FederatedScope-GNN: Towards a Unified, Comprehensive and Efficient Package for Federated Graph Learning

Zhen Wang
Alibaba Group
jones.wz@alibaba-inc.com

Weirui Kuang
Alibaba Group
weirui.kwr@alibaba-inc.com

Yuexiang Xie
Alibaba Group
yuexiang.xyx@alibaba-inc.com

Liuyi Yao
Alibaba Group
yly287738@alibaba-inc.com

Yaliang Li*
Alibaba Group
yaliang.li@alibaba-inc.com

Bolin Ding
Alibaba Group
bolin.ding@alibaba-inc.com

Jingren Zhou
Alibaba Group
jingren.zhou@alibaba-inc.com

ABSTRACT

The incredible development of federated learning (FL) has benefited various tasks in the domains of computer vision and natural language processing, and the existing frameworks such as TFF and FATE has made the deployment easy in real-world applications. However, federated graph learning (FGL), even though graph data are prevalent, has not been well supported due to its unique characteristics and requirements. The lack of FGL-related framework increases the efforts for accomplishing reproducible research and deploying in real-world applications. Motivated by such strong demand, in this paper, we first discuss the challenges in creating an easy-to-use FGL package and accordingly present our implemented package **F**ederated**S**cope-**G**NN (FS-G), which provides (1) a unified view for modularizing and expressing FGL algorithms; (2) comprehensive DataZoo and ModelZoo for out-of-the-box FGL capability; (3) an efficient model auto-tuning component; and (4) off-the-shelf privacy attack and defense abilities. We validate the effectiveness of FS-G by conducting extensive experiments, which simultaneously gains many valuable insights about FGL for the community. Moreover, we employ FS-G to serve the FGL application in real-world E-commerce scenarios, where the attained improvements indicate great potential business benefits. We publicly release FS-G, as submodules of FederatedScope, at https://github.com/alibaba/FederatedScope to promote FGL's research and enable broad applications that would otherwise be infeasible due to the lack of a dedicated package.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; **Machine learning algorithms**.

KEYWORDS

Federated Learning; Graph Neural Networks

1 INTRODUCTION

Along with the rising concerns about privacy, federated learning (FL) [28], a paradigm for collaboratively learning models without access to dispersed data, has attracted more and more attention from both industry and academia. Its successful applications include keyboard prediction [12], object detection [26], speech recognition [29], the list goes on. This fantastic progress benefits from the FL frameworks, e.g., TFF [5] and FATE [40], which save practitioners from the implementation details and facilitate the transfer from research prototype to deployed service.

However, such helpful supports have mainly focused on tasks in vision and language domains. Yet, the graph data, ubiquitous in real-world applications, e.g., recommender systems [36], healthcare [42], and anti-money laundering [33], have not been well supported. As a piece of evidence, most existing FL frameworks, including TFF, FATE, and PySyft [43], have not provided off-the-shelf federated graph learning (FGL) capacities, not to mention the lack of FGL benchmarks on a par with LEAF [6] for vision and language tasks.

As a result, FL optimization algorithms, including FedAvg [28], FedProx [23], and FedOPT [1], are mainly evaluated on vision and language tasks. When applied to optimize graph neural network (GNN) models, their characteristics are unclear to the community. Another consequence of the lack of dedicated framework support is that many recent FGL works (e.g., FedSage+ [42] and GCFL [37]) have to implement their methods from scratch and conduct experiments on respective testbeds.

We notice that such a lack of widely-adopted benchmarks and unified implementations of related works have become obstacles to

---

*Corresponding author.

developing novel FGL methods and the deployment in real-world applications. It increases engineering effort and, more seriously, introduces the risk of making unfair comparisons. Therefore, it is much in demand to create an FGL package that can save the effort of practitioners and provide a testbed for accomplishing reproducible research. To this end, we pinpoint what features prior frameworks lack for FGL and the challenges to satisfy these requirements:

(1) Unified View for Modularized and Flexible Programming. In each round of an FL course, a general FL algorithm (e.g., FedAvg) exchanges homogeneous data (here model parameters) for one pass. In contrast, FGL algorithms [36, 37, 42] often require several heterogeneous data (e.g., gradients, node embeddings, encrypted adjacency lists, etc.) exchanges across participants. Meanwhile, besides ordinary local updates/global aggregation, participants of FGL have rich kinds of subroutines to handle those heterogeneous data. FL algorithms are often expressed in most existing frameworks by declaring a static computational graph, which pushes developers to care about coordinating the participants for these data exchanges. Thus, an FGL package should provide a unified view for developers to express such heterogeneous data exchanges and the various subroutines effortlessly, allowing flexible modularization of the rich behaviors so that FGL algorithms can be implemented conveniently.

(2) Unified and Comprehensive Benchmarks. Due to the privacy issue, real-world FGL datasets are rare. Most prior FGL works are evaluated by splitting a standalone graph dataset. Without a unified splitting mechanism, they essentially use their respective datasets. Meanwhile, their GNN implementations have not been aligned and integrated into the same FL framework. All these increase the risk of inconsistent comparisons of related works, urging an FGL package to set up configurable, unified, and comprehensive benchmarks.

(3) Efficient and Automated Model Tuning. Most federated optimization algorithms have not been extensively studied with GNN models. Hence, practitioners often lack proper prior for tuning their GNN models under the FL setting, making it inevitable to conduct hyper-parameter optimization (HPO). Moreover, directly integrating a general HPO toolkit into an FL framework cannot satisfy the efficiency requirements due to the massive cost of executing an entire FL course [18]. Even a single model is tuned perfectly, the prevalent non-i.i.d.ness in federated graph data might still lead to unsatisfactory performances. In this situation, monitoring the FL procedure to get aware of the non-i.i.d.ness and personalizing (hyper-)parameters are helpful for further tuning the GNN models.

(4) Privacy Attacks and Defence. Performing privacy attacks on the FL algorithm is a direct and effective way to examine whether the FL procedure has the risk of privacy leakage. However, none of the existing FL frameworks contains this. Moreover, compared with the general FL framework, except for sharing the gradients of the global model, FGL may also share additional graph-related information among clients, such as node embeddings [36] and neighbor generator [42]. Without verifying the security of sharing such information, the application of FGL remains questionable.

Motivated by these, in this paper, we develop an FGL package FS-G to satisfy these challenging requirements:

(1) We choose to build FS-G upon an event-driven FL framework FederatedScope [38], which abstracts the exchanged data into messages and characterizes the behavior of each participant by defining the message handlers. Users who need to develop FGL

algorithms can simply define the (heterogeneous) messages and handlers, eliminating the engineering for coordinating participants. Meanwhile, different handlers can be implemented with respective graph learning backends (e.g., torch_geometric and tf_geometric).

(2) For the ease of benchmarking related FGL methods, FS-G provides a *GraphDataZoo* that integrates a rich collection of splitting mechanisms applicable to most existing graph datasets and a *GNNModelZoo* that integrates many state-of-the-art FGL algorithms. Thus, users can reproduce the results of related works effortlessly. It is worth mentioning that we identify a unique covariate shift of graph data that comes from the graph structures, and we design a federal random graph model for the corresponding further study.

(3) FS-G also provides a component for tuning the FGL methods. On the one hand, it provides fundamental functionalities to achieve low-fidelity HPO, empowering users of FS-G to generalize existing HPO algorithms to the FL settings. On the other hand, when a single model is inadequate to handle the non-i.i.d. graph data, our model-tuning component provides rich metrics to monitor the dissimilarity among clients and a parameter grouping mechanism for describing various personalization algorithms in a unified way.

