FedGKD: Unleashing the Power of Collaboration in Federated Graph Neural Networks

Qiying Pan
Shanghai Jiao Tong University
sim10_arity@sjtu.edu.cn

Ruofan Wu
Ant Group
ruofan.wrf@antgroup.com

Tengfei Liu
Ant Group
aaron.ltf@antgroup.com

Tianyi Zhang
Ant Group
zty113091@antgroup.com

Yifei Zhu
Shanghai Jiao Tong University
yifei.zhu@sjtu.edu.cn

Weiqiang Wang
Ant Group
weiqiang.wwq@antgroup.com

Abstract

Federated training of Graph Neural Networks (GNN) has become popular in recent years due to its ability to perform graph-related tasks under data isolation scenarios while preserving data privacy. However, graph heterogeneity issues in federated GNN systems continue to pose challenges. Existing frameworks address the problem by representing local tasks using different statistics and relating them through a simple aggregation mechanism. However, these approaches suffer from limited efficiency from two aspects: low quality of task-relatedness quantification and inefficacy of exploiting the collaboration structure. To address these issues, we propose FedGKD, a novel federated GNN framework that utilizes a novel client-side graph dataset distillation method to extract task features that better describe task-relatedness, and introduces a novel server-side aggregation mechanism that is aware of the global collaboration structure. We conduct extensive experiments on six real-world datasets of different scales, demonstrating our framework's outperformance.

1 Introduction

Federated training of Graph Neural Networks (GNN) has gained considerable attention in recent years due to its ability to apply a widely used privacy-preserving framework called Federated Learning (FL) [31, 21] to GNN training. This approach facilitates collaborative training among isolated datasets while preserving data privacy. The protocol generally follows a typical federated learning approach, whereby the server broadcasts a global GNN model to each client. Upon receiving the model, each client trains the model using its local graph and transmits its local model to the server, which then aggregates the local weights to obtain a new global model and initiates another round of the broadcast-training-aggregation procedure. Throughout the process, the client-side graphs are kept locally, thus safeguarding privacy. Federated GNNs have successfully addressed the challenge of isolating graph data in real-life scenarios where organizations and corporations collect their private graphs, restricting others from accessing them while seeking a collaborative training approach to improve their personalized model performance.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

However, the effectiveness of federated GNN frameworks is limited by graph heterogeneity issues. The isolated graph datasets exhibit significant variation, and thus, simple aggregation of all local models leads to a degradation of model performance [33]. The recent developments of personalized federated learning (PFL) [36, 1, 43, 3] have explored mechanisms that adapt to client heterogeneity by allowing each client to train their personalized model while encouraging inter-client collaboration through properly handling the collaboration structure among clients.

Although the PFL framework is effective in mitigating statistical heterogeneity issues, it is noteworthy that graph heterogeneity issues are more severe than that in the independent setting. This is primarily due to the complex nature of graphs that incorporate topology information. In particular, graph heterogeneity encompasses not only the distributional discrepancy of node features but also the presence of disparate graph structures [33]. To address this challenge, a range of graph-related solutions have been proposed. However, the existing solutions for graph heterogeneity issues suffer from limited efficiency due to two reasons:

Quality of task-relatedness quantification: It has been shown in recent studies [3, 48] that high-quality characterizations of the collaboration structure among clients are crucial for the generalization performance in PFL. Existing works on PFL over graphs rely on weights and gradients [42] or graph embeddings with random input graphs [2]. The former involves similarity computation of high-dimensional random vectors, which may be imprecise due to the curse of dimensionality [15], while the latter one discards information related to local labels and losses expressivity.

Efficacy of exploiting collaboration structure: Another reason for unsatisfactory efficiency is the inadequate handling of the collaboration structure, which is typically reflected in the personalized aggregation mechanisms. Previous works base their aggregation procedures on *pairwise relationships* among client tasks [2], which only exploit inter-client task relatedness *locally*. It is therefore of interest to explore mechanisms that operate from a *global* point of view in a principled way.

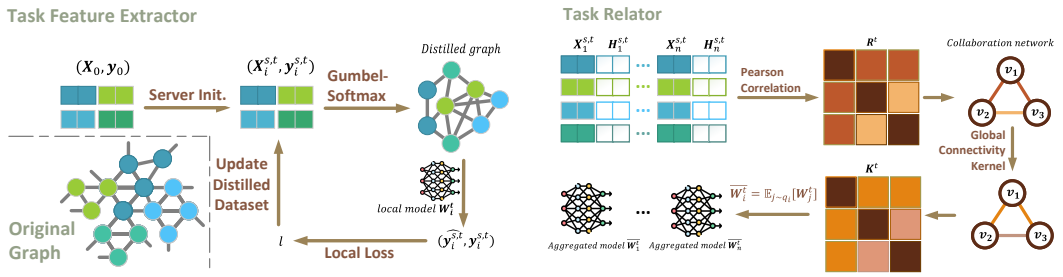
To address the problems, we propose a novel **Fed**erated **G**raph Neural Network with **K**ernelized aggregation of **D**istilled information (**FedGKD**) that achieves better utilization of the collaboration among clients. This framework comprises two modules: A **task feature extractor** that utilizes a novel graph data distillation method and a **task relator** that aggregates local models through a mechanism that is aware of the global property of the collaboration structure inferred from the outputs of the task feature extractor. Specifically, the task feature extractor devises a novel *dynamic graph dataset distillation mechanism* that represents each local task by distilling local graph datasets into size-controlled synthetic graphs at every training round, enabling efficient similarity computation between clients while sufficiently incorporating local task information. The task relator first constructs a collaboration network from the distilled task features and relates tasks through a novel aggregation mechanism based on the network's *global connectivity*, which measures the task-relatedness in a global sense. As a summary, our contributions are:

- We propose a task feature extractor based on a novel dynamic graph data distillation method, representing each local task with a distilled synthetic graph generated from all the local model weights trained at each round. The task features extracted from distilled graphs contain both data and model information, while also allowing for efficient evaluation of task-relatedness.
- We propose a task relator that constructs a collaboration network from the distilled graphs and relates tasks by operating a novel kernelized attentive aggregation mechanism upon local weights that encodes the global connectivity of the collaboration network.
- We conduct extensive experiments to validate that our framework consistently outperforms state-of-the-art personalized GNN frameworks on six real-world datasets of varying scales under both overlapping and non-overlapping settings.

2 Related Work

2.1 Federated Graph Representation Learning

In [33], the authors showed that naively applying FedAvg to distributed GNN training will result in irreducible error under distinct client-side graph structures, which hampers convergence. A recent line of work has been attempting to adopt personalization strategies for federated learning of graph neural networks. For instance, [37] uses client-side adaptation by sharing only a sub-structure of the



(a) Task feature extractor (b) Task relator

Figure 1: Overview of two modules in the proposed FedGKD framework.

client-side GNN. [47] equips each client with an auxiliary neighborhood generation task. [42] applies a server-side adaptation strategy that dynamically clusters clients using intermediate gradient updates. Moreover, [2] combines client-side and server-side adaptation methods and measures task similarity using GNN outputs based on a common input random graph.

