# Unleashing the Power of Pre-trained Graph Models in Federated Graph Learning

Huabin Sun[1⋆], Bo Yan[1⋆], Yaoqi Liu[1], Shaohua Fan[2], Yang Cao[3], and Chuan Shi[1✉]

[1] Beijing University of Posts and Telecommunications, Beijing, China
{sunhuabin, boyan, yaoqiliu, shichuan}@bupt.edu.cn
[2] Inner Mongolia University, Hohhot, China
shfan@imu.edu.cn
[3] Institute of Science Tokyo, Tokyo, Japan
cao@c.titech.ac.jp

**Abstract.** Federated graph learning (FGL) enables collaborative model training without sharing local data, but suffers from severe client heterogeneity, such as shifts in label distributions and graph sizes. Pre-trained graph models (PGMs), learned from large-scale graph data, encode general structural and semantic priors, offering a promising solution to alleviate heterogeneity in FGL. To this end, we first explores the potential of PGMs for FGL. We empirically reveal that directly fine-tuning PGMs in federated settings often results in degraded performance due to misalignment between the global prior and heterogeneous local distributions. To address this misalignment, we introduce PeFGL, a PGM-enhanced FGL framework that effectively adapts pre-trained priors to heterogeneous local graphs. PeFGL is built on two key components. First, we employ conditional generative diffusion models to augment local graphs, enriching structural diversity and compensating for missing patterns that hinder PGM adaptation. We further design a pre-trained-knowledge-guided filtering mechanism to select generated samples that align with both the PGM prior and local distribution. Then, based on the augmented and filtered data, we extract invariant subgraphs that can be shared across clients to guide PGM fine-tuning, which suppresses noisy substructures while reducing cross-client heterogeneity. Extensive experiments on four real-world datasets demonstrate that our framework consistently achieves state-of-the-art performance.

**Keywords:** Federated graph learning · Pre-trained graph models · Heterogeneous graph data

## 1 Introduction

Graph Neural Networks (GNNs) [5,10,20] have become a powerful framework to learn representations from structured graph data. Through neighborhood aggregation, they have demonstrated outstanding performance in diverse applications

⋆ Equal contribution

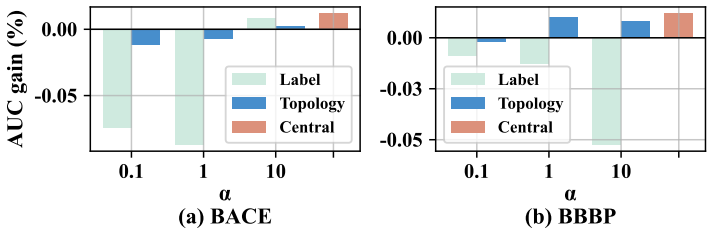


Fig. 1: The performance gains by fine-tuning PGMs under different types (label, topology) and degrees (0.1, 1, 10) of heterogeneity.

such as node and graph classification. Traditional GNNs hold a basic assumption that the graph data is centrally stored. In practice, however, the graph data is owned by different parties, and due to privacy concerns, it is hard to collect data centrally, which makes the centralized training of GNNs infeasible.

Federated learning (FL) [14,27] provides a paradigm for collaboratively training models by multiple parties without sharing raw data. To enable graph data to enjoy the benefits, researchers have developed federated graph learning (FGL) [28,22,32], where distributed graph owners jointly train GNNs under privacy constraints. A major challenge of FGL is data heterogeneity, i.e., the graph data of different clients is not independent and identically distributed (non-IID), which dramatically undermines the performance of FGL. Existing studies addressing heterogeneity can be divided into knowledge sharing and data augmentation. Knowledge sharing methods aim to share common knowledge (e.g., structural and spectral knowledge) across clients and leave personalized knowledge locally [18,19,23]. In contrast, data augmentation methods aim to augment local graphs to align the data distribution across clients [3,33]. However, these approaches largely operate within client-side data, where shared signals can be weak, and augmentation alignment is hard to control. This leaves open how to acquire stronger transferable knowledge to narrow client gaps.

Recently, public pre-trained knowledge, which captures transferable knowledge from large-scale datasets, has shown great potential in mitigating heterogeneity of FL in computer vision (CV) [31,15] and natural language processing (NLP) [1,13] domains. Typically, it has been shown that simply fine-tuning pre-trained models can yield significant improvements under heterogeneous settings [15,9]. On the other hand, pre-trained graph models (PGMs), such as GraphCL [29] and GCC [16], are trained on large-scale graph datasets to encode transferable structural and semantic priors, significantly boosting the performance of various downstream tasks. This naturally raises a question: *Can we directly fine-tune PGMs to mitigate the heterogeneity in FGL*? Answering this question helps to clarify whether PGMs' transferable priors can be easily adapted to FGL, or whether new designs are required to unleash their potentials.

Therefore, we conduct empirical studies to show the performance gains after fine-tuning PGMs under various degrees of heterogeneity, controlled by the Dirichlet parameter $\alpha$. We adopt the typical PGM provided by [7] and consider

two major types of graph heterogeneity (i.e., label and topology heterogeneity). The results on BACE and BBBP datasets are shown in Figure 1. We observe that PGMs improve the model performance in the centralized setting but suffers from negative transfer in the federated setting, which deviates from common observations in CV and NLP domains. This raises another question: *How to unleash the power of PGMs to tackle the heterogeneity issue in FGL*? Answering this question is non-trivial, since the complex structural priors captured by PGMs often conflict with heterogeneous local topologies and label distributions, making effective adaptation particularly challenging in federated settings.

In this paper, we propose a novel framework named PeFGL to effectively adapt PGMs to tackle the heterogeneity issue of FGL and further enhance the performance of FGL. Specifically, to mitigate the mismatch between the pre-trained knowledge and the heterogeneous local graph distributions, we employ a conditional generative diffusion model to augment local graphs, enriching their structural diversity and compensating for missing local patterns that impede PGM adaptation. We further introduce a pre-trained-knowledge-guided filtering mechanism that selectively retains generated samples consistent with both the PGM prior and the client-specific distribution. This mechanism measures the representation similarity between the original and generated samples via the PGM and preserves the top-ranked ones whose labels are consistently predicted by the PGM. Based on these augmented local graphs, we extract invariant subgraphs through local gradient regularization, which captures shared graph semantics across clients. These invariant structures provide reliable guidance for fine-tuning PGMs, enabling robust cross-client knowledge alignment while mitigating the adverse impact of noisy or domain-specific substructures.