(4) Considering the additional heterogeneous data exchanged in FGL, demonstrating the level of privacy leakage under various attacks and providing effective defense strategies are indispensable. FS-G includes a dedicated component to provide various off-the-shelf privacy attack and defence abilities, which are encapsulated as plug-in functions for the FGL procedure.

We utilize FS-G to conduct extensive experimental studies to validate the implementation correctness, verify its efficacy, and better understanding the characteristics of FGL. Furthermore, we employ FS-G to serve three real-world E-commerce scenarios, and the collaboratively learned GNN outperforms their locally learned counterparts, which confirms the business value of FS-G. We have open-sourced FS-G for the community, which we believe can ease the innovation of FGL algorithms, promote their applications, and benefit more real-world business.

2 RELATED WORK

2.1 Federated Learning

Generally, the goal of FL is to solve: $\min_\theta f(\theta) = \sum_{i=1}^N p_i F_i(\theta) = \mathbb{E}[F_i(\theta)]$, where N is the number of clients (a.k.a. devices or parties), $F_i(\cdot)$ is the local objective of the i-th client, $p_i > 0$, and $\sum_i^N p_i = 1$. As a special case of distributed learning, the essential research topic for FL is its optimization approaches. Concerning the communication cost, FedAvg [28] allows clients to make more than one local update at each round. The following works include FedProx [23], FedOPT [1], FedNOVA [35], SCAFFOLD [16], etc. These methods have been extensively studied on vision and language tasks, but when applied to optimize GNNs, their characteristics are less understood to the community. We refer readers to the survey papers [15, 22].

2.2 Federated Graph Learning

When handling graph-related tasks under the FL setting, several unique algorithmic challenges emerge, e.g., complete the cross-client edges [42], handle the heterogeneous graph-level tasks [37], augment each client's subgraph [36], and align the entities across clients [30]. Many recent FGL works have attempted to resolve such challenges, which usually require exchanging heterogeneous
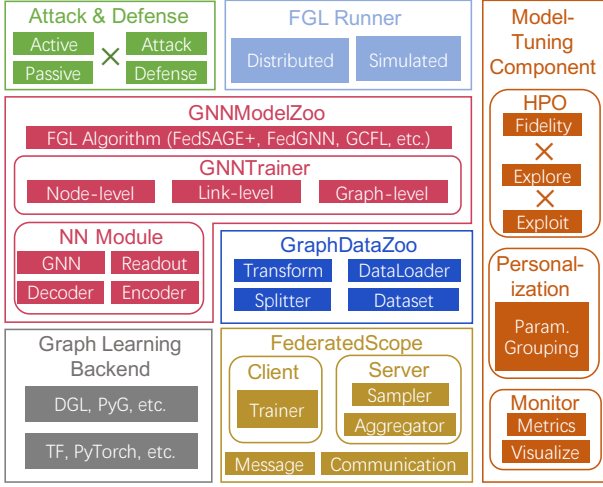
Figure 1: Overview of FS-G.

data across the participants, and the behaviors of the participants become richer than ordinary FL methods. These characteristics of FGL algorithms lead to the unique requirements (see Sec. 3).

2.3 FL Software

With the need for FL increasing, many FL frameworks [2, 5, 14, 27, 31, 32, 40, 43] have sprung up. Most of them are designed as a conventional distributed machine learning framework, where a computational graph is declared and split for participants. Then each specific part is executed by the corresponding participant. Users often have to implement their FL algorithms with declarative programming (i.e., describing the computational graph), which raises the bar for developers. This usability issue is exacerbated in satisfying the unique requirements of FGL methods. Consequently, most existing FL frameworks have no dedicated support for FGL, and practitioners cannot effortlessly build FGL methods upon them. An exception is FedML [14], one of the first FL frameworks built on an event-driven architecture, provides a FGL package FedGraphNN [13]. However, they still focus on the FGL algorithms that have simple and canonical behaviors. Many impactful FGL works have not been integrated, including those discussed above. Besides, they have ignored the requirements of efficiently tuning GNN models and conducting privacy attacks&defence for FGL algorithms, which are crucial in both practice and research.

3 INFRASTRUCTURE

We present the overview of FS-G in Fig. 1. At the core of FS-G is an event-driven FL framework FEDERATEDSCOPE [38] with fundamental utilities (i.e., framing the FL procedure) and it is compatible with various graph learning backends. Thus, we build our *GNNModelZoo* and *GraphDataZoo* upon FEDERATEDSCOPE with maximum flexibility for expressing the learning procedure and minimum care for the federal staff (e.g., coordinating participants). With such ease of development, we have integrated many state-of-the-art FGL algorithms into *GNNModelZoo*. We design the *Runner* class as a convenient interface to access FGL executions, which unifies the simulation and the distributed modes. Meanwhile, an auto model-tuning component for performance optimization and a component for privacy attack and defense purposes are provided in FS-G.

Table 1: A summary of three representative FGL algorithms, where we only list the behaviors other than ordinary local updates and aggregation.

Method	FedSage+ [42]	FedGNN [36]	GCFL+ [37]
Task	Node classification	Link prediction	Graph classification
Exchange	Model param. Node emb. NeighGen param. NeighGen grad.	Model param. Node emb. Adj. list	Model param. Model grad.
Server behavior	Broadcast emb. Broadcast grad.	Node clustering Broadcast emb.	Grad. clustering Param. deriving
Client behavior	Send emb. and NeighGen Apply cross-client grad.	Generate pseudo edges	None

3.1 Requirements of Federated Graph Learning

We first review the existing FGL algorithms and summarize their uniqueness against general FL. As shown in Table 1, the three very recent FGL works, targeting different tasks, need to exchange heterogeneous data across the participants. In contrast, in each round of a general FL procedure (e.g., using FedAvg), only homogeneous data are exchanged for one pass from server to clients and one pass back. As a result of this difference, the participant of an FGL course often executes rich kinds of subroutines to handle the received data and prepare what to send. In contrast, the participant of a general FL course has canonical behaviors, i.e., local updates or aggregation.

Thus, there is a demand for a unified view to express these multiple passes of heterogeneous data exchanges and the accompanying subroutines. In this way, developers can be agnostic about the communication, better modularize the FGL procedure, and choose the graph learning backend flexibly.

3.2 Development based on FederatedScope

To satisfy the unique requirements of FGL discussed above, we develop FS-G based on a event-driven FL framework named FEDERATEDSCOPE, which abstracts the data exchange in an FL procedure as message passing. With the help of FEDERATEDSCOPE, implementing FGL methods can be summarized in two folds: (1) defining what kinds of messages should be exchanged; (2) describing the behaviors of the server/client to handle these messages. From such a point of view, a standard FL procedure is shown in Fig. 2, where server and client pass homogeneous messages (i.e., the model parameters). When receiving the messages, they conduct aggregation and local updates, respectively. As for FGL algorithms, we take FedSage+ as an example. As shown in Fig. 3, we extract the heterogeneous exchanged messages according to FedSage+, including model parameters, gradients, and node embeddings. Then we frame and transform the operations defined in training steps to callback functions as subroutines to handle different types of received messages. For example, when receiving the request, the client employs the corresponding callback function to send node embeddings and "NeighGen" (i.e., a neighbor generation model) back to the server.