2.2 Personalized Federated Learning

The learning procedures of homogeneous federated learning [31, 21] are often considered as special forms of distributed optimization algorithms like local SGD [41]. However, these methods have been shown to suffer from client heterogeneity in terms of both convergence [22] and client-side generalization [7, 48]. Personalized federated learning approaches have primarily focused on addressing the latter issue by incorporating adaptation strategies that can be deployed at the client side, server side, or both. **Client-side adaptation** methods typically utilize parameter decoupling paradigms that enable flexible aggregation of partial parameters [1, 32] or control the optimization of local objectives by regularizing towards the global optimum [27]. However, these methods often overlook the overall collaboration structure among the clients [3, 48]. It has been shown that with a correctly-informed collaboration structure that precisely describes the task-relatedness between clients, simple procedures can achieve minimax optimal performance [48]. On the other hand, **server-side adaptation** methods aim to measure the task-relatedness among clients and derive refined aggregation mechanisms. [3] utilize tools from transfer learning theory to conduct an estimating procedure that clusters clients into subgroups. [46, 43] propose to optimize collaboration among clients on-the-fly by solving a quadratic program at the server-side during each aggregation step. It is important to note that server-side adaptation methods often involve the transmission of additional information other than the model parameters.

2.3 Dataset Distillation

The method of dataset distillation [40] was originally proposed as a way to improve training efficiency by distilling a large dataset into a significantly smaller one while keeping model performance almost intact. Later developments generalized the approach to graph-structured data [20, 19]. A notable property of dataset distillation is that the distilled datasets are observed to exhibit good privacy protection against empirically constructed adversaries [9]. This empirical property has also lead to innovations in one-shot federated learning [49], which is very different from the setups in PFL and is considered an orthogonal application.

3 Preliminaries

3.1 Graph Neural Network

Consider a graph $G = (V, E)$, where V represents the node set and E represents the edge set. The graph is associated with a node feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times D}$. We can use Graph Neural Networks (GNN) to embed nodes in the graph with low dimensional vectors. An L-layer GNN in the message passing form [12, 44] is recursively defined in (1), where $\mathbf{h}_u^l$ represents the embedding of node u output from the $l-$th GNN layer, UPD is a function that generates embeddings based on the former

layer outputs, $AGGR$ is a function that aggregates the embeddings together and $\mathcal{N}(u)$ represents the set of neighboring nodes of node u.

$$\mathbf{h}_u^l = UPD^l[\mathbf{h}_u^{l-1}, AGG^l(\{\mathbf{h}_v^{l-1} : v \in \mathcal{N}(u)\})]. \tag{1}$$

In general, the outputs of $L-$th GNN layers are passed through a customized READOUT layer to accomplish various graph-related tasks. For instance, in node classification tasks, a simple READOUT layer can be selected as a linear layer with the number of categories as the output dimension.

3.2 Federated Learning

Federated Learning (FL) was introduced to address data isolation issues while preserving privacy [31]. It enables collaborative training among clients without exposing raw datasets. A typical FL framework consists of three stages: (1) *model initialization*, where the server broadcasts initial model weights to all clients; (2) *local training*, where each client trains a local model using the initial model weights and its own dataset, and uploads the local model to the server; (3) *global aggregation*, wherein the server aggregates the local models into one or more new models and broadcasts the result to each client to initiate the next training round. The FL procedure typically alternates between stage (2) and (3) until convergence.

4 Problem Formulation

A personalized federated GNN framework aims to collaboratively learn local GNN models in a privacy-preserving manner. This allows the local models to fit their respective local datasets while leveraging information from other clients to improve sample efficiency. In the system, there are n clients and one server. Each client stores a local graph dataset $G_i = (V_i, E_i, \mathbf{X}_i)$, where V_i represents the node set, E_i represents the edge set, $\mathbf{X}_i \in \mathbb{R}^{|V_i| \times D}$ contains the node features, and $[n]$ denotes the set of positive integers from 1 to n. Within the system, the n clients train n GNN models $f(G_i; \mathbf{W}_i), i \in [n]$, with the same structure but different parameters $\mathbf{W}i, i \in [n]$. Additionally, we assume that the prediction function $f = g \circ h$ is composed of an L-layer message passing GNN module h with a hidden dimension of d, and a READOUT module g that maps the node embedding extracted by h to downstream task predictions. The goal of a personalized federated GNN framework is formulated as

$$\min_{\mathbf{W}_i \in \Omega_i, i \in [n]} \sum_{i=1}^{n} \mathcal{L}(f(G_i; \mathbf{W}_i), \mathbf{y}_i) \tag{2}$$

where $\mathcal{L}$ is the loss function, and $\mathbf{y}_i$ contains all the labels belonging to client i. To encourage collaboration among clients, the parameter spaces $\Omega_i, i \in [n]$, are usually assumed to be related [10]. The precise structure of this relation is sometimes implicit and instead reflected in the optimization procedure [6]. Under the graph representation learning setup, we require Ω_i to capture the relatedness between corresponding tasks, taking into account the topological structure of the graph [13], as well as feature and label information. Additionally, we impose an extra constraint to ensure that local models do not deviate significantly from each other. This is achieved by adding a proximal regularization term that prevents overfitting to local data, which has been shown to benefit many federated learning procedures [28, 27].

5 Design

5.1 Overview

In this section, we provide a detailed introduction to our personalized federated graph learning framework, which aims to address two major problems:

- How to extract task features from the local dataset G_i and local model parameters $\mathbf{W}_i$?
- How to relate local tasks with each other using the task features to aggregate $\mathbf{W}_i$?

To address the first question, we propose a feature extractor based on dataset distillation, as illustrated in Fig. 1a, that captures all the information within the local model. The feature extractor generates a small graph in each round based on the current local model weights. To mitigate graph heterogeneity,

the server distributes a common initial graph to all clients, preventing significant deviations among the distilled graphs.

To address the second question, we draw insights from recent advancements in kernel formulations of self-attention [38, 5]. We view the personalized aggregation process as an attentive mechanism operating on the *collaboration network* among clients. We observe that several contemporary aggregation schemes overlook the global task relatedness. Leveraging tools from kernel theory, we derive a refined aggregation scheme based on exponential kernel construction that effectively incorporates global information, as shown in Fig. 1b.

5.2 Task Feature Extractor

5.2.1 Motivation

It is well known in the theory of multi-task learning [14, 10] that correct specifications of task relatedness may fundamentally impact the model performance, which has also been recently discovered in PFL [48]. In hindsight, the ideal characterization of a (local) graph representation learning task might be either the *joint* distribution of the local graph, feature and label variables; or the Bayes optimal learner dervied from the joint distribution [48]. However, none of this information are available during FL, and various surrogates have been proposed in the context of graph PFL that extracts *task features* from the (local) empirical distribution and the learned model.

The most ad-hoc solution is to use weights [29, 46] and gradients [34], which are typically high-dimensional (random) vectors. However, computing their relations using metrics like Euclidean or cosine similarity can be unreliable due to the curse of dimensionality phenomenon [15], as empirically validated in [2]. As a notable state-of-the-art model, FedPUB [2] uses low-dimensional graph embeddings that are produced by passing a shared random graph between clients. However, since the embedding computation only involves message passing GNN layers, the resulting embeddings are *incomplete*, as they fail to represent the READOUT layer that follows these GNN layers. The READOUT layer encodes label-related information. This limitation is significant when two datasets share similar graph distributions but have divergent label distributions. This can result in two local models with similar GNN layer weights but different READOUT layer weights. In such cases, the embeddings, which are outputs of similar GNN layers, cannot distinguish between the two datasets. We conducted a small experiment to validate this point, which is elaborated in Appendix C.