The contribution of this paper is summarized as follows:

- To the best of our knowledge, this is the first work to explore the potential of PGMs for FGL, enabling more consistent and transferable graph representation learning across heterogeneous clients.
- We propose PeFGL, a novel framework that adapts pre-trained graph priors to heterogeneous local graphs. PeFGL first augments minority local classes via conditional diffusion and then aligns client updates by extracting invariant subgraphs shared across clients.
- Experiments on four datasets demonstrate that PeFGL outperforms existing FGL methods, achieving up to 6.69% improvements compared to baselines.

## 2 Related work

### 2.1 Federated graph learning

Briefly speaking, FGL can be divided into inter-graph and intra-graph FGL [30]. Intra-graph FGL assumes that each client owns a part of the entire graph while inter-graph FGL assumes that multiple entire graphs exist in each client [23,25,3,26]. This work considers the inter-graph FGL scenario. To tackle the heterogeneity issue in inter-graph FGL, GCFL [23] designs a sequence-based

gradient clustering method to identify homogeneous clients and aggregate gradients from the same clusters. FedStar [18] introduces a decoupled GNN that only shares the parameters which capture graph structure patterns. FedSSP [19] shares generic spectral knowledge among clients. Other studies seek to generate common graphs [3,19]. FedVN [3] introduces virtual nodes locally, and FedGOG [33] further explores to generate out-of-distributions (OOD) samples to achieve better generalization. Despite the success, none of the above methods consider leveraging external knowledge (e.g, PGMs) to enhance the inter-graph FGL.

### 2.2 Graph pre-training

The key insight of graph pre-training is to learn transferable knowledge from large unlabeled graphs, enabling efficient adaptation to downstream tasks via fine-tuning. [2]. Typical graph pre-training methods can be divided into contrastive-based and predictive-based. The contrastive-based methods aim to maximize the mutual information (MI) between two augmented graph views, thereby learning discriminative graph features [17,16,21,29]. For example, DGI [21] learns the node embeddings by maximizing the MI between global and local graph representations. GraphCL [29] designs different graph views by perturbing the graph information. Different from contrast-based methods, predictive-based methods pre-train the model by predicting the graph properties [6,7,8]. Among them, AttrMasking and ContextPred [7] predict masked node/edge attributes and subgraph-based contexts respectively. GPT-GNN [8] further generates the whole node attributes based on the graph structure. Nevertheless, when adapting these PGMs to downstream tasks, all these efforts assume that the downstream data is centrally stored, without considering the real-world privacy requirements.

## 3 Preliminary

In this section, we first present some basic definitions and concepts related to our work, including pre-trained graph models and federated graph learning, then we formalize our problem.

Let $G = (X, A)$ denote a graph $G$ with node features $X \in \mathbb{R}^{n \times d}$ and adjacency matrix $A$. We define the pre-trained graph models as follows.

**Definition 1.** ***Pre-trained Graph Model (PGM)****. A PGM refers to a graph neural network $f_{\boldsymbol{\phi}}$ that is first trained on a large-scale source graph dataset $\mathcal{D}_{pre}$ using supervised or self-supervised objectives to learn generalizable graph representations. Formally, the pre-training process can be expressed as:*

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathcal{L}_{pre}(f_{\boldsymbol{\phi}}, \mathcal{D}_{pre}), \tag{1}$$

*where $\mathcal{L}_{pre}$ denotes the pre-training loss (e.g., contrastive, predictive, or generative). The obtained parameters $\boldsymbol{\phi}^*$ encode transferable structural and semantic priors that can be fine-tuned on downstream graph tasks.*

Our graph model is composed of a pre-trained graph encoder $f_\phi : (X, A) \to \mathbb{R}^{d_z}$ and a classifier head $c_\psi : \mathbb{R}^{d_z} \to \mathbb{R}^C$, denoting as:

$$F_\theta = c_\psi \circ f_\phi, \tag{2}$$

where $\theta = (\phi, \psi)$ is the total parameters. Given a graph $G$, the graph model first encodes $G$ into a vector $z = f_\phi(G)$ and then obtains the class probabilities $p_\psi(y \mid G) = \text{softmax}(c_\psi(z))$. In practice, $\phi$ or $\psi$ can be fine-tuned with downstream graph data to adapt to specific tasks and enhance performance.

**Definition 2.** ***Federated Graph Learning (FGL).*** *FGL aims to collaboratively train a global graph model $\theta$ across multiple clients $\mathcal{C} = \{1, 2, \ldots, K\}$ without sharing raw data. The overall objective can be formalized as:*

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{K} w_k \frac{1}{|\mathcal{D}_k|} \sum_{(G,y)\in\mathcal{D}_k} \ell(F_\theta(G), y), \quad s.t. \sum_{k=1}^{K} w_k = 1. \tag{3}$$

Specifically, at each communication round $t$, each client $k$ locally optimizes its model parameters $\theta_k$ based on its private graph dataset $\mathcal{D}_k = \{(G_k^i, y_k^i)\}_{i=1}^{n_k}$. The global model is then updated by aggregating the local parameters:

$$\theta^{t+1} = \text{Agg}(\{\theta_k^t\}_{k=1}^K). \tag{4}$$

A typical aggregation function $\text{Agg}(\cdot)$ is Fedavg [14], which is defined as follows:

$$\theta^{t+1} = \sum_{k=1}^{K} \frac{n_k}{\sum_{j=1}^{K} n_j} \theta_k^t, \tag{5}$$

where $n_k$ denotes the number of samples in client $k$.