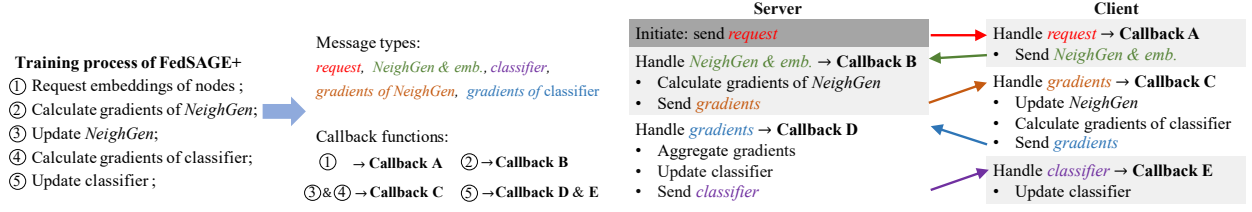**Figure 2: Implement a standard FL algorithm based on FEDERATEDSCOPE.**

Figure 3: Implement FedSage+ based on the event-driven framework FEDERATEDSCOPE.

The goal of FS-G includes both *convenient usage* for the existing FGL methods and *flexible extension* for new FGL approaches. Benefited from FEDERATEDSCOPE, the heterogeneous exchanged data and various subroutines can be conveniently expressed as messages and handlers, which supports us to implement many state-of-the-art FGL methods, including FedSage+, FedGNN, and GCFL+, by providing different kinds of message (e.g., model parameters, node embeddings, auxiliary model, adjacent list, etc) and participants' behavior (e.g., broadcast, cluster, etc). The modularization of a whole FGL procedure into messages and handlers makes it flexible for developers to express various operations defined in customized FGL methods separately without considering coordinating the participants in a static computational graph.

4 GRAPHDATAZOO

A comprehensive GraphDataZoo is indispensable to provide a unified testbed for FGL. To satisfy the various experiment purposes, we allow users to constitute an FL dataset by configuring the choices of *Dataset*, *Splitter*, *Transform*, and *Dataloader*. Conventionally, the *Transform* classes are responsible for mapping each graph into another, e.g., augmenting node degree as a node attribute. The *Dataloader* classes are designed for traversing a collection of graphs or sampling subgraphs from a graph. We will elaborate on the *Splitter* and the Dataset classes in this section.

Tasks defined on graph data are usually categorized as follow: *(1) Node-level task:* Each instance is a node which is associated with its label. To make prediction for a node, its k-hop neighborhood is often considered as the input to a GNN. *(2) Link-level task:* The goal is to predict whether any given node pair is connected or the label of each given link (e.g., the rating a user assigns to an item). *(3) Graph-level task:* Each instance is an individual graph which is associated with its label. For the link/node-level tasks, transductive setting is prevalent, where both the labeled and unlabeled links/nodes appear in the same graph. As for the graph-level task, a standalone dataset often consists of a collection of graphs.

4.1 Splitting Standalone Datasets

Existing graph datasets are a valuable source to satisfy the need for more FL datasets [15]. Under the federated learning setting, the dataset is decentralized. To simulate federated graph datasets by existing standalone ones, our *GraphDataZoo* integrates a rich collection of *splitters*. These *splitters* are responsible for dispersing a given standalone graph dataset into multiple clients, with configurable statistical heterogeneity among them. For the node/link-level tasks,

each client should hold a subgraph, while for the graph-level tasks, each client should hold a subset of all the graphs.

We aim to enable related works to compare on unified, configurable, and comprehensive federated graph datasets, and thus provide many off-the-shelf *splitters*. Some splitters split a given dataset by specific meta data or the node attribute value, expecting to simulate realistic FL scenarios. Some other splitters are designed to provide various non-i.i.d.ness, including covariate shift, concept drift, and prior probability shift [15]. Details about the provided splitters and the FL datasets constructed by applying them can be found in the Appendix A.1 and Appendix A.3, respectively.

4.2 New Federated Learning Datasets

In addition to the the strategy of splitting existing standalone datasets, we also construct three federated graph datasets from other real-world data sources or federal random graph model:

(1) FedDBLP: We create this dataset from the latest DBLP dump, where each node corresponds to a published paper, and each edge corresponds to a citation. We use the bag-of-words of each paper's abstract as its node attributes and regard the theme of paper as its label. To simulate the scenario that a venue or an organizer forbids others to cite its papers, FS-G allows users to split this dataset by each node's venue or the organizer of that venue.

(2) Cross-scenario recommendation (CSR): We create this dataset from the user-item interactions collected from an E-commerce platform. FS-G allows users to split the graph by each item's category or by which scenario an interaction happens.

(3) FedcSBM: Graph data consist of attributive and structural patterns, but prior federated graph datasets have not decoupled the covariate shifts of these two aspects. Hence, we propose a federal random graph model *FedcSBM* based on cSBM [10]. *FedcSBM* can produce the dataset where the node attributes of different clients obey the same distribution. Meantime, the homophilic degrees can be different among the clients, so that covariate shift comes from the structural aspect.

5 GNNMODELZOO AND MODEL-TUNING COMPONENT

As an FGL package, FS-G provides a *GNNModelZoo*. As discussed in Sec. 4, models designated for tasks of different levels will encounter heterogeneous input and/or output, thus requiring different architectures. Hence, in the *NN module* of FS-G, we modularize a general neural network model into four categories of building bricks: *(1) Encoder*: embeds the raw node attributes or edge attributes, e.g., atom encoder and bond encoder. *(2) GNN*: learns discriminative

representations for the nodes from their original representations (raw or encoded) and the graph structures. *(3) Decoder*: recovers these hidden representations back into original node attributes or adjacency relationships. *(4) Readout*: aggregates node representations into a graph representation, e.g., the mean pooling. With a rich collection of choices in each category, users can build various kinds of neural network models out-of-the-box. Particularly, in addition to the vanilla GNNs [9, 11, 19, 34, 39], our *GNNModelZoo* also includes the GNNs that decouples feature transformation and propagation, e.g., GPR-GNN [8]. Such a kind of GNNs has been ignored by FedGraphNN [13]. However, we identify their unique advantages in FGL—handling covariate shift among the client-wise graphs that comes from graph structures.

For the convenience of developers, FS-G integrates the *GNNTrainer* class, which encapsulates the local training procedure. It can be easily configured to adjust for different levels of tasks and full-batch/graph-sampling settings. Thus, developer can focus on the FGL algorithms without caring for the procedure of local updates. Then we implement many state-of-the-art FGL algorithms and integrate them into our *GNNModelZoo*.

With *GraphDataZoo* and *GNNModelZoo*, users are empowered with FGL capacities. However, the performances of FGL algorithms are often sensitive to their hyper-parameters. It is indispensable to make hyper-parameter optimization (HPO) to create reproducible research. To this ends, FS-G incorporates a *model-tuning component*, which provides the functionalities for (1) making efficient HPO under the FL setting; (2) monitoring the FL procedure and making personalization to better handle non-i.i.d. data.

5.1 Federated Hyper-parameter Optimization

In general, HPO is a trial-and-error procedure, where, in each trial, a specific hyper-parameter configuration is proposed and evaluated. HPO methods mainly differ from each other in exploiting the feedback of each trial and exploring the search space based on previous trials. Whatever method, one of the most affecting issues is the cost of making an exact evaluation, which corresponds to an entire training course and then evaluation. This issue becomes severer under the FGL setting since an entire training course often consists of hundreds of communication rounds, and FGL methods, in each round, often exchange additional information across participants, where even a single training course is extremely costly.

