	83.48	83.45	80.67	80.17	69.47	69.38	88.87	88.17
FedGPL		**86.45**	**86.42**	**85.59**	**85.56**	**82.72**	**81.87**	**85.28**	**85.26**	**90.14**	**90.09**

A.9 Overall Performance for Supervised and Prompt Learning

We evaluate the performance by training modules from scratch (*i.e.*, Supervised) and freezing the pre-trained GNN (*i.e.*, Prompt) in Table 8. Beyond fine-tuning, the results of the two training schemes depict that FedGPL behaves more competitively than others on two datasets.

Table 8: ACC (%) and F1 (%) of different federated algorithms and graph prompting methods. Supervised: training models from scratch; Prompt: fine-tuning models based on a frozen pre-trained GraphTransformer.

Training Scheme	Federated Method	Prompt Method	Cora		CiteSeer	
			ACC	F1	ACC	F1
Training	FedAvg	GPF	85.67	85.34	84.62	84.2
		ProG	87.96	87.61	85.71	85.87
		SUPT	86.31	86.24	85.98	85.51
	FedProx	GPF	81.34	80.51	83.22	82.71
		ProG	85.61	85.40	84.41	83.85
		SUPT	82.31	82.45	83.49	83.8
	FedGPL		**89.14**	**88.57**	**88.12**	**87.94**
Prompt	FedAvg	GPF	80.50	79.72	83.22	82.71
		ProG	84.02	83.77	84.41	83.85
		SUPT	81.83	80.61	83.49	83.8
	FedProx	GPF	77.81	76.97	82.61	82.33
		ProG	82.54	81.81	82.64	81.67
		SUPT	78.67	78.34	82.45	82.37
	FedGPL		**87.61**	**87.13**	**87.58**	**87.74**

A.10 Overall Performance with Pre-trained GCN

We test the fine-tuned FGL performance based on a pre-trained GCN on Cora and Citeseer datasets in Table 9. FedGPL outperform baseline models in terms of ACC and F1.

Table 9: ACC (%) and F1 (%) of different federated algorithms and graph prompting methods based on a pre-trained GCN [15].

Federated Method	Prompt Method	Cora		CiteSeer	
		ACC	F1	ACC	F1
Local	GPF	83.14	82.64	81.63	81.37
	ProG	84.91	84.33	82.21	81.44
	SUPT	82.41	82.01	82.14	82.07
FedAvg	GPF	84.61	84.37	84.24	83.67
	ProG	85.61	85.03	84.81	84.02
	SUPT	83.17	83.34	83.61	83.44
FedProx	GPF	79.47	79.31	82.67	82.33
	ProG	83.44	82.97	83.41	83.01
	SUPT	78.67	78.34	82.76	82.03
SCAFFOLD	GPF	76.61	75.51	72.51	72.03
	ProG	72.61	72.81	73.67	73.11
	SUPT	73.35	72.85	72.61	72.17
FedGPL		**86.87**	**86.27**	**85.34**	**85.22**

Table 10: Comparison of ACC (%) for different graph prompting methods and training schemes. N, E, and G: node-, edge-, and graph-level task, (f): few-shot settings.

Method	Cora			CiteSeer			Physics		
	N	E	G	N	E	G	N	E	G
Supervised	80.72	95.21	95.14	81.00	97.45	88.83	83.07	92.78	99.78
Pre-train	77.06	87.52	88.61	80.68	93.19	72.25	75.44	78.94	95.36
Fine-tune	81.01	95.36	95.57	81.33	97.27	85.67	83.81	92.47	99.87
GPF	76.17	88.24	87.05	82.11	93.81	74.17	76.51	86.66	98.42
ProG	77.56	89.49	89.48	82.51	95.57	75.94	76.42	87.12	97.45
VPG	79.19	90.35	91.02	82.94	96.48	77.49	83.37	91.03	99.50
GPF(f)	74.35	82.33	85.62	81.02	92.88	73.11	75.27	79.74	96.04
ProG(f)	75.14	87.51	89.57	81.41	93.43	74.75	80.78	78.41	97.47
VPG(f)	77.64	89.46	90.35	81.89	94.82	76.84	82.14	83.78	98.17

A.11 Privacy Analysis

We evaluate the impact of using differential privacy on the performance of FedGPL. Here we conduct experiments by varying the privacy scale of ϵ and test it on the Cora dataset. The results are shown in Figure 4, which demonstrates the relationship between ACC and privatization parameters. We observe that the ACC is reduced as we increase the privacy constraints (decreasing ϵ). Similar conclusions are drawn from different settings. The finding indicates that there is a trade-off between performance and privacy in FedGPL, which is common in FL [44]. In practical utilization, we will choose proper ϵ for different degrees of privacy protection demands on GNN and graph data.

Figure 5: Transferability. N, E, and G: node-, edge-, and graph-level task, S: source task, and T: target task.
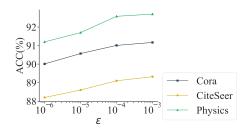

Figure 4: Performance-privacy correlation curves.