Based on the above preliminaries, we define our problem as follows:

**Definition 3.** ***PGM-enhanced FGL.*** *Given a PGM with parameters $\theta^*$ stored in the server, and a set of clients $\mathcal{C} = \{1, 2, \ldots, K\}$ with private dataset $\mathcal{D}_k$ on client $k$, the objective of PGM-enhanced FGL is:*

$$\min_{\theta} \sum_{k=1}^{K} w_k \frac{1}{|\mathcal{D}_k|} \sum_{(G,y)\in\mathcal{D}_k} \ell(F_\theta(G), y) + \Omega(\theta, \theta^*), \quad s.t. \sum_{k=1}^{K} w_k = 1, \tag{6}$$

*where $\ell$ is the loss function and $\Omega(\theta, \theta^*)$ is a regularization that aligns the global model $\theta$ with the pre-trained prior $\theta^*$.*

$\Omega(\theta, \theta^*)$ enables the model to retain global transferable knowledge while adapting to heterogeneous client distributions. A naive solution is directly setting $\theta^{(0)} = \theta^*$ and performing federated fine-tuning. However, it leads to optimization conflicts and even negative transfer, as shown before. Thus, the core challenge of PGM-enhanced FGL lies in mitigating the misalignment between $p(\theta^*)$ and the heterogeneous $\{p(\theta_k)\}_{k=1}^K$.

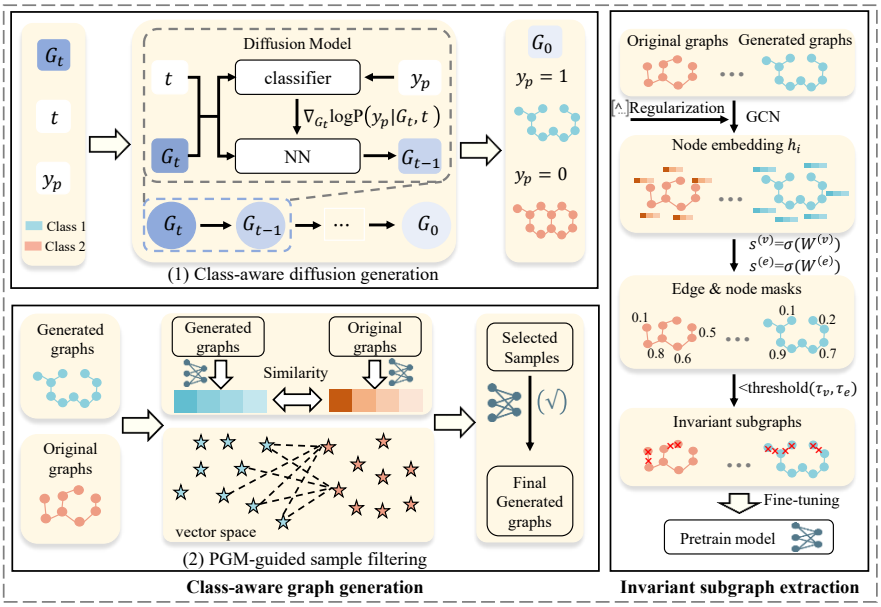


Fig. 2: The overall framework of PeFGL.

## 4 Methodology

In this section, we first present an overview of the proposed PeFGL, then detail the two key components of PeFGL, including class-aware graph generation and invariant subgraph extraction.

### 4.1 Overall framework

The overall framework of PeFGL is shown in Figure 2. PeFGL consists of two stages, graph generation and subgraph extraction. In the graph generation stage, all the clients first collaboratively train a class-aware diffusion model and then each client generates their minority-class graphs locally, therefore enhancing the data diversity and mitigating heterogeneity. To align the generated graph distribution with the local distribution and prior knowledge, a PGM-guided filter mechanism is further applied. First, the generated graphs are fed into the PGM encoder to obtain their representations, and the top-$k$ samples with the closest distances to the client's original graphs are selected as candidates. Among these candidates, only those whose predicted labels by the current PGM classifier match the true label are retained. In the subgraph extraction stage, based on the original and generated data, a collaboratively trained graph masker automatically extracts subgraphs that are invariant across clients and consistent with the prior knowledge. To achieve this, a global gradient regularizer is applied to the masker, aligning its local updates with global gradients. Finally, the extracted subgraphs are used to fine-tune PGM, thus mitigating negative transfer.

### 4.2 Class-aware graph generation

As discussed, naive fine-tuning of the PGM will harm performance due to the mismatch between PGM's prior and the heterogeneous local data distributions. This mismatch stems from their disparity: the PGM, trained on large-scale graph datasets, provides a general unbiased knowledge, whereas client data exhibits distributional bias due to class imbalance. To alleviate the resulting bias and mismatch, we adopt a generate-and-select strategy: we first use class-aware diffusion to generate minority-class samples; subsequently, recognizing that synthetic graphs may be noisy or inconsistent with the prior, we introduce a PGM-guided filter that retains only samples consistent with both the frozen prior and the local data statistics.

***Class-aware diffusion generation.*** Diffusion-based graph generation has two key phases. In the forward phase, a clean graph $G = (X, A)$ is gradually perturbed by a time-dependent noise kernel $q_t$. This produces noisy graphs $G_t = (X_t, A_t)$ at different times $t$, where larger $t$ means more noise has been added to the graph. In the reverse phase, a score-based diffusion model starts from noise and iteratively denoises $G_t$. It learns a time-dependent score function $s_\theta(G_t) \approx \nabla_{G_t} \log p_t(G_t)$ that approximates the score of the perturbed graph distribution $p_t$ at diffusion time $t$ and provides update directions that move $G_t$ toward higher-density regions. This reverse process defines an unconditional generative model that does not use label information.

We next extend this framework to class-conditional graph generation. Given a target class $y_i$, we construct a class-conditional score function that guides the reverse process according to both the data distribution and the label information. At diffusion time $t$, we define the class-guided score as

$$\begin{aligned} \tilde{s}_\theta(G_t; y_i) &= s_\theta(G_t) + \Lambda \nabla_{G_t} \log p(y_i \mid G_t, t), \\ \Lambda &= \mathrm{diag}(\lambda_X I_X, \lambda_A I_A). \end{aligned} \tag{7}$$

where $\nabla_{G_t} \log p(y_i \mid G_t, t)$ is the gradient of the class-posterior, and $\lambda_X, \lambda_A \geq 0$ control the guidance strength on node and edge components, respectively. This construction is consistent with Bayes' rule, since the score of the class-conditional distribution can be decomposed into a marginal term and a class-posterior term.