A general and prosperous strategy to reduce the evaluation cost is making multi-fidelity HPO [25]. FS-G allows users to reduce the fidelity by (1) making a limited number of FL rounds instead of an entire FL course for each trial; (2) sampling a small fraction of clients, e.g., $K(K \ll N)$, in each round. We use Successive Halving Algorithm (SHA) [20] as an example to show how multi-fidelity HPO methods work in the FL setting. As Fig. 4 shows, a set of candidate configurations are maintained, each of which will be evaluated in each stage of SHA. When performing low-fidelity HPO, each specific configuration can be evaluated by restoring an FL course from its corresponding checkpoint (if it exists), making only a few FL rounds to update the model, and evaluating the model to acquire its performance. Then these configurations are sorted w.r.t. their performances and only the top half of them are reserved for the next stage of SHA. This procedure continues until only one configuration remains, regarded as the optimal one.
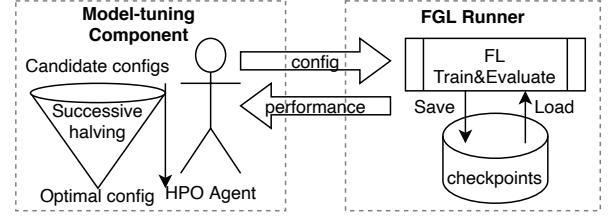
Figure 4: An example of HPO for FGL: FS-G allows each training course to be restored from a given checkpoint, proceed for any specified number of rounds, and be saved for continual training.

We identify the requirement of functionalities to save and restore an FGL training course from this algorithmic example. These functionalities are also indispensable to achieving a reliable failover mechanism. Thus, we first show which factors determine the state of an FL course: *(1) Server-side*: Basically, the current round number and the global model parameters must be saved. When the aggregator is stateful, e.g., considering momentum, its maintained state, e.g., the moving average of a certain quantity, needs to be kept. *(2) Client-side*: When the local updates are made with mini-batch training, the client-specific data-loader is usually stateful, whose index and order might need to be held. When personalization is utilized, the client-specific model parameters need to be saved. With sufficient factors saved as a checkpoint, FS-G can restore an FGL training course from it and proceed.

Meanwhile, we follow the design of FEDERATEDSCOPE and make the entry interface of FS-G as a callable FGL runner, which receives a configuration and returns a collection of metrics for the conducted FGL training course (entire or not). Consequently, each HPO algorithm incorporated in our model-tuning component can be abstracted as an agent, repeatedly calling the FGL runner to collect the feedback. Benefiting from this interface design and the capacity to save and restore an FGL training course, any one-shot HPO method can be effortlessly generalized to the FGL setting upon FS-G. It is worth mentioning that what to return by the FGL runner is configurable, where efficiency-related metrics, e.g., the average latency of each round, can be included. Therefore, optimizing hyper-parameters from the system perspective [41] is also supported by FS-G.

5.2 Monitoring and Personalization

Practitioners often monitor the learning procedure by visualizing the curves of training loss and validation performance to understand whether the learning has converged and the GNN model has overfitted. When it comes to FGL, we consider both the client-wise metrics, e.g., the local training loss, and some metrics to be calculated at the server-side. Specifically, we have implemented several metrics, including B-local dissimilarity [23] and the covariance matrix of gradients, calculated from the aggregated messages to reflect the statistical heterogeneity among clients. The larger these metrics are, the more different client-wise graphs are. As shown in Fig. 5a, the B-local dissimilarity on non-i.i.d. data is larger than that on i.i.d. data at almost all stages of the training course, which becomes particularly noticeable at the end.
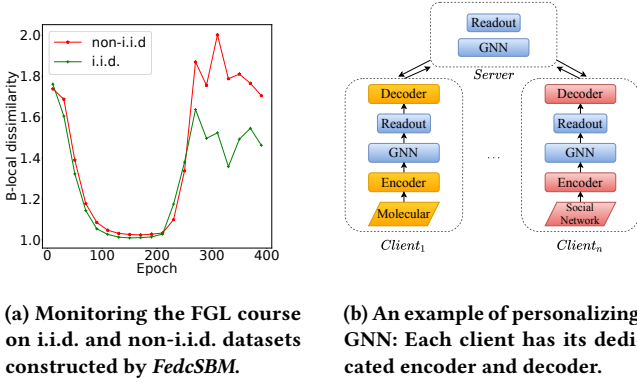
(a) Monitoring the FGL course on i.i.d. and non-i.i.d. datasets constructed by *FedcSBM*.

(b) An example of personalizing GNN: Each client has its dedicated encoder and decoder.

Figure 5: Examples of monitoring and personalization.

Then we build the monitoring functionality upon related toolkits (e.g., WandB and TensorBoard) to log and visualize the metrics. To use the out-off-shelf metrics, users only need to specify them in the configuration. Meantime, FS-G has provided the API for users to register any quantity calculated/estimated during the local update/aggregation, which would be monitored in the execution.

Once some monitored metrics indicate the existence of non-i.i.d.ness, users can further tune their GNN by personalizing the model parameters and even the hyper-parameters. We present an example of personalization in Fig. 5b. Each client has its specific encoder and decoder as the tasks among the clients come from different domains with different node attributes and node classes. In practice, a more fine-grained personalization might be preferred, where only some layers or even some variables are client-specific.

To satisfy such purposes, FS-G first allows users to instantiate the model of each participant individually. Then we build a flexible parameter grouping mechanism upon the naming systems of underlying machine learning engines. Specifically, this mechanism allows users to easily declare each part (with flexible granularity) of the model as client-specific or shared. Only the shared parts will be exchanged.

6 OFF-THE-SHELF ATTACK AND DEFENCE ABILITIES

The privacy attack&defence component of FEDERATEDSCOPE has integrated various off-the-shelf passive privacy attack methods, including class representative inference attack, membership inference attack, property inference attack, and training inputs and labels inference attack. These methods have been encapsulated as optional hooks for our *GNNTrainer*. Once the user has selected a specific hook, the GNN model and some needed information about the target data would automatically be fed into the hook during the training procedure. Besides the previous passive attack setting where the adversaries are honest-but-curious, FS-G also supports the malicious adversary setting. The attackers can deviate from the FL protocol by modifying the messages. To defend the passive privacy attacks, FS-G can leverage FEDERATEDSCOPE's plug-in defense strategies, including differential privacy, MPC, and data compression. Meanwhile, FEDERATEDSCOPE provides the information checking mechanism to effectively detect anomaly messages and defend the malicious attacks.

7 EXPERIMENTS

We utilize FS-G to conduct extensive experiments, with the aim to validate the implementation correctness of FS-G, set up benchmarks for FGL that have long been demanded, and gain more insights about FGL. Furthermore, we deploy FS-G in real-world E-commerce scenarios to evaluate its business value.

7.1 An Extensive Study about Federated Graph Learning

In this study, we consider three different settings: (1) LOCAL: Each client trains a GNN model with its data. (2) FGL: FedAvg [28], FedOpt [1], and FedProx [23] are applied to collaboratively train a GNN model on the dispersed data, respectively. (3) GLOBAL: One GNN model is trained on the completed dataset. By comparing these settings with various GNN architectures and on diverse tasks, we intend to set up comprehensive and solid benchmarks for FGL.

7.1.1 Node-level Tasks.

Protocol. For the purposes of training and validation, the client-wise data are subgraphs deduced from the original graph. Yet, global evaluation is considered for testing, where models are evaluated on the test nodes of original graph. Thus, in both local and FGL settings, we train and validate each client's model on its incomplete subgraph and test the model on the complete global graph. In the global setting, we train and evaluate each model on the complete global graph. For each setting, we consider popular GNN architectures: GCN, GraphSage, GAT, and GPR-GNN. To conduct the experiments uniformly and fairly, we split the nodes into train/valid/test sets, where the ratio is 60% : 20% : 20% for citation networks and 50% : 20% : 30% for FedDBLP. We randomly generate five splits for each dataset. Each model is trained and evaluated with these five splits, and we report the averaged metric and the standard deviation. To compare the performance of each model, we choose accuracy as the metric for all node-level tasks. In addition, we perform hyper-parameter optimization (HPO) for all methods with the learning rate $\in \{0.01, 0.05, 0.25\}$ in all settings, and the local update steps $\in \{1, 4, 16\}$ in FGL.