To address the challenges encountered in PFL frameworks, we leverage graph dataset distillation, a method that simultaneously compresses the local data distribution and the learned local model into a size-controlled small dataset that is comparable across all client tasks. In the following sections, we will introduce dataset distillation and explain how we incorporate it into our framework.

5.2.2 Dataset Distillation

Dataset distillation [40] (DD) is a centralized knowledge distillation method that aims to distill large datasets into smaller ones. For client i, a distilled dataset $(G_i^s, \mathbf{y}_i^s)$ is defined such that a neural network model trained on G_i^s can achieve comparable performance to the one trained on the original dataset G_i, as formulated in (3).

$$\min_{G_i^s, \mathbf{y}_i^s} \mathcal{L}(f(G_i; \mathbf{W}_i^s), \mathbf{y}_i) \quad \text{s.t. } \mathbf{W}_i^s = \min_{\mathbf{W}_i'} \mathcal{L}(f(G_i^s; \mathbf{W}_i'), \mathbf{y}_i^s) \tag{3}$$

According to previous studies [20], many datasets can be distilled into condensed ones with sizes that are only around 1% of the original dataset while still preserving model performance. Moreover, it has been empirically reported that distilled datasets offer good privacy protection [9].

Based on this observation, we propose using *statistics of the distilled local datasets* as features that describe local tasks and obtain task-relatedness by evaluating the similarities between distilled dataset characteristics. As a straightforward adaptation of vanilla DD to federated settings, we may conduct isolated distillation steps *before* the federated training and fix the estimated task-relatedness during federated training. This strategy could be implemented using off-the-shelf DD algorithms on graphs [20, 19]. However, the quality of the distilled local datasets may be affected by (local) sample quality and quantity. Since PFL approaches typically improve local performance, we propose a refinement of the aforementioned *static distillation* strategy that allows clients to distill their local datasets *during* the federated training process, resulting in a series of distilled datasets $G_i^{s,t}, \mathbf{y}_i^{s,t} t \in [T]$ for each

client i, with its corresponding distillation objective at round t being:

$$\min_{G_i^{s,t}, \mathbf{y}_i^{s,t}} \mathcal{L}(f(G_i^{s,t}; \mathbf{W}_i^t), \mathbf{y}_i^{s,t}). \tag{4}$$

Apart from its capability to adapt to the federated learning procedure, the objective (4) is computationally more efficient than the vanilla DD objective (3) as it avoids the bi-level optimization problem, which is difficult to solve [40]. Instead, the objective (4) leverages the strength of the federated learning process, which usually produces performative intermediate results after a few rounds of aggregations. We refer to (4) as a *dynamic distillation* strategy. Next, we present a detailed implementation of the proposed dynamic distillation procedure.

5.2.3 Implementation

There are two algorithmic goals regarding the implementation of (4): Firstly, the distilled datasets should allow efficient similarity comparisons. Secondly, the problem should be efficiently solved so that the extra computation cost for each client is controllable. Note that both goals are non-trivial since the optimization involves a graph-structured object that is not affected by permutations, resulting in alignment issues when performing similarity computation. The solution is detailed in Algorithm 2 in Appendix A. Specifically, at each round $t \in [T]$, the size of the distilled graph across all clients will be fixed at $m \times C$, where m represents the number of representative nodes in each category. The server first initializes node features $\mathbf{X}_0$, with each entry drawn independently from a standard Gaussian distribution $\mathcal{N}(0, 1)$. The initial labels $\mathbf{y}_0$ are set to ensure that there are m nodes belonging to each category. The tuple $(\mathbf{X}_0, \mathbf{y}_0)$ is broadcast to each client as the initial value of their local objectives, while the construction of the distilled graph structure is left to the clients' side to reduce communication cost. This common initialization technique alleviates the alignment issue between distilled graphs.

After each client receives the initial features $\mathbf{X}_0$ and labels $\mathbf{y}_0$, it begins to update the features and labels. Since directly optimizing the graph structure (i.e., among the space of possible binary matrices) is computationally intractable, we use the following simple generative model that describes the relationship between node features and edge adjacency for the distilled graph: For a pair of nodes (regarding the distilled graph) u and v with features $\mathbf{x}_u^s$ and $\mathbf{x}_v^s$, the probability of them being adjacent is given by

$$\mathbb{P}[\mathbf{A}_{uv}^s = 1] = \frac{e^{\langle \mathbf{x}_u^s, \mathbf{x}_v^s \rangle - \gamma}}{1 + e^{\langle \mathbf{x}_u^s, \mathbf{x}_v^s \rangle - \gamma}}, \tag{5}$$

where $\gamma > 0$ is a hyperparameter that controls edge sparsity. [1] Construction of the distilled graph involves sampling from the above distribution, which is not differentiable. Hence we adopt the Gumbel-softmax mechanism [18, 30] to generate approximate yet differentiable samples. In particular, for each u, v, we first draw two independent samples ω and ω' from the standard Gumbel distribution. Next, we compute the following approximation:

$$p_{uv}(\tau_g) = \frac{e^{\left(\langle \mathbf{x}_u^s, \mathbf{x}_v^s \rangle - \gamma + \omega - \omega'\right)/\tau_g}}{1 + e^{(\langle \mathbf{x}_u^s, \mathbf{x}_v^s \rangle - \gamma + \omega - \omega')/\tau_g}}, \tag{6}$$

which adopts a distribution limit $\lim_{\tau_g \to 0} p_{uv}(\tau_g) \overset{d}{=} \mathbf{A}_{uv}^s$. In practice, we use the straight-through trick [18] to obtain discrete samples from (6) while allowing smooth differentiation. We denote the distilled graph as $G^s = (\mathbf{X}^s, \mathbf{P})$ with $\mathbf{P}$ being the (approximated) adjacency matrix, with each entry derived from (6).

We utilize the local model weights to assess how well the distilled dataset fits the model and update $\mathbf{X}^s$ and $\mathbf{y}^s$ based on the same classification loss as each client's local learning objective. In practice, we have found that a few steps of gradient updates suffice for the learning performance. After obtaining the distilled graph, we extract task features $M_{i\, i \in V_c}^t$ for client i at round t as follows:

$$\mathbf{M}_i^t = \left[\mathbf{X}_i^{s,t} \,\middle\|\, \mathbf{H}_i^{s,t}\right], \quad \mathbf{H}_i^{s,t} := h(\mathbf{G}_i^{s,t}, \mathbf{W}_i^t). \tag{7}$$

Note that although the distilled labels are not included in the task feature, the label information is fused into $\mathbf{X}^s$ during the distillation process. We will present an empirical study regarding other potential choices of task feature maps in section F.5.

[1] This construction is inherently *homophilic*. In principle, one could propose more sophisticated generative mechanisms with learnable parameters, but this may increase the computational cost of distillation. Experimentally, we have found this simple construction to be quite effective.

5.3 Task relator

5.3.1 Motivation

We represent the estimated relationship among tasks using a *time-varying collaboration network* $G_c^t = (V_c, \mathbf{R}^t)$, where $t \in [T]$, $V_c = [n]$, and $\mathbf{R}^t \in \mathbb{R}^{n \times n}$ represents the time-dependent task relation matrix. The entry r_{ij}^t measures the task-relatedness between client i and client j, obtained by computing similarities of their corresponding task features $\mathbf{M}_i^t$ and $\mathbf{M}_j^t$. This idea has been adopted in some recent PFL proposals [6, 46].