To instantiate the class-posterior term in Eq. (7), we require estimates of $\nabla_{G_t} \log p(y_i \mid G_t, t)$ across different noise levels $t$. We obtain these estimates by federatively training a noise-aware classifier $f_{noi}(G_t, t)$. On client $k$, the classifier is trained on the local dataset $\mathcal{D}_k$. For each sample $(G, y) \in \mathcal{D}_k$, we first sample a time $t$, apply the same forward kernel $q_t(\cdot \mid G)$ as in the diffusion process to obtain a noisy graph $G_t$, and then compute loss on $G_t$. The classification objective on client $k$ is defined as

$$\mathcal{L}_k^{\mathrm{clf}}(f) = \ell_{\mathrm{CE}}(f_{noi}(G_t, t),\, y), \tag{8}$$

where $\ell_{\mathrm{CE}}$ denotes the cross-entropy loss between the softmax of $f_{\mathrm{noise}}(G_t, t)$ and the label $y$. During federated training, each client uploads only classifier

parameters for weighted aggregation on the server, and the raw graphs remain on local devices.

After obtaining $\tilde{s}_\theta$, we use it in the reverse-time stochastic differential equation (SDE) that defines the denoising process. For the graph state $G_t$, the reverse SDE is written as

$$\mathrm{d}G_t = [f_t(G_t) - g_t^2\, \tilde{s}_\theta(G_t; y_i)]\mathrm{d}t + g_t\, \mathrm{d}\bar{W}, \tag{9}$$

where $f_t$ is a drift coefficient, $g_t$ is a diffusion coefficient, and $\bar{W}$ is a standard Wiener process (Brownian motion). Starting from pure noise and integrating Eq. (9) down to $t = 0$ yields, for each client $k$, a set of generated graphs $\widetilde{\mathcal{D}}_k$ corresponding to the target classes, which augments minority labels and enhances structural diversity.

***PGM-guided sample filtering.*** Class-aware diffusion, however, does not guarantee that all generated graphs are useful. In practice, the generated graphs vary in quality and may deviate from the local data distribution. Furthermore, some graphs may be incompatible with the graph priors encoded in the PGM, leading to negative transfer when used directly for fine-tuning. To address this issue, we perform PGM-guided sample filtering: we measure proximity in the PGM's representation space and retain only those generated graphs that are closest to real graphs of the same class and correctly predicted by the downstream classifier.

Based on the original graphs $\mathcal{D}_k = \{(X, A)\}$ and the generated graphs $\widetilde{\mathcal{D}}_k = \{(\widetilde{X}, \widetilde{A})\}$ on client $k$, we compute graph representations $z = f_\phi(X, A)$ and $\tilde{z} = f_\phi(\widetilde{X}, \widetilde{A})$ by the PGM encoder $f_\theta$. For each generated graph, we define its minimum distance to all local graphs:

$$d_{\min}^{\mathrm{pre}}(\widetilde{X}, \widetilde{A}) = \min_{(X,A)\in\mathcal{D}_k} \| \tilde{z} - z \|_2. \tag{10}$$

Then, we rank all generated graphs in $\widetilde{\mathcal{D}}_k$ by $d_{\min}^{\mathrm{pre}}$ in ascending order, and, after sorting, we write the ordered set as $\widetilde{\mathcal{D}}_k = \{\widetilde{G}^{(1)}, \widetilde{G}^{(2)}, \ldots\}$. We select those with the $k$ smallest distances. The resulting candidate set is $\mathcal{C}_k = \{\widetilde{G}^{(m)}\}_{m=1}^{k}$. After obtaining the candidate set $\mathcal{C}_k$, we further apply a label-consistency filter. Let $\hat{y}_\psi(\widetilde{X}, \widetilde{A})$ denote the predicted label produced by the classifier head $c_\psi$. We retain only those candidates whose predicted labels match their target labels:

$$\mathcal{S}_k = \{\hat{y}_\psi(\widetilde{X}, \widetilde{A}) = \widetilde{y}\}. \tag{11}$$

After filtering, we construct the final local training set $T_k = D_k \cup S_k$.

This distance-and-consistency-based selection acts as a generation correction step. By measuring how close each generated graph is to the real local data in the PGM representation space, it filters out samples that deviate from the client distribution and the prior, preventing noisy augmentations from misleading the model. The label-consistency criterion further enforces semantic alignment with the task, favoring generated graphs near the decision boundary but on the correct side, thus providing informative, margin-enlarging examples. Overall, this strategy improves the alignment between the prior and the local data distribution, leading to more stable fine-tuning and reduced cross-client discrepancy.

### 4.3 Invariant subgraph extraction

In FGL, graphs often exhibit topology heterogeneity, i.e., differences in graph size across clients, which may hinder the adaptation of PGMs. Moreover, many substructures may be irrelevant or noisy to the downstream tasks. To mitigate task-irrelevant variations while retaining label-relevant signals, we extract invariant predictive subgraphs that are stable across clients. Making predictions on these subgraphs rather than full graphs suppresses client-specific nuisances and promotes more consistent PGM adaptation under heterogeneity.

***Subgraph extraction.*** Given a graph $G = (X, A)$, we first adapt a vanilla GCN [10] with parameters $\omega^{(gcn)}$ to encode each node $i$ into an embedding $h_i$. Based on these embeddings, we compute a node score for each node and an edge score for each pair of nodes:

$$\begin{aligned} [s^{(v)}(X, A)]_i &= \sigma(\mathrm{W}^{(v)} h_i) \in [0, 1], \\ [s^{(e)}(X, A)]_{ij} &= \sigma(\mathrm{W}^{(e)} [\, h_i \,\|\, h_j \,]) \in [0, 1], \end{aligned} \tag{12}$$

where $\mathrm{W}^{(v)}$ and $\mathrm{W}^{(e)}$ are trainable parameters that project node and edge embeddings to scalar scores, and $\sigma$ is the sigmoid function. With fixed thresholds $\tau_v, \tau_e \in (0, 1)$, we obtain mask matrices:

$$\begin{aligned} \mathrm{M}^{(v)}(X, A) &= \mathbb{I}[s^{(v)}(X, A) \geq \tau_v], \\ \mathrm{M}^{(e)}(X, A) &= \mathbb{I}[s^{(e)}(X, A) \geq \tau_e]. \end{aligned} \tag{13}$$

where $\mathbb{I}[\cdot]$ is the indicator function. Using these masks, we construct the masked graph inputs:

$$\widehat{X} = X \odot \mathrm{M}^{(v)}(X, A), \qquad \widehat{A} = A \odot \mathrm{M}^{(e)}(X, A), \tag{14}$$

where $\odot$ denotes the Hadamard product. Thus, the subgraph extraction can be seen as a trainable masker $\omega = (\omega^{(gcn)}, W^{(v)}, W^{(e)})$ applied to the original graphs. The obtained subgraph $\widehat{G} = (\widehat{X}, \widehat{A})$ is used to fine-tune the PGM $F_\theta$.