Results and Analysis. We present the results on three citation networks in Table 2 and Table 3, with *random_splitter* and *community_splitter*, respectively. Overall, FGL can improve performance regarding those individually trained on local data, as the experience in vision and language domains suggests. In most cases on the randomly split datasets, the GNN trained on the global (original) data performs better than that of FGL. With the splitting, the union of all client-wise graphs still has fewer edges than the original graph, which may explain this performance gap. In contrast, the FGL setting often performs better than those trained on the global graph when our *community_splitter* constructs the federated graph datasets. At first glance, this phenomenon is counterintuitive, as the splitter also removes some of the original edges. However, these citation graphs exhibit homophily, saying that nodes are more likely to connect to nodes with the same label than those from other classes. When split by the community detection-based algorithm, we identify the changes that removed edges often connect nodes with different labels, which improves the homophilic degree of the graphs. As a result, the resulting federated graph dataset becomes

Table 2: Results on representative node classification datasets with _random_splitter_: Mean accuracy ± standard deviation.

	Cora					CiteSeer					PubMed				
	Local	FedAvg	FedOpt	FedProx	Global	Local	FedAvg	FedOpt	FedProx	Global	Local	FedAvg	FedOpt	FedProx	Global
GCN	80.95±1.49	86.63±1.35	86.11±1.29	86.60±1.59	86.89±1.82	74.29±1.35	76.48±1.52	77.43±0.90	77.29±1.20	77.42±1.15	85.25±0.73	85.29±0.95	84.39±1.53	85.21±1.17	85.38±0.33
GraphSAGE	75.12±1.54	85.42±1.80	84.73±1.58	84.83±1.66	86.86±2.15	73.30±1.30	76.86±1.38	75.99±1.96	78.05±0.81	77.48±1.27	84.58±0.41	86.45±0.43	85.67±0.45	86.51±0.37	86.23±0.58
GAT	78.86±2.25	85.35±2.29	84.40±2.70	84.50±2.74	85.78±2.43	73.85±1.00	76.37±1.11	76.96±1.75	77.15±1.54	76.91±1.02	83.81±0.69	84.66±0.74	83.78±1.11	83.79±0.87	84.89±0.34
GPR-GNN	84.90±1.13	89.00±0.66	87.62±1.20	88.44±0.75	88.54±1.58	74.81±1.43	79.67±1.41	77.99±1.25	79.35±1.11	79.67±1.42	86.85±0.39	85.88±1.24	84.57±0.68	86.92±1.25	85.15±0.76

Table 3: Results on representative node classification datasets with _community_splitter_: Mean accuracy ± standard deviation.

	Cora					CiteSeer					PubMed				
	Local	FedAvg	FedOpt	FedProx	Global	Local	FedAvg	FedOpt	FedProx	Global	Local	FedAvg	FedOpt	FedProx	Global
GCN	65.08±2.39	87.32±1.49	87.29±1.65	87±16±1.51	86.89±1.82	67.53±1.87	77.56±1.45	77.80±0.99	77.62±1.42	77.42±1.15	77.01±3.37	85.24±0.69	84.11±0.87	85.14, 0.88	85.38±0.33
GraphSAGE	61.29±3.05	87.19±1.28	87.13±1.47	87.09±1.46	86.86±2.15	66.17±1.50	77.80±1.03	78.54±1.05	77.70±1.09	77.48±1.27	78.35±2.15	86.87±0.53	85.72±0.58	86.65±0.60	86.23±0.58
GAT	61.53±2.81	86.08±2.52	85.65±2.36	85.68±2.68	85.78±2.43	66.17±1.31	77.21±0.97	77.34±1.33	77.26±1.02	76.91±1.02	75.97±3.32	84.38±0.82	83.34±0.87	84.34±0.63	84.89±0.34
GPR-GNN	69.32±2.07	88.93±1.64	88.37±2.12	88.80±1.29	88.54±1.58	71.30±1.65	80.27±1.28	78.32±1.45	79.73±1.52	79.67±1.42	78.52±3.61	85.06±0.82	84.30±1.57	86.77±1.16	85.15±0.76

Table 4: Results on representative link prediction datasets with _label_space_splitter_: Hits@n.

	WN18															FB15k-237															
	Local			FedAvg			FedOpt			FedProx			Global			Local			FedAvg			FedOpt			FedProx			Global		
	1	5	10	1	5	10	1	5	10	1	5	10	1	5	10	1	5	10	1	5	10	1	5	10	1	5	10	1	5	10
GCN	20.70	55.34	73.85	30.00	79.72	96.67	22.13	78.96	94.07	27.32	83.01	96.38	29.67	86.73	97.05	6.07	20.29	30.35	9.86	34.27	48.02	4.12	18.07	31.79	4.66	28.74	41.67	7.80	32.46	44.64
GraphSAGE	21.06	54.12	79.88	23.14	78.85	93.70	22.82	79.86	93.12	23.14	78.52	93.67	24.24	79.86	93.84	3.95	14.64	24.47	7.13	23.38	36.60	2.20	19.21	27.64	5.85	24.05	36.33	6.19	23.57	35.98
GAT	20.89	49.42	72.48	23.14	77.62	93.49	23.14	74.64	93.52	23.53	78.40	93.00	24.24	80.18	93.76	3.44	15.02	25.14	6.06	25.76	39.04	2.71	18.89	32.76	6.19	25.09	38.00	6.94	24.43	37.87
GPR-GNN	22.86	60.45	80.73	26.67	82.35	96.18	24.46	73.33	87.18	27.62	81.87	95.68	29.19	82.34	96.24	4.45	13.26	21.24	9.62	32.76	45.97	2.01	9.81	16.65	3.72	15.62	27.79	10.62	33.87	47.45

Table 5: Results on representative graph classification datasets: Mean accuracy (%) ± standard deviation.

	PROTEINS					IMDB					Multi-task				
	Local	FedAvg	FedOpt	FedProx	Global	Local	FedAvg	FedOpt	FedProx	Global	Local	FedAvg	FedOpt	FedProx	Global
GCN	71.10±4.65	73.54±4.48	71.24±4.17	73.36±4.49	71.77±3.62	50.76±1.14	53.24±6.04	50.49±8.32	48.72±6.73	53.24±6.04	66.37±1.78	65.99±1.18	69.10±1.58	68.59±1.99	-
GIN	69.06±3.47	73.74±5.71	60.14±1.22	73.18±5.66	72.47±5.53	55.82±7.56	64.79±10.55	51.87±6.82	70.65±8.35	72.61±2.44	75.05±1.81	63.40±2.22	63.33±1.18	63.01±0.44	-
GAT	70.75±3.33	71.95±4.45	71.07±3.45	72.13±4.68	72.48±4.32	53.12±5.81	53.24±6.04	47.94±6.53	53.82±5.69	53.24±6.04	67.72±3.48	66.75±2.97	69.58±1.21	69.65±1.14	-

easier to handle by most GNN architecture. Studies on FedSage+ and the comparisons on _FedDBLP_ are deferred to Appendix A.2.