Since the matrix $\mathbf{R}^t$ encodes pairwise relationships among client tasks, it offers great flexibility in defining personalized aggregation protocols. We formulate the protocols as the following expectation:

$$\overline{\mathbf{W}_i^t} \leftarrow \mathbb{E}_{j \sim q_i} \left[\mathbf{W}_j^t \right] \tag{8}$$

where q_i is a client-specific distribution over $[n]$, with a trivial case of uniform distribution that corresponds to the aggregation rule in FedAVG. The above formulation is closely connected to the self-attention mechanism [39]. In particular, inspired by recent developments that generalize self-attention using kernel theory [38], we parameterize q_i using a kernel-induced distribution:

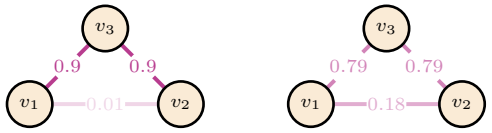
$$q_i[j] = \frac{k(i,j)}{\sum_{j' \in [n]} k(i,j')}, \tag{9}$$

where $k(\cdot, \cdot)$ is a kernel function.

The most straightforward choice would be the softmax kernel [8] that uses the exponentiated edge weights $k(i,j) = e^{r_{ij}}$. However, this method disregards other weights, resulting in a kernel function that only takes *local connectivity* in the collaboration network into account, overlooking *global connectivity*. We illustrate this point using the example in Fig. 2, where a collaboration network with three vertices has weighted links of $r_{12} < r_{23} = r_{13}$. If we directly use r_{ij} and normalize them, the average local model weights $\overline{\mathbf{W}_1^t}$ will be very close to $\mathbf{W}_3^t$ and far from $\mathbf{W}_2^t$. However, the relation between node 1 and 2 is much stronger than what the quantity r_{12} indicates, as they are also linked by another two-hop path comprising two heavily-weighted edges $(1,3)$ and $(2,3)$. A recent work [6] attempts to capture information beyond local task pairs by incorporating a GNN-like mechanism over a sparsified collaboration network. This approach aims to integrate more information through a few rounds of message passing. While the approach in [6] extends the scope of similarity evaluation, it still operates in a *local sense* due to the finiteness of message passing rounds and the inherent limitation of oversmoothing phenomenon. To address this limitation, our framework proposes a novel kernel function that incorporates all the global connectivity. Specifically, our kernel extracts connectivity at hops from 1 to infinity while favoring connectivity with fewer hops.

5.3.2 Construction of the task relator

According to the previous discussions, implementing the task relator involves the design of two modules: A *collaboration graph construction procedure* based on the extracted task features $\{M_i^t\}_{i \in V_c}$ and a *global-connectivity-aware aggregation mechanism*.



(a) local connectivity (b) global connectivity

Figure 2: Comparison of local and global connectivity

To form the collaboration graph, we use the following feature-wise average correlation as the pairwise task-relatedness:

$$r_{ij}^t = \frac{1}{d+D} \sum_{k=1}^{d+D} \mathsf{corr} \left(\mathbf{M}_i^t[:,k], \mathbf{M}_j^t[:,k] \right), \tag{10}$$

where corr stands for Pearson's correlation coefficient[26]. Next we discuss the construction of the global-connectivity-aware aggregation mechanism. Inspired by the property of exponential kernels

[25] that translates local structure into global ones, we use an element-wise exponentiated matrix exponential with two temperature parameters τ and τ_s:

$$k(i,j) = e^{\tau_s s_{ij}}, \mathbf{S} = \{s_{ij}\}_{i\in V, j\in V} = e^{\tau \mathbf{R}} = \sum_{\kappa=1}^{\infty} \frac{1}{\kappa!}(\tau \mathbf{R})^{\kappa}. \tag{11}$$

From the right hand side of (11), we may interpret $s_{i,j}$ as encoding the relatedness of client i and j via conducting infinite rounds of message passing, thereby reflecting their *global connectivity* structure. Then, $k(i,j)$ scales the global connectivity into $\mathbb{R}^+$ range for further normalization into the client-specific distribution q_i over $[n]$. The parameter τ is used to control the level of personalization, where $\tau \to 0$ indicates no personalization (FedAVG) and $\tau \to \infty$ indicates local training. The parameter τ_s strikes a balance between the contribution of local and global information. It is worth noting that there are other notions of global connectivity measures, such as effective resistance [4], which are also applicable. However, here we stick to the matrix exponential due to its loose requirements of arbitrary local similarity $\mathbf{R}$. We will prove in appendix G that the function k in (11) is a valid kernel over the domain $[n]$.

6 Experiments

This section presents the empirical analysis of our framework, which includes a performance comparison, convergence analysis, and multiple ablation studies.

6.1 Experiment Setup

6.1.1 Datasets

We have tested the performance on six different graph datasets of varying scales: Cora, CiteSeer, PubMed, Amazon Computer, Amazon Photo, and Ogbn arxiv [45, 35, 17]. To split each graph into subgraphs, we employed the Metis graph partition algorithm following the setup in [2]. Each client stores one subgraph from the original graph. We have conducted a node classification task by sampling the vertices into training, validation and testing vertices according to ratio 0.3, 0.35 and 0.35 before splitting. Appendix D.1 provides detailed descriptive statistics of the datasets.

6.1.2 Baselines

We compared our framework with eight different baselines, categorized into four types: (1) **Local**, which serves as the standard baseline without federated learning; (2) Two traditional FL baselines, including **FedAvg** [31] and **FedProx** [28]; (3) One state-of-the-art personalized FL baseline, **FedPer** [1]; (4) Four state-of-the-art personalized federated GNN baselines, including **FedPub** [2], **FedSage** [47], **GCFL** [42], and **FedStar** [37]. For detailed introductions of the baselines, please refer to Appendix D.2.

6.2 Implementation Details

We utilize a two-layer GCN [24] followed by a linear READOUT layer. The dimension of the embeddings is set to 128. To optimize learning, we employ the Adam optimizer with weight decay 10^{-6} [23]. The smoothing parameter τ_g in Gumbel-softmax is set to 1, following the technique in [18]. To monitor the training progress, we use an early-stop mechanism. If the validation accuracy decreases for 20 consecutive rounds, the FL framework stops immediately. Each experiment is conducted over 3 runs with different random seeds. Implementation of all methods is done using PyTorch Geometric [11] on an NVIDIA Tesla V100 GPU. For further details, please refer to Appendix D.3.

6.3 Results

6.3.1 Performance Comparison

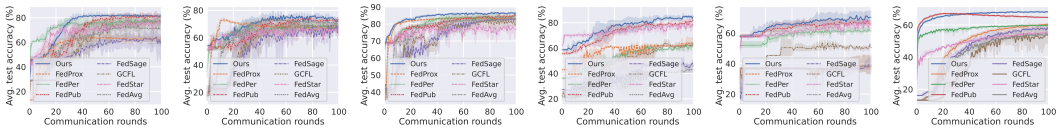
We evaluate the node classification performance of various frameworks using six real-world datasets of differing scales. Table 1 present the average test accuracy and its standard deviation for non-overlapping settings. As for the results on overlapping settings, please refer to Table 5 in Appendix

E. Our framework consistently outperforms all other methods across all datasets. Traditional FL baselines, such as FedAvg and FedProx, which lack local adaptation, are consistently inferior to our Local framework in multiple scenarios. Despite using local task statistics for personalization, the feature extraction schemes of FedPer, FedStar, FedSage, and GCFL are less effective and perform worse than our framework. FedPub performs worse than our method due to the absence of information contained in local READOUT layers when describing tasks and inefficient exploitation of the global collaboration structure.