***Invariant regularizer.*** To further ensure the extracted subgraphs are invariant, i.e., they should preserve a consistent relationship with the target labels across clients, we draw inspiration from [4] and propose to add a global gradient penalty to the parameter of subgraph extraction.

Let the supervised fine-tuning loss on the extracted subgraphs be

$$\ell_{\text{task}}(X, A, y; \theta, \omega) = \mathrm{CE}\left(\mathrm{softmax}(F_\theta(\widehat{X}, \widehat{A})),\, y\right). \tag{15}$$

Define the local masker gradient of client $k$ and the global masker gradient as

$$\begin{aligned} g_w^{(l)} &= \mathbb{E}_{(X,A,y)\in T_k} \nabla_\omega \, \ell_{\text{task}}(X, A, y; \theta, \omega), \\ g_w^{(g)} &= \mathrm{Agg}(\{\, g_j \,\}_{j \in \mathcal{C}_r}), \end{aligned} \tag{16}$$

where $\mathcal{C}_r$ denotes the participating clients in round $r$. Then we add a $L_2$ regularizer between these two gradients:

$$\mathcal{R}_k^{\text{inv}}(\omega) = \| g_w^{(l)} - g_w^{(g)} \|_2. \tag{17}$$

In this way, the local objective is

$$\mathcal{L}_k(\theta, \omega) = \mathbb{E}_{(X,A,y)\in T_k} \ell_{\text{task}}(X, A, y; \theta, \omega) + \gamma \mathcal{R}_k^{\text{inv}}(\omega), \tag{18}$$

where the penalty weight $\gamma$ scales the local-global gradient alignment strength. Note that $\mathcal{R}_k^{\text{inv}}$ only affects the masker parameters $\omega$. As a result, the masker can mask irrelevant or noisy substructures and preserve subgraphs that exhibit invariant relationships with the target labels across clients.

## 5 Experiment

### 5.1 Experimental settings

**Datasets.** We evaluate our method on four representative graph datasets from [7], namely BACE, BBBP, HIV, and PPI. Cross-client heterogeneity is introduced via label-based and topology-based partitioning [12], where the degree of heterogeneity is controlled by a Dirichlet distribution with parameter $\alpha$.
**Baseline.** We select seven representative baselines from two categories: (1) FL methods including FedAvg [14], FedProx [11], and FedIIR [4]; (2) FGL methods including GCFL+ [23], FedStar [18], FedSSP [19], and FedVN [3]. For all FGL baselines, we use GIN [24] as the backbone.
**Implementation details.** The Dirichlet parameter $\alpha$ is selected from $\{0.1, 1, 10\}$, where a smaller $\alpha$ indicates stronger heterogeneity. For class-aware diffusion, we set the conditional control weights to $\lambda_X = 0.4$ and $\lambda_A = 0$ on BACE and BBBP, and $\lambda_X = 0.4$ and $\lambda_A = 0.2$ on HIV. We set $k = 15$ in the filtering step. The masking module is a two-layer GCN with 128 hidden dimensions. The node and edge thresholds are the same. We optimize the model with Adam using a learning rate of $10^{-3}$ and use ROC-AUC as the evaluation metric.

### 5.2 Overall performance

As is shown in Table 1, under different degrees of label heterogeneity, our method consistently outperforms all baselines. Under severe heterogeneity (i.e., $\alpha = 0.1$), our method achieves substantial performance gains, indicating that the class-conditional data generation effectively replenishes minority-class samples, while invariant subgraph extraction suppresses structural conflicts during aggregation. Moreover, PeFGL generally exhibits smaller standard deviations, demonstrating robustness under severe heterogeneity. Table 2 presents results under various degrees of topology heterogeneity and PeFGL also achieves the SOTA performance. When topology heterogeneity is severe (e.g., in HIV), traditional FGL models tend to capture topology-specific spurious correlations. In contrast, our

Table 1: Performance (%) of baselines under label heterogeneity.

| | BACE | | | BBBP | | | HIV | | | PPI | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Centralized | 86.39±0.86 | | | 95.83±0.37 | | | 77.87±1.95 | | | 65.15±1.88 | | |
| | 0.1 | 1 | 10 | 0.1 | 1 | 10 | 0.1 | 1 | 10 | 0.1 | 1 | 10 |
| Fedavg | 82.74±4.47 | 81.58±3.84 | 81.87±1.81 | 88.32±7.77 | 95.68±0.92 | 96.30±0.72 | 77.01±5.67 | 81.67±3.90 | 77.14±3.31 | 79.17±6.08 | 66.25±0.88 | 69.53±2.40 |
| FedProx | 80.25±2.97 | 80.17±5.66 | 83.32±2.28 | 89.04±7.89 | 95.53±0.77 | 96.47±0.32 | 75.28±5.89 | 74.23±1.13 | 72.60±3.56 | 64.20±1.45 | 64.54±2.87 | 64.65±0.84 |
| FedIIR | 73.63±2.13 | 80.35±1.96 | 82.84±0.08 | 75.61±1.16 | 94.17±1.67 | 95.02±0.37 | 67.68±8.41 | 73.73±3.74 | 74.84±4.25 | 59.80±1.29 | 60.81±1.70 | 61.40±1.36 |
| GCFL+ | 79.65±1.84 | 83.94±3.14 | 84.14±1.60 | 89.50±7.16 | 96.23±0.68 | 96.38±0.40 | 68.73±7.79 | 74.47±3.85 | 70.38±1.04 | 70.88±2.02 | 64.54±4.60 | 64.81±2.03 |
| Fedstar | 59.59±0.61 | 62.29±2.92 | 70.04±3.16 | 78.73±5.36 | 85.18±4.40 | 86.18±3.43 | 69.33±6.78 | 68.66±5.07 | 69.34±5.86 | 51.71±0.81 | 51.99±1.18 | 55.61±2.46 |
| Fedssp | 60.45±7.36 | 65.85±9.23 | 56.02±4.50 | 57.75±2.34 | 75.18±6.43 | 79.79±2.48 | 53.10±5.73 | 53.33±6.41 | 54.91±0.50 | 79.64±2.84 | 73.86±1.52 | 67.21±0.87 |
| Fedvn | 72.88±4.92 | 76.45±2.54 | 80.78±2.22 | 89.07±4.22 | 89.59±5.25 | 95.54±0.98 | 71.72±1.91 | 73.20±5.26 | 69.16±4.10 | 73.48±7.66 | 74.88±4.82 | 66.51±7.23 |
| PeFGL | **84.25±2.07** | **84.33±3.54** | **87.83±0.92** | **93.39±4.10** | **98.11±0.67** | **97.52±0.45** | **79.34±0.56** | **87.10±1.69** | **82.30±3.54** | **79.29±4.88** | **75.89±3.84** | **76.19±2.32** |