7.1.2 Link-level Tasks.

Protocol. We largely follow the experimental setup in Sec. 7.1.1. Specifically, we split the links into train/valid/test sets, where the ratio is 80% : 10% : 10% for recommendation dataset Ciao. We use the official train/valid/test split for the knowledge graphs WN18 and FB15k-237. In addition, the link predictor for each model is a two-layer MLP with 64 as the hidden layer dimension, where the input is the element-wise multiplication of the node embeddings of the node pair. We report the mean accuracy for the recommendation dataset and Hits at 1, 5, and 10 for the knowledge graphs.

Results and Analysis. We present the results on the knowledge graphs in Table 4 and the results on _Ciao_ in Table 9. Overall, the performances of the FGL setting are better than those of the local setting, which is consistent with the conclusion drawn from node-level tasks and further confirms the effectiveness of FGL. Notably, the splitting strategy results in unbalanced relation distributions among the clients on knowledge graphs, making the triplets of each client insufficient to characterize the entities. Thus, the performance gaps between the local and the FGL settings are broad. Meantime, the performances of the FGL setting become comparable to those of the global setting.

7.1.3 Graph-level Tasks.

Protocol. We largely follow the experimental setup in Sec. 7.1.2. Specifically, a pre-linear layer is added to encode the categorical attributes preceding GNN. And a two-layer MLP is added to adapt

GNN to multi-task classification as a classifier following GNN. Notably, for multi-task FGL in multi-domain datasets, only the parameters of GNN are shared.

Results and Analysis. We present the main results in Table 5, where there is no global setting on _Multi-task_ dataset, as the same model cannot handle the graphs of different tasks. Overall, the relationships between different settings have been preserved compared to those on node-level tasks. One exception is on the _Multi-task_ dataset, where the non-i.i.d.ness is severe, e.g., the average number of nodes of the graphs on a client deviates from other clients a lot. Although we have personalized the models by FS-G, the gradients/updates collected at the server-side might still have many interferences. Without dedicated techniques to handle these, e.g., GCFL+, FedAvg cannot surpass the individually trained models. But with federated optimization algorithms, such as FedOpt and FedProx, some GNN architectures surpass the individually trained models, which proves the effectiveness of these federated optimization algorithms. In addition, we implement personalized FL algorithms (here FedBN [24] and Ditto [21]) and apply them for GIN on the multi-task dataset. FedBN and Ditto lead to performances (i.e., mean accuracy with standard deviation) 72.90±1.33 (%) and 63.35±0.6 (%), respectively. Due to the label unbalance, all kinds of GNNs result in the same accuracy on the _HIV_ dataset. Hence, we have to solely report their ROC-AUC in Table 10 of Appendix A.2, from which we can draw similar conclusions. Meantime, GIN surpasses GCN with FedAvg on almost all datasets, which implies the capability of distinguishing isomorphism graphs is still critical for federated graph-level tasks. Our studies about GCFL+ are deferred to Appendix A.2.

7.2 Study about Hyper-parameter Optimization

In this experiment, we conduct HPO for federated learned GCN and GPR-GNN, intending to verify the effectiveness of our model-tuning component. Moreover, we can set up an HPO benchmark for the FGL setting and attain some insights, which have long been missed but strongly demanded.

Protocol. We take SHA (see Sec. 5.1) to optimize the hyper-parameters for GCN/GPR-GNN learned by FedAvg. we study low-fidelity HPO, where the number of FL rounds for each evaluation $\in \{1, 2, 4, 8\}$, and the client sampling rate for each round of FedAvg $\in \{20\%, 40\%, 80\%, 100\%\}$. For each choice of fidelity, we repeat the SHA five times with different seeds, and we report the average ranking of its searched hyper-parameter configuration.

(a) GCN.   **(b) GPR-GNN.**
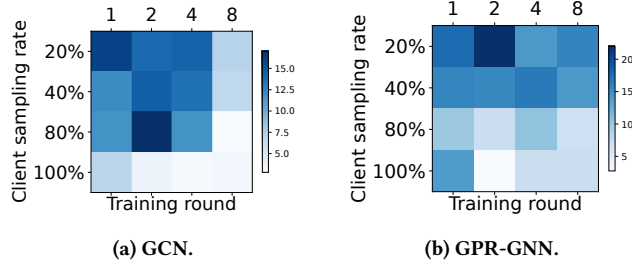
Figure 6: SHA with various fidelity to optimize GCN/GPR-GNN: We report the average ranking of searched hyper-parameter configuration (the small, the better).

Results and Analysis. There are in total 72 possible configurations, with each of which we conduct the FGL procedure and thus acquire the ground-truth performances. These results become a lookup table, making comparing HPO methods efficient. We present the experimental results in Fig. 6, where higher fidelity leads to better configuration for both kinds of GNNs. At first, we want to remind our readers that the left-upper region in each grid table corresponds to extremely low-fidelity HPO. Although their performances are worse than those in the other regions, they have successfully eliminated a considerable fraction of poor configurations. Meanwhile, increasing fidelity through the two aspects, i.e., client sampling rate and the number of training rounds, reveal comparable efficiency in improving the quality of searched configurations. This property provides valuable flexibility for practitioners to keep a fixed fidelity while trading-off between these two aspects according to their system status (e.g., network latency and how the dragger behaves). All these observations suggest the application of low-fidelity HPO to FGL, as well as the effectiveness of FS-G's model-tuning component.

7.3 Study about Non-I.I.D.ness and Personalization

We aim to study the unique covariate shift of graph data—node attributes of different clients obey the identical distributions, but their graph structures are non-identical. Meantime, we evaluate whether the personalization provided by FS-G's model-tuning component can handle such non-i.i.d.ness.

Protocol. We choose *FedcSBM* (see Sec. 4.2) as the dataset. Specifically, we randomly generate a sample from our *FedcSBM*, containing

eight graphs with different homophilic degrees. Then we choose GPR-GNN as the GNN model and consider three settings: *(1) Ordinary FL*: We apply FedAvg to optimize the model where the clients collaboratively optimize all model parameters; (2) *Local*: We allow each client to optimize its model on its graph; (3) *Personalization*: We apply FedAvg to optimize the parameters for feature transformation while the spectral coefficients (i.e., that for propagation) are client-specific. We repeat such a procedure five times and report the mean test accuracy and the standard deviation. More details can be found in Appendix A.2.
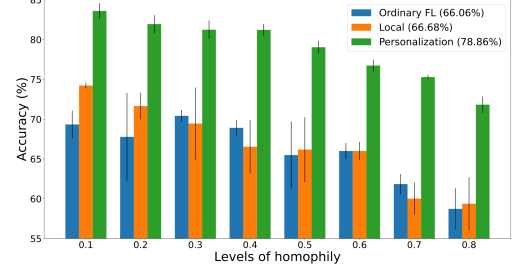
Figure 7: Accuracy by levels of homophily and methods.

Results and Analysis. We illustrate the results in Fig. 7, where each level of homophilic degree corresponds to a client. Overall, the personalization setting outperforms others. We attribute this advantage to making appropriate personalization, as the personalization setting consistently performs better across different clients, i.e., on graphs with different levels of homophilic degrees. On the other hand, the ordinary FL exhibits comparable performances with the local setting, implying the collaboration among clients fails to introduce any advantage. FedAvg might fail to converge due to the dissimilarity among received parameters, as the B-local dissimilarities shown in Fig. 5a indicate.

7.4 Deployment in Real-world E-commerce Scenarios

We deploy FS-G to serve one federation with three E-commerce scenarios operated by different departments. These scenarios include search engine and recommender systems before and after the purchase behavior. As they all aim to predict whether each given user will click a specific item, and their users and items have massive intersections, sharing the user-item interaction logs to train a model is promising and has long been their solution. However, such data sharing raises the risk of privacy leakage and breaks the newly established regulations, e.g., Cybersecurity Law of China.