Dataset	Cora			CiteSeer			PubMed		
# Clients	5	10	20	5	10	20	5	10	20
Local	80.44±1.77	79.58±1.07	79.46±0.51	71.28±1.32	68.93±0.79	70.49±0.88	84.98±0.67	83.34±0.42	82.92±0.41
FedAvg	72.91±6.59	69.23±1.03	47.79±2.61	72.43±0.99	70.33±1.37	67.18±1.17	82.02±0.15	83.17±1.35	76.50±0.20
FedProx	63.96±3.07	71.39±4.16	70.88±5.90	73.63±0.85	42.86±2.59	42.31±3.18	83.99±0.17	83.57±0.09	83.93±0.89
FedPer	81.37±1.58	76.73±0.95	77.24±1.62	70.45±2.00	67.14±6.95	71.13±0.76	85.59±0.18	85.42±0.12	83.75±0.14
FedPub	82.33±1.46	78.27±1.46	79.15±1.08	74.11±1.58	72.12±1.83	68.16±1.41	86.22±0.21	85.58±0.31	84.79±0.46
FedSage	72.07±0.36	69.66±0.27	59.28±0.38	70.64±3.04	65.54±6.95	63.02±1.49	84.64±0.60	83.39±1.29	84.92±0.45
GCFL	79.91±1.93	73.25±4.39	76.37±1.82	71.37±2.54	67.58±0.61	63.54±3.34	84.24±0.57	83.47±0.29	83.72±0.47
FedStar	79.33±0.69	78.26±0.22	80.40±0.30	69.47±1.77	70.25±1.26	68.50±0.68	81.96±0.96	81.39±0.17	80.15±0.66
Ours	**83.37±1.59**	**80.06±1.27**	**81.17±0.63**	**75.25±1.38**	**74.18±1.14**	**71.17±1.69**	**87.05±0.19**	**86.53±0.73**	**86.38±0.30**
Dataset	Amazon Photo			Amazon Computers			Ogbn Arxiv		
# Clients	5	10	20	5	10	20	5	10	20
Local	77.97±0.29	86.14±1.05	86.37±0.17	65.90±0.29	74.41±1.51	81.81±0.50	56.93±0.89	56.54±0.37	57.79±0.89
FedAvg	53.49±5.87	45.82±1.88	35.15±1.03	46.03±1.93	39.04±3.68	43.74±8.15	55.84±0.88	61.02±0.32	59.30±0.18
FedProx	71.08±3.11	56.78±4.31	44.61±5.89	37.72±0.94	36.44±0.35	36.89±0.27	62.05±1.10	61.77±0.78	57.79±0.26
FedPer	68.19±1.68	77.15±0.14	78.96±0.68	64.30±0.34	64.47±0.20	70.44±0.57	61.57±0.50	61.52±0.37	62.73±0.26
FedPub	86.76±1.71	87.80±2.44	88.72±3.09	68.65±2.53	77.02±0.87	80.71±0.79	67.50±0.32	66.80±0.32	62.11±0.56
FedSage	51.28±7.30	51.68±7.28	51.39±7.22	42.88±5.23	50.41±7.84	57.06±0.42	58.63±1.29	61.65±0.45	54.86±1.77
GCFL	68.17±8.37	82.74±3.15	87.55±2.28	55.36±2.50	72.53±1.38	82.87±1.83	59.75±3.46	63.63±0.36	55.35±4.58
FedStar	85.67±0.31	86.85±0.09	87.60±0.68	71.88±1.70	78.81±1.41	83.42±0.73	58.96±0.82	60.77±0.46	61.36±0.14
Ours	**89.16±0.04**	**88.83±0.85**	**89.53±0.73**	**72.75±2.16**	**82.68±0.73**	**84.23±0.41**	**68.52±0.14**	**67.87±0.27**	**65.27±0.51**

Table 1: Node classification performance (%) on non-overlapping datasets

6.3.2 Convergence Analysis



(a) Cora (b) CiteSeer (c) PubMed (d) Photo (e) Computers (f) Arxiv

Figure 3: Convergence plot for the non-overlapping setting with 5 clients. We visualize the first 100 communication rounds.

Fig.3 illustrates the convergence behavior of the average test accuracy over the first 100 rounds with 5 clients in the system. It is evident that our proposed framework exhibits a rapid convergence rate towards the highest average test accuracy. This can be attributed to the framework's ability to efficiently capture the local task features and identify pairwise relationships.

6.3.3 Additional experiments

We conduct extensive sensitivity analysis and ablation studies for our framework in Appendix F, which shows the outperformance of each module in our framework.

7 Conclusion

Our paper proposes a novel framework that overcomes the limitations of existing federated GNN frameworks in local task featuring and task relating. We utilize graph distillation in task featuring, and introduce a novel kernelized attentive aggregation mechanism based on a collaborated network to incorporate global connectivity during model aggregation. The extensive experimental results demonstrate that our framework outperforms state-of-the-art methods.

References

[1] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[2] Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. Personalized subgraph federated learning. In *Proc. ICML*, 2022.

[3] Wenxuan Bao, Haohan Wang, Jun Wu, and Jingrui He. Optimizing the collaboration structure in cross-silo federated learning. In *Proc. ICML*, 2023.

[4] Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in gnns through the lens of effective resistance. In *Proc. ICML*, pages 2528–2547. PMLR, 2023.

[5] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, pages 3469–3489. PMLR, 2022.

[6] Fengwen Chen, Guodong Long, Zonghan Wu, Tianyi Zhou, and Jing Jiang. Personalized federated learning with graph. In *Proc. IJCAI*, 2022.

[7] Shuxiao Chen, Qinqing Zheng, Qi Long, and Weijie J Su. A theorem of the alternative for personalized federated learning. *arXiv preprint arXiv:2103.01901*, 2021.

[8] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[9] Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *International Conference on Machine Learning*, pages 5378–5396. PMLR, 2022.

[10] Yaqi Duan and Kaizheng Wang. Adaptive and robust multi-task learning. *arXiv preprint arXiv:2202.05250*, 2022.

[11] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[12] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[13] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[14] Steve Hanneke and Samory Kpotufe. A no-free-lunch theorem for multitask learning. *The Annals of Statistics*, 50(6):3119–3143, 2022.

[15] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[16] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Proc. NeurIPS*, pages 22118–22133, 2020.

[18] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[19] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proc. ACM SIGKDD*, pages 720–730, 2022.

[20] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *Proc. ICLR*, 2022.

[21] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

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

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. ICLR*, 2017.

[25] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proc. ICML*, volume 2002, pages 315–322, 2002.

[26] Erich L Lehmann and George Casella. *Theory of point estimation*. Springer Science & Business Media, 2006.

[27] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *Proc. ICML*, pages 6357–6368. PMLR, 2021.

[28] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proc. MLSys*, volume 2, pages 429–450, 2020.

[29] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. Multi-center federated learning: clients clustering for better personalization. *World Wide Web*, 26(1):481–500, 2023.

[30] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

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

[32] Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohamed, Mike Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. In *International Conference on Machine Learning*, pages 17716–17758. PMLR, 2022.

[33] Morteza Ramezani, Weilin Cong, Mehrdad Mahdavi, Mahmut Kandemir, and Anand Sivasubramaniam. Learn locally, correct globally: A distributed algorithm for training graph neural networks. In *International Conference on Learning Representations*, 2021.

[34] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

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

[36] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Proc. NeurIPS*, 30, 2017.