Table 2: Performance (%) of baselines under topology heterogeneity.

| | BACE | | | BBBP | | | HIV | | | PPI | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Centralized | 86.39±0.86 | | | 95.83±0.37 | | | 77.87±1.95 | | | 65.15±1.88 | | |
| | 0.1 | 1 | 10 | 0.1 | 1 | 10 | 0.1 | 1 | 10 | 0.1 | 1 | 10 |
| Fedavg | 82.43±0.57 | 83.61±0.63 | 83.58±0.22 | 94.97±0.77 | 93.46±2.97 | 95.96±0.42 | 70.73±1.34 | 71.62±1.52 | 71.63±2.84 | 60.43±2.02 | 63.95±1.15 | 63.57±1.92 |
| FedProx | 84.59±2.81 | 84.33±2.52 | 84.18±1.17 | 96.03±0.98 | 92.35±5.52 | 95.43±0.79 | 69.10±0.86 | 71.89±2.67 | 69.58±1.87 | 59.26±4.56 | 62.40±2.12 | 62.45±1.45 |
| FedIIR | 80.99±1.09 | 79.66±1.28 | 80.74±1.60 | 95.05±0.52 | 93.65±0.38 | 94.73±0.67 | 74.58±3.54 | 73.53±2.70 | 76.75±4.82 | 58.61±2.35 | 61.98±0.46 | 62.03±0.67 |
| GCFL+ | 82.53±3.37 | 84.37±1.73 | 84.29±2.57 | 95.95±0.76 | 92.23±5.71 | 96.10±0.13 | 72.40±3.01 | 72.93±2.99 | 74.16±4.40 | 59.82±3.30 | 64.25±1.93 | 64.44±1.91 |
| Fedstar | 70.79±1.52 | 66.69±1.18 | 69.95±2.30 | 84.37±2.08 | 84.10±2.61 | 85.91±1.23 | 71.21±3.67 | 71.58±3.41 | 70.59±2.34 | 54.98±2.04 | 56.37±4.48 | 56.27±3.31 |
| Fedssp | 63.14±4.78 | 62.95±3.20 | 62.63±4.50 | 83.18±2.69 | 78.03±1.96 | 80.74±1.57 | 61.92±1.30 | 58.08±1.84 | 61.00±1.42 | 56.49±0.64 | 56.39±1.15 | 59.32±1.19 |
| Fedvn | 75.93±0.64 | 79.24±2.36 | 75.20±3.40 | 93.64±1.15 | 92.62±1.70 | 94.12±1.29 | 74.80±3.43 | 69.72±2.38 | 67.41±2.53 | 57.43±0.61 | 56.58±1.51 | 53.74±1.76 |
| PeFGL | **86.79±0.11** | **87.12±0.88** | **88.75±1.02** | **96.37±0.12** | **94.18±2.58** | **96.42±0.84** | **76.77±3.39** | **77.48±2.65** | **79.03±3.75** | **68.41±2.79** | **71.76±3.07** | **71.96±1.53** |

invariant-subgraph constraint concentrates on learning structures shared among different topologies and thus shows great generalization capabilities. These findings further verify that aligning clients to the pre-trained prior and then fine-tuning the PGM allows the model to fit local distributions while maintaining global consistency.

### 5.3 Ablation study

To show the effect of each component in PeFGL, we conduct ablation studies with five variants: `ours-g-m` (federated fine-tuning of the pre-trained model only), `ours-g` (invariant subgraph extraction only), `ours-m` (minority-class generation only), `random_gen` (randomly selecting generated samples), and `ours` (PeFGL). Figure 3 and Figure 4 report the results under different types of heterogeneity. Among different $\alpha$, PeFGL achieves the best AUC. For `ours-g-m`, adding either single component brings consistent gains, indicating that minority-class generation and invariant structural regularization are both effective. `random_gen` performs worse than PeFGL and even inferior to the single-component variants, and in a few cases slightly worse than `ours-g-m`, showing that prior-guided selection is necessary to avoid including noisy synthetic samples. Under label heterogeneity, `ours-m` tends to yield larger improvements, whereas under topology heterogeneity, `ours-g` contributes more. It suggests that when label imbalance is severe, addressing the data-coverage gap is most useful, whereas when class coverage is sufficient, structural constraints become more critical.

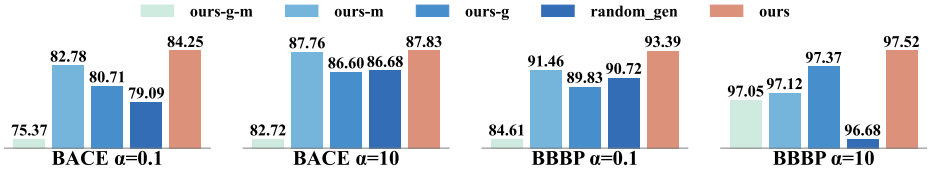


Fig. 3: Ablation study under label heterogeneity.

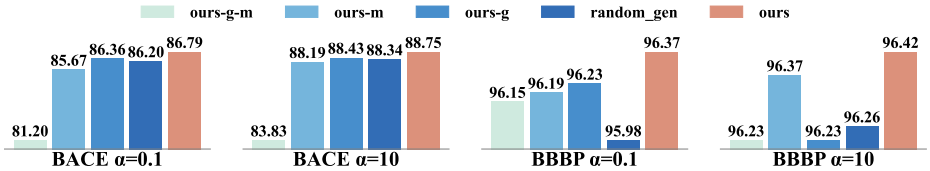


Fig. 4: Ablation study under topology heterogeneity.