Without data sharing, each party has to train its model on its user-item bipartite graph, missing the opportunity to borrow strength from other graphs. Considering the scales of their graphs are pretty different, e.g., one scenario has only 184,419 interactions while another has 2,860,074, this might severely hurt the performance on "small" scenarios. With our FGL service, these parties can collaboratively train a GraphSAGE model for predicting the link between each user-item pair. Evaluation on their held-out logs shows that the ROC-AUC of individually trained GraphSAGE models is 0.6209±0.0024 while that of FGL is 0.6278±0.0040, which is a significant improvement. It is worth noting that a small improvement (at 0.001-level) in offline ROC-AUC evaluation means a substantial difference in real-world click-through rate prediction for

search/recommendation/advertisements. Therefore, the improvement confirms the business value of FS-G.

8 CONCLUSION

In this paper, we implemented an FGL package, FS-G, to facilitate both the research and application of FGL. Utilizing FS-G, FGL algorithms can be expressed in a unified manner, validated against comprehensive and unified benchmarks, and further tuned efficiently. Meanwhile, FS-G provides rich plug-in attack and defence utilities to assess the level of privacy leakage for the FGL algorithm of interest. Besides extensive studies on benchmarks, we deploy FS-G in real-world E-commerce scenarios and gain business benefits. We will release FS-G to create greater business value from the ubiquitous graph data while preserving privacy.

REFERENCES

[1] Muhammad Asad, Ahmed Moustafa, and Takayuki Ito. 2020. FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning. *Applied Sciences* 10 (2020).
[2] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. 2020. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390* (2020).
[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.
[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* (2008), P10008.
[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems*. 374–388.
[6] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).
[7] Chuan Chen, Weibo Hu, Ziyue Xu, and Zibin Zheng. 2021. FedGL: Federated Graph Learning Framework with Global Self-Supervision. *arXiv preprint arXiv:2105.03170* (2021).
[8] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2020. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.
[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *NeurIPS* (2016), 3844–3852.
[10] Yash Deshpande, Subhabrata Sen, Andrea Montanari, and Elchanan Mossel. 2018. Contextual Stochastic Block Models. In *NeurIPS*.
[11] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.
[12] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).
[13] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S. Yu, Yu Rong, Peilin Zhao, Junzhou Huang, Murali Annavaram, and Salman Avestimehr. 2021. FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks. *arXiv preprint arXiv:2104.07145* (2021).
[14] Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. 2020. FedML: A Research Library and Benchmark for Federated Machine Learning. *arXiv preprint arXiv:2007.13518* (2020).
[15] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).
[16] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *ICML*. 5132–5143.
[17] George Karypis and Vipin Kumar. 2000. Multilevel k-way hypergraph partitioning. *VLSI design* (2000), 285–300.

[18] Mikhail Khodak, Renbo Tu, Tian Li, Liam Li, Nina Balcan, Virginia Smith, and Ameet Talwalkar. 2021. Federated Hyperparameter Tuning: Challenges, Baselines, and Connections to Weight-Sharing. In *NeurIPS*.
[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[20] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* (2017), 6765–6816.
[21] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and Robust Federated Learning Through Personalization. In *ICML*. 6357–6368.
[22] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.
[23] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* (2020), 429–450.
[24] Xiaoxiao Li, Meirui JIANG, Xiaofei Zhang, Michael Kamp, and Qi Dou. 2021. FedBN: Federated Learning on Non-IID Features via Local Batch Normalization. In *ICLR*.
[25] Yaliang Li, Zhen Wang, Yuexiang Xie, Bolin Ding, Kai Zeng, and Ce Zhang. 2021. Automl: From methodology to application. In *CIKM*. 4853–4856.
[26] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. 2020. Fedvision: An online visual object detection platform powered by federated learning. In *AAAI*. 13172–13179.
[27] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, et al. 2020. Ibm federated learning: an enterprise framework white paper v0. 1. *arXiv preprint arXiv:2007.10987* (2020).
[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*. 1273–1282.
[29] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, et al. 2021. Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications. *arXiv preprint arXiv:2102.08503* (2021).
[30] Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. 2021. Differentially Private Federated Knowledge Graphs Embedding. In *CIKM*. 1416–1425.
[31] Nuria Rodríguez-Barroso, Goran Stipcich, Daniel Jiménez-López, José Antonio Ruiz-Millán, Eugenio Martínez-Cámara, Gerardo González-Seco, M Victoria Luzón, Miguel Angel Veganzones, and Francisco Herrera. 2020. Federated Learning and Differential Privacy: Software tools analysis, the Sherpa. ai FL framework and methodological guidelines for preserving data privacy. *Information Fusion* (2020), 270–292.
[32] Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robert Sandmann, Robin Roehm, and Michael A Hoeh. 2021. Pyvertical: A vertical federated learning framework for multi-headed splitnn. *arXiv preprint arXiv:2104.00489* (2021).
[33] Toyotaro Suzumura, Yi Zhou, Natahalie Baracaldo, Guangnan Ye, Keith Houck, Ryo Kawahara, Ali Anwar, Lucia Larise Stavarache, Yuji Watanabe, Pablo Loyola, et al. 2019. Towards federated graph learning for collaborative financial crimes detection. *arXiv preprint arXiv:1909.12946* (2019).
[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[35] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. 2020. In *NeurIPS*. 7611–7623.
[36] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).
[37] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. *NeurIPS* 34 (2021).
[38] Yuexiang Xie, Zhen Wang, Daoyuan Chen, Dawei Gao, Liuyi Yao, Weirui Kuang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2022. FederatedScope: A Flexible Federated Learning Platform for Heterogeneity. *https://arxiv.org/abs/2204.05011* (2022).
[39] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *ICLR*.
[40] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. 2019. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3 (2019), 1–207.
[41] Huanle Zhang, Mi Zhang, Xin Liu, Prasant Mohapatra, and Michael DeLucia. 2021. Automatic Tuning of Federated Learning Hyper-Parameters from System Perspective. *arXiv preprint arXiv:2110.03061* (2021).
[42] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. *NeurIPS* 34 (2021).
[43] Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemke, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, et al. 2021. PySyft: A Library for Easy Federated Learning. In *Federated Learning Systems*. 111–139.

A APPENDIX

A.1 Details of Off-the-shelf Splitters

To this end, we have implemented mainly six classes of *splitters*:

(1) community_splitter: This is often adopted in node-level tasks to simulate the locality-based federated graph data [42], where nodes in the same client are densely connected while cross-client edges are unavailable. Specifically, community detection algorithms (e.g., Louvain [4] and METIS [17]) are at first applied to partition a graph into several clusters. Then these clusters are assigned to the clients, optionally with the objective of balancing the number of nodes in each client.

(2) random_splitter: Random split is often adopted in node-level tasks, e.g., FedGL [7]. Specifically, the node set of the original graph is randomly split into N subsets with or without intersections. Then, the subgraph of each client is deduced from the nodes assigned to that client. Optionally, a specified fraction of edges is randomly selected to be removed.

(3) meta_splitter: In many cases, there are meta data or at least interpretable edge/node attributes that allow users to simulate a real FL setting via splitting the graph based on the meta data or the values of those attributes. In citation networks, papers published in different conferences or organizations usually focus on different themes. Splitting by conference/organization naturally leads to node (i.e., paper) classification tasks with non-identical label distributions (i.e., prior probability shift [15]). Meanwhile, in recommender systems, the same user often has different tendencies to items in different domains/scenarios, where splitting by domain/scenario can provide concept shift among clients.