A.12 Case Study: Transferability

To further discuss how HiDTA becomes robust against two kinds of heterogeneity, we conduct a case study on transferability according to the aggregating weight matrix, to find out how transferability models the heterogeneous knowledge transfer. Specifically, Figure 5 shows the averaged and normalized transferability values in the last 20 training epochs on CiteSeer. The results suggest that HiDTA prioritizes learning directions with higher transferability. Besides, the optimization process of a target task mainly depends on itself, aligned with our motivations that aim to model asymmetric knowledge transfer.
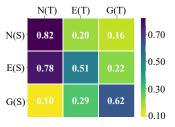
A.13 Performance on Few-shot Settings

Besides the standard setting with full training samples, we follow GPL works [27] to deploy a few-shot setting **(f)**. Few-shot settings are usually incorporated in prompt learning, which acts as an efficient learning manner. It lets each client have only a limited number of data samples. In our experiments, we utilize a 100-shot scenario, where each client possesses only 100 labeled samples. The prediction results are shown in Table 7. We observe that FedGPl still outperforms baselines in few-shot settings in a prompt tuning way, which demonstrates that its generalization ability successfully address FGL in a task- and data- heterogeneous scenario.

A.14 Experiments on Graph Prompting

To further evaluate the effectiveness of VPG, the graph prompting module of FedGPL, we compare VPG with GPF and ProG under the local graph prompting setting on Cora, CiteSeer, and Physics datasets, as shown in Table 10. We also compared with (1) supervised training: training a graph model from scratch, (2) fine-tuning: fine-tuning the GNN with labeled data, and (3) pre-training: freezing GNN with only a learnable task head. We observe that VPG performs better than ProG and GPF by 0.43% to 7.28% across all tasks on three datasets. Generally, prompting unleashes more potential of GNN on different downstream tasks with small-scale trainable parameters. In some cases, prompting even surpasses fine-tuning and supervised learning, which may be attributed to a lightweight model for more effective optimization.

Table 11: Comparison of ACC (%) for different graph prompting methods and federated algorithms. N, E, and G: node-, edge-, and graph-level task.

Federated Method	Prompt Method	Cora			CiteSeer			Physics		
		N	E	G	N	E	G	N	E	G
Local	GPF	76.17	88.24	87.05	82.11	93.81	74.17	76.51	86.66	98.42
	ProG	77.56	89.49	89.48	80.74	91.57	73.03	78.09	80.83	98.03
	VPG	79.19	90.35	91.05	82.94	96.48	77.49	83.37	91.03	99.51
FedAvg	GPF	75.87	87.21	86.74	82.44	93.71	75.03	77.94	88.96	98.57
	ProG	77.61	89.14	90.13	82.86	86.42	86.27	79.74	82.84	99.13
	VPG	79.34	90.57	91.26	83.31	96.78	78.21	83.42	91.21	99.34
FedProx	GPF	72.21	82.84	83.11	82.86	87.33	75.86	75.64	87.24	96.48
	ProG	75.84	87.31	85.74	81.03	91.91	73.13	81.87	83.13	98.32
	VPG	76.61	88.16	86.81	81.37	94.47	76.61	83.25	91.12	99.04
SCAFFOLD	GPF	69.28	78.48	79.51	75.41	79.18	69.37	72.57	84.47	94.42
	ProG	58.97	62.41	65.48	61.47	68.81	64.23	73.51	85.24	94.57
	VPG	70.57	77.04	75.52	72.61	84.76	67.11	79.88	87.34	95.21
HiDTA	GPF	74.13	85.91	86.01	81.47	94.51	76.68	79.92	87.87	98.82
	ProG	74.24	86.04	86.11	82.83	95.84	77.74	82.55	88.16	98.72
FedGPL		**82.19**	**92.49**	**92.02**	**83.41**	**96.81**	**78.41**	**83.86**	**91.79**	**99.47**

A.15 Hyperparameter Sensitivity

We also investigate the impact of different α_n, as shown in Figure 6, where we conclude that optimal α_n of tasks are distinct. For example, $\alpha_n = 0.3$ is optimal for the graph-level task, while $\alpha_n = 0.5$ is optimal for the edge-level task. Results further reveal the existence of heterogeneity between different levels of tasks. This heterogeneity indicates personalized prompting is essential and motivates us to design a selective cross-client knowledge sharing method to improve prediction accuracy.
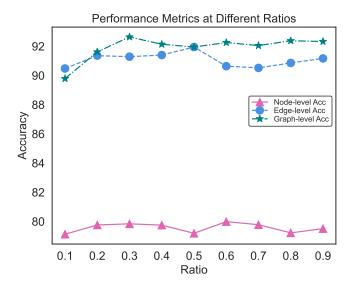
A.16 Task Performance

Table 11 is a detailed comparison of the accuracy of different federated learning algorithms combined with graph prompting methods on three datasets across three types of tasks. From the table, it is evident that FedGPL outperforms other methods in all three tasks. For example, on the Cora dataset, compared to local training, FedGPL shows improvements of 4.02%, 2.14%, and 0.97% in node-level, edge-level, and graph-level tasks, respectively. This demonstrates that FedGPL enhances prompting performance by facilitating knowledge sharing across different tasks. On the other hand, compared to FedAvg, HiDTA exhibits better performance with VPG. For instance, on the Cora dataset, HiDTA shows improvements of 3.00%, 2.14%, and 0.97% in the three tasks compared to FedAvg, indicating that HiDTA better mitigates the heterogeneity between different tasks, enabling more effective knowledge sharing.

Figure 6: Parameter sensitivity of α_n.