### 5.4 Parameter analysis

We conduct parameter analysis regarding the mask threshold $\tau$ and the regularization weight $\gamma$, which is reported in Figure 5.

**The threshold of edge/node masks.** Since we set the node and edge thresholds to be identical, we use a unified threshold $\tau$ ($\tau_v = \tau_e = \tau$) to control the sparsity of the invariant subgraph. As shown in Fig. 5, under label heterogeneity, the AUC first increases and then decreases as $\tau$ grows. When $\tau$ is too small, the mask retains many client-specific edges and introduces noise, whereas overly large $\tau$ removes too many informative edges. Under topology heterogeneity, small thresholds are more stable, and progressively increasing $\tau$ leads to a clear degradation in AUC, consistent with the intuition that cross-client shared structures are relatively sparse and can be easily damaged by aggressive pruning.

**The weight of the regularization term.** For the regularization weight $\gamma$, performance under both label and topology heterogeneity is highest when $\gamma$ remains small and declines as $\gamma$ increases. Within this range, increasing $\gamma$ tends to improve performance, whereas larger values lead to a clear performance drop, with a sharper decline under topology heterogeneity. These results suggest that a small amount of regularization suffices to encourage cross-client alignment of invariant subgraphs, whereas larger penalties over-constrain the model, blur class-discriminative structure, and diminish the benefit of the pre-trained priors.

### 5.5 Results of existing methods with PGM

Table 3 reports results for existing heterogeneity-mitigation methods combined with a PGM, evaluated on three datasets with $\alpha = 0.1$. Under label heterogeneity, integrating the pre-trained PGM generally yields worse performance than training the same backbone without the pre-trained prior. Under topology heterogeneity, we observe minor gains only on BBBP. These results indicate

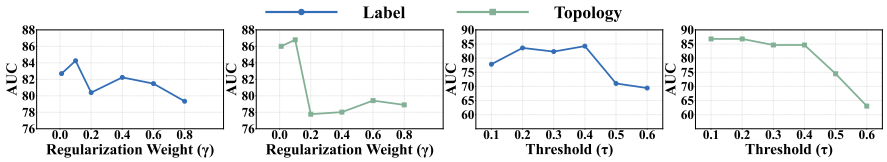


Fig. 5: Parameter sensitivity on BACE ($\alpha = 0.1$).

Table 3: Results w/ and w/o PGM ($\alpha = 0.1$).

| Method | BACE | | | | BBBP | | | |
|---|---|---|---|---|---|---|---|---|
| | Label | | Topology | | Label | | Topology | |
| | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ |
| FedIIR | 73.63±2.13 | 65.11±9.83 | 80.99±1.09 | 77.87±0.82 | 75.61±1.16 | 78.84±12.61 | 95.05±0.52 | 94.21±0.16 |
| FedProx | 80.25±2.97 | 70.77±9.87 | 84.59±2.81 | 81.59±0.64 | 89.04±7.89 | 81.71±16.52 | 96.03±0.98 | 96.20±0.61 |
| GCFL+ | 79.65±1.84 | 76.36±4.22 | 82.53±3.37 | 81.07±0.60 | 89.50±7.16 | 85.63±12.47 | 95.95±0.76 | 96.23±0.54 |
| Fedavg | 82.74±4.47 | 75.37±4.95 | 82.43±0.57 | 81.20±1.58 | 88.32±7.78 | 84.61±10.35 | 94.97±0.77 | 96.15±0.08 |

that addressing heterogeneity alone does not resolve the negative transfer when adapting pre-trained models in federated settings.

### 5.6 Results of different fine-tuning strategies

To assess the difference between classifier-only fine-tuning and full fine-tuning, we compare the two strategies in Figure 6 ($\alpha$=0.1). Full fine-tuning yields substantial improvements over classifier-only fine-tuning. This observation is consistent with our hypothesis: the pre-trained encoder carries source-domain priors that are misaligned with heterogeneous client distributions. Simply optimizing the classifier provides merely shallow adjustments and hardly mitigates the representation drift induced by label imbalance or topology shifts. These findings motivate our framework that builds on full fine-tuning and we additionally employ class-aware generation, prior-guided sample selection, and invariant subgraph constraints to enable more effective encoder adaptations.

### 5.7 Results on unseen graphs

As shown in Figure 7, we evaluate PeFGL on unseen graphs to show whether PeFGL truly learns transferable structural regularities rather than overfitting to client-specific biases. The results reveal that vanilla FedAvg generalizes poorly to unseen graphs, and naively federated fine-tuning a pre-trained graph model can be brittle, exhibiting unstable gains and even negative transfer. In contrast, PeFGL consistently shows stronger robustness, indicating that it effectively mitigates the prior–local mismatch during federated optimization and encourages the model to rely on more transferable patterns.

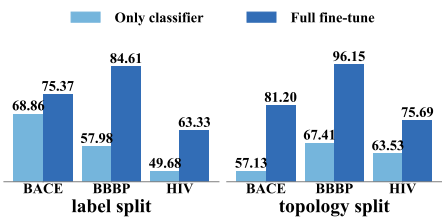


Fig. 6: Fine-tuning strategy comparison.

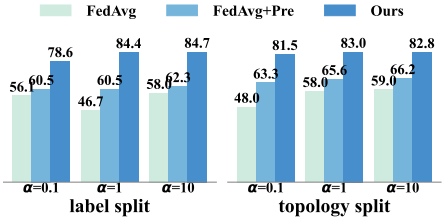


Fig. 7: Results on unseen graphs.

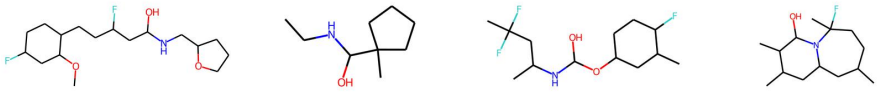

Fig. 8: Visualization of generated graphs.