[37] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. Federated learning on non-iid graphs via structural knowledge sharing. In *Proc. AAAI*, volume 37, pages 9953–9961, 2023.

[38] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proc. EMNLP*, 2019.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[41] Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? In *International Conference on Machine Learning*, pages 10334–10343. PMLR, 2020.

[42] Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs. In *Proc. NeurIPS*, volume 34, pages 18839–18852, 2021.

[43] Jian Xu, Xinyi Tong, and Shao-Lun Huang. Personalized federated learning with feature alignment and classifier collaboration. In *Proc. ICLR*, 2023.

[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[45] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *Proc. ICML*, pages 40–48. PMLR, 2016.

[46] Rui Ye, Zhenyang Ni, Fangzhao Wu, Siheng Chen, and Yanfeng Wang. Personalized federated learning with inferred collaboration graphs. In *Proc. ICML*, 2023.

[47] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. In *Proc. NeurIPS*, volume 34, pages 6671–6682, 2021.

[48] Xuyang Zhao, Huiyuan Wang, and Wei Lin. The aggregation–heterogeneity trade-off in federated learning. In Gergely Neu and Lorenzo Rosasco, editors, *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pages 5478–5502. PMLR, 12–15 Jul 2023.

[49] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.

Appendix

A Algorithm

We present the algorithm of FedGKD framework in Alg.1.

Algorithm 1 FedGKD Framework

Input: Local datasets $\{G_i\}$;
Output: Personalized local models $\{W_i^T\}$;
Server
for $t = 1, \cdots, T$ **do**
  if $t > 1$ **then**
    Receive local GNN models $\{\mathbf{W}_i^t\}$
    Run task feature extractor;
    Get distilled node features $\{\mathbf{X}_i^{s,t}\}$ and their embeddings $\mathbf{H}_i^{s,t}$;
    Construct a collaboration network $G^{c,t}$;
    Kernelize $G^{c,t}$ to get a matrix $\mathbf{S}$ using (11);
    Aggregate local parameters into $\{\overline{\mathbf{W}_i^t}\}$
  end if
  Send GNN models $\{\overline{\mathbf{W}_i^t}\}$ to each client;
  Request each client to train local models;
end for

Client i
Receive $\overline{\mathbf{W}_\mathbf{i}^\mathbf{t}}$ from the server;
$W_i^t \leftarrow \overline{\mathbf{W}_\mathbf{i}^\mathbf{t}}$;
for $e = 1, \cdots, E_t$ **do**
  Update $\mathbf{W}_\mathbf{i}^\mathbf{t}$ with the loss function $\mathcal{L}(f(G_i; \mathbf{W}), \mathbf{y}_i) + \lambda ||\mathbf{W} - \overline{\mathbf{W}_\mathbf{i}^\mathbf{t}}||_2^2$;
end for

Algorithm 2 Task Feature Extractor

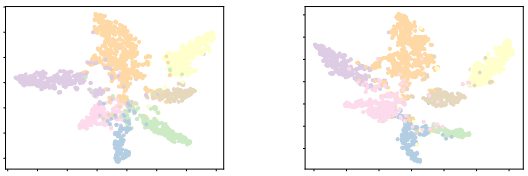
Input: Local model weights $\{\mathbf{W}_i^t\}$;
Output: Distilled node features $\{\mathbf{X}_i^{s,t}\}$ and their embeddings $\mathbf{H}_i^{s,t}$;
Server
Receive local GNN models $\{\mathbf{W}_i^t\}$ from each client;
Initialize node features $\mathbf{X}_0 \sim \mathcal{N}(0, 1)$;
Initialize labels $\mathbf{y}_0$;
Send $\mathbf{X}_0, \mathbf{y}_0$ to each client;
Receive distilled node features $\{\mathbf{X}_i^{s,t}\}$ and their embeddings $\mathbf{H}_i^{s,t}$ from each client;

Client i
Receive $\mathbf{X}_0$ and $\mathbf{y}_0$ from the server;
$\mathbf{X}_i^{s,t} \leftarrow \mathbf{X}_0$; $\mathbf{y}_i^{s,t} \leftarrow \mathbf{y}_0$;
Convert $\mathbf{y}_i^{s,t}$ into one-encoding form;
for $e = 1, \cdots, E_d$ **do**
  Use Gumbel-softmax to sample edges using (6);
  Compute loss $\mathcal{L}(f(G_i^{s,t}; \mathbf{W}_i^t), \mathbf{y}_i^{s,t})$;
  Update $\mathbf{X}_i^{s,t}$ and $\mathbf{y}_i^{s,t}$;
end for
Send $\mathbf{X}_i^{s,t}$ and their embeddings to the server;

B Complexity considerations

In comparison to vanilla FedAVG, the proposed framework requires additional local computations, a server-side matrix exponential operation, as well as extra communication costs. Let us briefly discuss the complexity of these procedures. Firstly, the dataset distillation procedure operates on small-scale graphs, making the total computation cost negligible compared to local training. Secondly, the matrix exponential operation upon $\mathbf{R}$ has a time complexity of $O(n^3)$. This complexity is controllable in practice since the number of clients, n, is typically small or moderate. Finally, the extra communication cost per client per aggregation step depends on the formulation of the task feature map. According to (7), the extra communication cost is $O(bmC(d+D))$, where b is the number of bits required to represent a floating-point number. This cost is comparable to the communication cost of a GNN. It is noteworthy that since GNN models are typically shallow, the communication cost of parameters is often dominated by the computation cost of local training. Moreover, the communication cost can be further reduced if a more compact task feature, such as the distilled graph embedding, is used. We will empirically investigate such alternatives in section F.5.

C Motivation study on limitation of embeddings



(a) Embeddings on Cora with the original labels

(b) Embeddings on Cora with relabelled nodes

Figure 4: Embedding spaces (depicted using two leading principle components) trained on two same graphs with divergent labels: vertices belonging to the same community have the same color.

As discussed above, embeddings fail to represent the READOUT layers following GNN layers in each model, discarding information. We visualized the embeddings for GCN layers trained on two datasets with similar graph distributions but divergent label distributions in Fig. 4. To be more specific, we trained node embeddings on the original Cora graph [45] and a revised Cora graph in which the label $y_v \in [C]$ of any node v is modified to $C+1-y_v$, where $C=7$ is the number of classes in Cora. The results show that the two embedding spaces are similar, as vertices belonging to the same community are located in similar positions in the space, as shown in Fig. 4.

D Experiment Setup

D.1 Datasets

We display the graph statistics in Table 2. The 6 graphs are of different scales and have different clustering coefficients.

Dataset	Cora	CiteSeer	PubMed	Amazon Photo	Amazon Computers	Obgn Arxiv
# Nodes	2,485	2,120	19,717	7,487	13,381	169,343
# Edges	10,138	7,358	88,648	238,086	491,556	2,315,598
# Features	1,433	3,703	500	745	767	128
# Categories	7	6	3	8	10	40
Clustering	0.238	0.170	0.062	0.411	0.351	0.226

Table 2: Original Graph Statistics

For non-overlapping and overlapping datasets, we display their average clustering coefficents and heterogeneity measured by medians of the client-wise label Jensen–Shannon divergences. Generally, a larger number of clients in the federated system leads to more heterogeneous local graphs.

To create federated datasets with overlapping vertices, we first use Metis graph partitioning algorithm to split the full graph into $\lfloor \frac{n}{5} \rfloor$ subgraphs. Then we randomly sample a half of the vertices from each subgraph for 5 times to create 5 graphs with overlapping vertices. After operating the same procedures

Dataset	Cora			CiteSeer			PubMed		
# Clients	5	10	20	5	10	20	5	10	20
# Nodes	497	249	124	424	212	106	3,943	1,972	986
# Edges	1,871	891	424	1,411	673	327	16,411	7,557	3,616
Clustering	0.250	0.261	0.266	0.176	0.177	0.181	0.063	0.066	0.068
Heterogeneity	0.304	0.358	0.391	0.142	0.229	0.255	0.087	0.097	0.119
Dataset	Amazon Photo			Amazon Computers			Obgn Arxiv		
# Clients	5	10	20	5	10	20	5	10	20
# Nodes	1,497	749	374	2,676	1,338	669	33,868	16,934	8,467
# Edges	42,930	19,294	8,300	84,202	35,589	24,577	406,896	182,758	86,150
Clustering	0.431	0.458	0.478	0.383	0.405	0.418	0.245	0.255	0.268
Heterogeneity	0.470	0.546	0.568	0.350	0.373	0.460	0.372	0.398	0.412

Table 3: Non-overlapping Dataset Statistics

for all the $\lfloor \frac{n}{5} \rfloor$ subgraphs, we consider the resulted n graphs belonging to different community but with overlapping vertices as the n local datasets. The dataset statistics of the overlapping datasets are shown in Table 4. Overlapping datasets suffer from great losses of edges compared with the global graph. In general, a small number of clients leads to a large number of missing edges.

Dataset	Cora			CiteSeer			PubMed		
# Clients	10	30	50	10	30	50	10	30	50
# Nodes	621	207	124	530	176	106	4,929	1,643	986
# Edges	752	246	141	590	184	104	6,494	2,090	1,158
Clustering	0.069	0.087	0.080	0.053	0.056	0.050	0.020	0.021	0.022
Heterogeneity	0.176	0.310	0.358	0.230	0.215	0.225	0.153	0.090	0.089
Dataset	Amazon Photo			Amazon Computers			Obgn Arxiv		
# Clients	10	30	50	10	30	50	10	30	50
# Nodes	1,872	623	374	3,345	1,115	669	42,336	14,112	8,467
# Edges	18,311	5,591	2,987	36,597	10,283	5,516	175,717	51,520	28,487
Clustering	0.291	0.297	0.307	0.256	0.258	0.268	0.122	0.127	0.131
Heterogeneity	0.310	0.416	0.535	0.316	0.422	0.344	0.371	0.363	0.507

Table 4: Overlapping Dataset Statistics

D.2 Baselines

We have compared the performance of our framework with 8 baselines. We will introduce the baselines briefly.

- Local: A local learning framework in which each client trains a model upon the local dataset without any collaboration.
- FedAvg: Clients send the local weights to the server and the server averages all the weights to initiate next round training.
- FedProx: Clients train the local models based on a local loss function added by a regularization term with importance hyper-parameter 0.001.
- FedPer: Clients only send the GCN layers to the server without uploading READOUT layer weights.
- FedPub: Server computes n graph embeddings from a randomized graph input to feature each task and aggregates the weights according to the graph embeddings.

- FedSage: Clients learn a GraphSage model and aggregate weights according to FedAvg framework.
- GCFL: Server cluster clients into groups so that clients belonging to the same groups aggregate weights together. We use the recommended parameters in the paper [42].
- FedStar: Clients decouple the node embeddings into shared structural embeddings and local embeddings. Clients only upload the GNN weights generating structural embeddings for server to aggregate. We use the recommended parameters in the paper [37].

D.3 Parameter Configuration

We perform a hyperparameter tuning using a grid-search method within the following range:

- Learning rate: $\{0.005, 0.01\}$;
- Number of local training epochs: $\{1, 3, 5, 7\}$;
- Sparsity control γ: $\{0.001, 0.75, 1.5, 2.5, 5.0\}$;
- Temperature on element-wise exponential τ_s: $\{1, 3, 5, 7, 9\}$;
- Temperature on matrix exponential τ: $\{0.05, 0.1, 0.25, 0.5, 0.75, 1.0\}$;
- Weights on proximal term λ: $\{10^{-5}, 10^{-3}\}$.

E Performance Comparison on Overlapping Datasets

Dataset	Cora			CiteSeer			PubMed		
# Clients	10	30	50	10	30	50	10	30	50
Local	46.88±1.23	66.45±0.81	70.32±0.68	51.42±1.75	59.06±1.64	61.40±1.45	76.75±0.20	77.46±0.20	76.02±0.33
FedAvg	49.75±1.32	46.20±2.67	43.48±3.97	54.79±2.86	54.14±1.41	57.52±1.98	78.53±0.68	80.99±0.26	79.53±0.06
FedProx	50.79±2.00	54.72±5.21	62.11±2.02	56.31±5.81	59.41±0.68	63.29±1.21	77.32±0.88	80.99±0.51	79.60±0.21
FedPer	52.83±0.55	67.15±0.85	70.27±0.34	57.14±1.45	62.21±1.80	63.26±1.95	79.85±0.31	80.59±0.06	80.28±0.13
FedPub	52.58±1.51	67.30±0.99	42.81±5.70	56.06±2.29	62.12±0.49	64.18±1.88	79.70±0.21	80.97±0.22	80.56±0.23
FedSage	49.25±0.50	59.42±1.03	59.99±0.23	55.54±6.95	55.63±7.00	62.73±1.09	77.87±0.50	80.97±0.24	79.36±0.73
GCFL	49.52±0.33	46.78±4.32	45.55±6.03	56.03±2.04	53.91±0.38	56.43±0.41	76.03±2.04	79.58±0.13	78.68±0.15
FedStar	43.09±0.72	61.60±0.30	67.77±1.25	46.45±0.17	54.78±2.12	58.96±1.81	75.45±0.14	76.45±0.43	74.71±0.52
Ours	**53.26±1.42**	**67.88±1.09**	**70.41±0.51**	**58.19±1.82**	**62.30±1.33**	**64.58±0.55**	**79.90±0.53**	**81.65±0.34**	**80.82±0.20**
Dataset	Amazon Photo			Amazon Computers			Ogbn Arxiv		
# Clients	10	30	50	10	30	50	10	30	50
Local	46.57±0.15	69.25±0.25	79.42±0.34	51.82±0.62	65.69±0.94	68.57±0.35	34.76±0.50	46.98±0.18	47.45±0.19
FedAvg	43.10±2.68	44.75±4.82	46.38±1.07	44.45±0.26	52.93±1.05	53.91±0.57	41.40±0.46	44.22±0.80	43.74±2.60
FedProx	43.58±2.05	45.29±0.53	42.76±5.23	42.59±4.17	53.58±1.56	53.91±0.57	41.35±0.20	44.68±0.62	47.02±0.52
FedPer	52.20±0.99	71.76±0.57	81.65±0.06	55.04±0.33	67.55±1.42	68.79±0.51	38.77±0.44	47.46±0.18	50.51±0.15
FedPub	45.69±2.10	64.50±0.48	76.58±0.85	50.15±1.57	60.81±0.52	63.82±0.62	42.18±0.36	50.58±0.21	51.11±0.56
FedSage	47.79±0.76	58.26±2.35	58.99±1.58	47.98±0.84	56.82±0.43	63.13±0.86	42.18±0.11	45.43±0.40	46.08±0.27
GCFL	46.93±0.55	48.95±0.48	40.76±8.08	56.02±1.04	58.38±2.16	52.97±1.74	41.32±0.23	46.26±1.12	46.16±0.34
FedStar	39.05±0.43	64.51±0.28	77.35±0.69	47.93±0.82	61.67±1.01	64.34±0.71	40.92±0.13	44.28±0.19	45.27±0.35
Ours	**56.07±1.44**	**72.15±0.49**	**81.91±0.29**	**56.56±0.15**	**68.43±0.28**	**69.11±1.07**	**43.02±0.04**	**48.32±0.14**	**52.38±0.12**