### 5.8 Visualization

Figure 8 visualizes several molecular graphs generated by our diffusion model on the BACE dataset. The generated samples are chemically valid and include functional motifs, such as hydroxyl, amine, ether, and fluorinated groups, that are closely associated with BACE inhibitory activity. These functional groups serve as key determinants of molecular bioactivity. The generated graphs enrich the diversity of ring structures while preserving chemically meaningful substructures. Such augmentation enhances the representation of activity-related patterns and facilitates cross-client alignment in federated settings.

## 6 Conclusion

In this work, we propose PeFGL, a framework that adapts pre-trained graph priors to heterogeneous clients by coupling class-aware graph generation with invariant subgraph extraction. We apply the diffusion model to generate local graphs, which are then filtered by both the PGM prior and local distributions. Then, we design the invariant subgraphs extraction for fine-tuning. Evaluations on several real-world datasets against strong baselines achieve consistent improvements, and further analysis also demonstrates the effectiveness of PeFGL.

**Acknowledgments.** This work is supported in part by the National Natural Science Foundation of China (No. 62550138, 62192784, 62572064, 62472329, 62402263), JST PRESTO JPMJPR23P5, JST CREST JPMJCR21M2, and JST NEXUS JPMJNX25C4.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References


1. Agarwal, A., Rezagholizadeh, M., Parthasarathi, P.: Practical takes on federated learning with pretrained language models. In: Findings of the Association for Computational Linguistics: EACL 2023. pp. 454–471 (2023)
2. Cao, Y., Xu, J., Yang, C., Wang, J., Zhang, Y., Wang, C., Chen, L., Yang, Y.: When to pre-train graph neural networks? from data generation perspective! In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 142–153 (2023)
3. Fu, X., Chen, Z., He, Y., Wang, S., Zhang, B., Chen, C., Li, J.: Virtual nodes can help: Tackling distribution shifts in federated graph learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 16657–16665 (2025)
4. Guo, Y., Guo, K., Cao, X., Wu, T., Chang, Y.: Out-of-distribution generalization of federated learning via implicit invariant relationships. In: International Conference on Machine Learning. pp. 11905–11933. PMLR (2023)
5. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
6. Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., Tang, J.: Graphmae: Self-supervised masked graph autoencoders. In: Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining. pp. 594–604 (2022)
7. Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for pre-training graph neural networks. arXiv preprint arXiv:1905.12265 (2019)
8. Hu, Z., Dong, Y., Wang, K., Chang, K.W., Sun, Y.: Gpt-gnn: Generative pre-training of graph neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1857–1867 (2020)
9. Jhunjhunwala, D., Sharma, P., Xu, Z., Joshi, G.: Initialization matters: Unraveling the impact of pre-training on federated learning. arXiv preprint arXiv:2502.08024 (2025)
10. Kipf, T.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
11. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proceedings of Machine learning and systems **2**, 429–450 (2020)
12. Li, X., Zhu, Y., Pang, B., Yan, G., Yan, Y., Li, Z., Wu, Z., Zhang, W., Li, R.H., Wang, G.: Openfgl: A comprehensive benchmark for federated graph learning. arXiv preprint arXiv:2408.16288 (2024)
13. Lit, Z., Sit, S., Wang, J., Xiao, J.: Federated split bert for heterogeneous text classification. In: 2022 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2022)
14. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
15. Nguyen, J., Wang, J., Malik, K., Sanjabi, M., Rabbat, M.: Where to begin? on the impact of pre-training and initialization in federated learning. arXiv preprint arXiv:2206.15387 (2022)
16. Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., Tang, J.: Gcc: Graph contrastive coding for graph neural network pre-training. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1150–1160 (2020)

17. Sun, F.Y., Hoffmann, J., Verma, V., Tang, J.: Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. arXiv preprint arXiv:1908.01000 (2019)
18. Tan, Y., Liu, Y., Long, G., Jiang, J., Lu, Q., Zhang, C.: Federated learning on non-iid graphs via structural knowledge sharing. In: Proceedings of the AAAI conference on artificial intelligence. vol. 37, pp. 9953–9961 (2023)
19. Tan, Z., Wan, G., Huang, W., Ye, M.: Fedssp: Federated graph learning with spectral knowledge and personalized preference. Advances in Neural Information Processing Systems **37**, 34561–34581 (2024)
20. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
21. Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. arXiv preprint arXiv:1809.10341 (2018)
22. Wan, C., Li, Y., Li, A., Kim, N.S., Lin, Y.: Bns-gcn: Efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling. Proceedings of Machine Learning and Systems **4**, 673–693 (2022)
23. Xie, H., Ma, J., Xiong, L., Yang, C.: Federated graph classification over non-iid graphs. Advances in neural information processing systems **34**, 18839–18852 (2021)
24. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
25. Yan, B., Cao, Y., Wang, H., Yang, W., Du, J., Shi, C.: Federated heterogeneous graph neural network for privacy-preserving recommendation. In: Proceedings of the ACM Web Conference 2024. pp. 3919–3929 (2024)
26. Yan, B., He, S., Yang, C., Liu, S., Cao, Y., Shi, C.: Federated graph condensation with information bottleneck principles. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 12990–12998 (2025)
27. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)
28. Yao, Y., Jin, W., Ravi, S., Joe-Wong, C.: Fedgcn: Convergence-communication tradeoffs in federated training of graph convolutional networks. Advances in neural information processing systems **36**, 79748–79760 (2023)
29. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. Advances in neural information processing systems **33**, 5812–5823 (2020)
30. Zhang, H., Shen, T., Wu, F., Yin, M., Yang, H., Wu, C.: Federated graph learning–a position paper. arXiv preprint arXiv:2105.11099 (2021)
31. Zhang, J., Liu, Y., Hua, Y., Cao, J.: An upload-efficient scheme for transferring knowledge from a server-side pre-trained generator to clients in heterogeneous federated learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12109–12119 (2024)
32. Zhang, K., Yang, C., Li, X., Sun, L., Yiu, S.M.: Subgraph federated learning with missing neighbor generation. Advances in neural information processing systems **34**, 6671–6682 (2021)
33. Zhou, P., Chen, C., Liu, W., Liao, X., Shen, W., Xu, J., Fu, Z., Wang, J., Wen, W., Zheng, X.: Fedgog: Federated graph out-of-distribution generalization with diffusion data exploration and latent embedding decorrelation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 22965–22973 (2025)