Table 5: Node classification performance (%) on overlapping datasets

F Sensitivity Analysis and Ablation Study

F.1 Effects of the proposed task relator

The primary goal of this study is to investigate the benefits of incorporating the global information. Specifically, we tested with FedGKD along with two local variants obtained via replacing the matrix S in (11) on the Cora dataset. The *local* variant corresponds to the standard softmax kernel that sets $\mathbf{S} = \mathbf{R}$. The *square* variant corresponds to setting $\mathbf{S} = \mathbf{R}^2$, which can be understood as performing a two-layer message passing. As shown in Fig.5, we observe that incorporating global information leads to improved federated learning performance, especially when the number of clients is large. This implies that inter-client information is more complicated, and the proposed task relater provides a more nuanced solution. Furthermore, we also compare these variations with FedPub framework and observe that even using local connectivity generated from distilled datasets

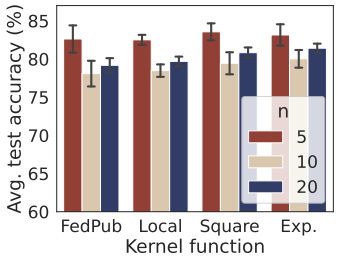


Figure 5: Effects of kernel functions

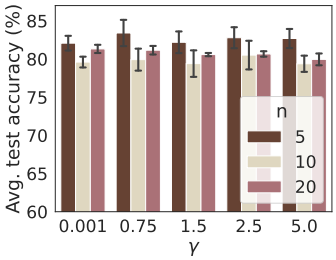


Figure 6: Effects of sparsity control coefficient γ

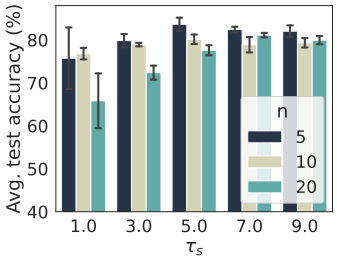


Figure 7: Effects of temperature on element-wise exponential τ_s

instead of graph embeddings on a random graph input to relate tasks, our framework outperforms FedPub. This suggests that distilled graphs are more representative than graph embeddings due to their incorporation of READOUT layers.

F.2 Effects of Sparsity-controlling Coefficient γ

We conduct an ablation study on Cora to assess the impact of the sparsity control coefficient γ in distilled graphs. A larger γ generates a more sparse distilled graph. We vary γ across $\{10^{-3}, 0.75, 1.5, 2.5, 5\}$ to test the effect. Our findings show that the optimum value for γ in our framework is 0.75. We hypothesize that distilled graphs' density is influenced by the constraint of containing few nodes, as a sparse small graph results in almost no connections. This observation is consistent with results from centralized graph distillation[20, 19].

F.3 Effects of Temperature on Element-wise Exponential τ_s

We investigate the impact of varying the temperature values on the element-wise exponential τ_s defined in (11) on Cora dataset. τ_s is a metric that regulates the influence of the local model weights W_i^t on the aggregated weights $\overline{W_i^t}$. In a federated GNN system with significantly heterogeneous local datasets, a large value of τ_s is required to achieve optimal performance. This idea is supported by the results presented in Fig.7. In Appendix D.1, we demonstrate that the number of clients results in more heterogeneous graphs within the system, necessitating a larger value of τ_s to attain optimal performance in FedGKD.

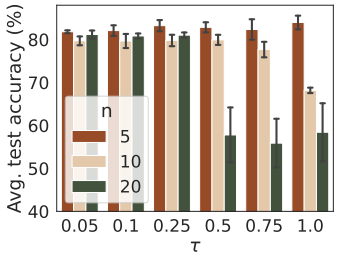


Figure 8: Effects of temperature on local connectivity matrix exponential τ

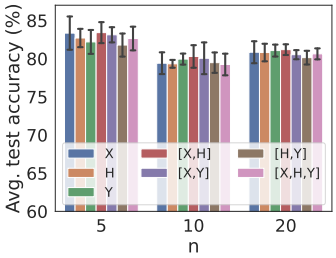


Figure 9: Effects of of task features from distilled graphs

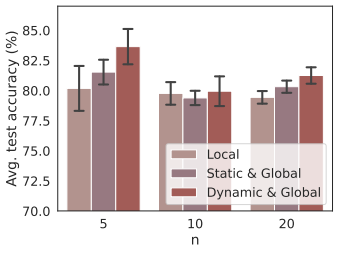


Figure 10: Effects of static and dynamic task feature extractor

F.4 Effects of Temperature on Matrix Exponential τ

We experiment with varying the values of τ in the exponential of the relation matrix, as defined in (11) on Cora. τ is introduced to avoid the singularity of the matrix exponential. As shown in Fig.8, a large τ may results in extremely low rank aggregation weight matrix thereby deteriorating model performance. Therefore, it is essential to set an appropriate value of τ to guarantee non-singularity.

F.5 Effects of Task Features from Distilled Graphs

We experiment with multiple choices of statistics obtained from distilled graphs to compute the pairwise task-relatedness in (10) on Cora dataset. Fig.9 illustrates that choices of statistics are robust to model performance but the concatenation of node feature $\mathbf{X}$ and embeddings $\mathbf{H}$ outperforms others slightly. It is worth noting that if communication overhead is a major concern, we can further reduce the extra communication cost by transmitting only $\mathbf{H}$ or even the distilled labels, which incurs only a slight performance degradation.

F.6 Ablation Study on Dynamic Task Feature Extractor

We conducted a series of experiments in order to compare the effects of static and dynamic task feature extractors within our framework. These experiments were carried out on Cora dataset using non-overlapping settings. To accomplish this, we replaced our dynamic feature extractor, which distilled graphs based on the current model weights during each communication round, with a static extractor. This static extractor utilized a graph distillation method introduced in [19] to ensure that the performance of the model training on the synthetic small graph closely approximated the "final" performance achieved when training on the original, larger graph before the federated learning procedures. In this way, the static extractor offers static representations of tasks, which remain unchanged throughout the entire federated learning process. As shown in Fig. 10, the dynamic task feature extractor is visually replaced by the static extractor for our experimental purposes. Notably, this replacement results in a 2.4% degradation in performance. The inferior performance of static task feature extractor comes from the unsatisfying local sample quality and quantity. Additionally, we compared the performance of the static extractor with a local training baseline and found that the static extractor still outperforms local learning.

G On the Validity of k in (11)

Below we state the result that k in (11) is a valid kernel:

Proposition G.1. *The function k is a valid kernel over the domain $[n]$.*

Proof. Let $\{\lambda_1, \ldots, \lambda_n\}$ be the eigenvalues of $\mathbf{R}$. It then follows that the eigenvalues of $\mathbf{S}$ are $\{e^{t\lambda_1}, \ldots, e^{t\lambda_n}\}$ which are non-negative, therefore $\mathbf{S}$ is positive semidefinite. The proposition follows by [16, Theorem 7.5.9] □