(4) instance_space_splitter: It is responsible for creating feature distribution skew (i.e., covariate shift). To realize this, we sort the graphs based on their values of a certain aspect, e.g., for Molhiv, molecules are sorted by their scaffold, and then each client is assigned with a segment of the sorted list.

(5) label_space_splitter: It is designed to provide label distribution skew. For classification tasks, e.g., relation prediction for knowledge graph completion, the existing triplets are split into the clients by latent dirichlet allocation (LDA) [3]. For regression tasks, e.g., PCQM4M, FS-G can discretize the label before conducting LDA.

(6) multi_task_splitter: This is mainly designed for multi-task learning or personalized learning. Sometimes different clients have different tasks, e.g., in the domain of the molecule, some clients have the task of determining the toxicity, while some clients have the task of predicting the HOMO-LUMO gap. A more challenging case [37] is that the graphs come from the different domains, e.g., molecules, proteins, and social networks.

A.2 Details about Our Experiments

Experimental settings. In node-level tasks, the detailed hyperparameters in our experiments are as follows: the number of training rounds is 400, the early stopping is 100, the GNN layers is 2 (in GPR-GNN, K is 10), the hidden layer dimension is 64 on citation networks and 1024 on FedDBLP, the dropout is 0.5, weight decay is 0.0005, the number of clients is five on citation networks and the optimizer is SGD.

More results and analysis about node-level tasks. Particularly, we consider one of the recently proposed FGL algorithms,

FedSAGE+ [42], which is highlighted by simultaneously training generative models for predicting the missing links so that its Graph-SAGE models can be trained on the mended graphs. We show the results in Table 7, where FedSage+ significantly outperforms its baseline (i.e., GraphSage) on most of the datasets. It benefits from the jointly learned generative models, which enable each client to reconstruct the missing cross-client links under federated setting.

We present the results on *FedDBLP* in Table 8, where the fraction of removed edges reaches 60% and 40% when split by venue and organizer, respectively. Besides, there are 20 and 8 clients under the two splitting settings, respectively, larger than the previous experiments. Since the client-specific graphs are tiny, w.r.t. the original one, the available training examples are limited for the local setting. All these factors make the performances of different GNNs unsatisfactory under the local setting. As for FGL, since FedAvg aggregates the clients' updates, it somehow exploits all the training examples and thus achieves comparable performances against the global setting. Considering that *FedDBLP* has simulated the data interruption in real life, these results confirm the effectiveness of FGL to handle this emerging challenge.

More results and analysis about link-level tasks. We provide more experimental results on the FedDBLP, Ciao, and HIV in the Table. 8, Table. 9, and Table. 10. All experimental settings are consistent with Sec. 7.1.1, Sec. 7.1.2, and Sec. 7.1.3, respectively.

More results and analysis about graph-level tasks. In addition, we consider the recently proposed FGL algorithm GCFL+ [37], which clusters clients and performs FedAvg in a cluster-wise manner so that clients with similar data distributions share a common GIN model. In Table 11, GCFL+ outperforms its baseline (i.e., GIN federally learned by FedAvg) on both the IMDB and Multi-task datasets, and achieves comparable performance on PROTEINS. GCFL+ clusters the clients according to their sequences of gradients, where clients belonging to the same cluster share the same model parameters. Its advantages are likely to come from this smart mechanism, which handles the non-i.i.d.ness among clients better.

More details about the protocol of our studies on hyperparameter optimization. We largely follow the experimental setup in the Sec. 7.1.1 while extend the search space of the hyperparameters, where hidden dim $\in \{32, 64\}$, dropout $\in \{0.0, 0.5\}$, and weight decay in $\in \{0, 0.0005\}$. As PubMed has much more nodes and edges than the other two citation networks, we target the node classification task on PubMed to draw statistically reliable conclusions. In order to simulate the FL setting, we apply our *community_splitter* to divide the PubMed into five parts for five clients.

Non-I.I.D.ness and Personalization Study. In order to generate different level of homophily graphs, we set $\phi \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ for cSBM model. We repeat our experiment with GPR-GNN for three-time with a different seed. In FGL setting, each client shares the parameters of linear layers, while the parameters of the label propagation layer are personalized.

Real-world Deployment. We provide more statistical information about the real-world E-commerce scenarios dataset. Search engine scenario contains 106,222 users and 464,904 items. In the other two scenarios, the first contains 12,588 users and 78,996 items, and the second contains 107,589 users and 559,796, respectively.

Table 6: Datasets statistics.

Task	Domain	Dataset	Splitter	# Graph	Avg. # Nodes	Avg. # Edges	# Class	Evaluation
Node-level	Citation network	Cora	*random&community*	1	2,708	5,429	7	ACC
	Citation network	CiteSeer	*random&community*	1	4,230	5,358	6	ACC
	Citation network	PubMed	*random&community*	1	19,717	44,338	5	ACC
	Citation network	FedDBLP	*meta*	1	52,202	271,054	4	ACC
Link-level	Recommendation System	Ciao	*meta*	28	5,875.68	20,189.29	6	ACC
	Recommendation System	Taobao	*meta*	3	443,365	2,015,558	2	ACC
	Knowledge Graph	WN18	*label_space*	1	40,943	151,442	18	Hits@n
	Knowledge Graph	FB15k-237	*label_space*	1	14,541	310,116	237	Hits@n
Graph-level	Molecule	HIV	*instance_space*	41,127	25.51	54.93	2	ROC-AUC
	Proteins	Proteins	*instance_space*	1,113	39.05	145.63	2	ACC
	Social network	IMDB	*label_space*	1,000	19.77	193.06	2	ACC
	Multi-task	Mol	*multi_task*	18,661	55.62	1,466.83	-	ACC

Table 11: Comparisons between GCFL+ and GIN (with FedAvg) on graph classification datasets: Mean accuracy (%) ± standard deviation.

	PROTEINS	IMDB	Multi-task
GIN	73.74±5.71	64.79±10.55	63.40±2.22
GCFL+	73.00±5.72	69.47±8.71	65.14±1.23

Table 7: Comparisons between FedSage+ and GraphSAGE (with FedAvg) on representative node classification datasets: Mean accuracy (%) ± standard deviation.

	Cora		CiteSeer		PubMed	
	random	community	random	community	random	community
GraphSAGE	85.42±1.80	87.19±1.28	76.86±1.38	77.80±1.03	86.45±0.43	86.87±0.53
FedSage+	85.07±1.20	87.68±1.55	78.04±0.91	77.98±1.23	88.19±0.32	87.94±0.27

Table 8: Results on representative node classification datasets with *meta_splitter*: Mean accuracy (%) ± standard deviation.

	FedDBLP (by venue)			FedDBLP (by publisher)		
	Local	FGL	Global	Local	FGL	Global
GCN	58.86±0.32	77.53±0.03	78.50±0.04	67.59±0.44	77.98±0.04	78.50±0.04
GraphSAGE	51.05±0.43	78.57±0.03	78.96±0.39	61.60±0.19	79.23±0.05	78.96±0.39
GAT	59.06±0.48	77.31±0.07	OOM	68.38±0.61	77.52±0.06	OOM
GPR-GNN	58.07±0.39	76.53±0.23	OOM	66.10±0.25	78.17±0.12	OOM

Table 9: Results on link classification dataset *Ciao* with *meta_splitter*: Mean accuracy (%) ± standard deviation.

	Ciao		
	Local	FGL	Global
GCN	46.76±0.44	49.18±0.00	49.18±0.00
GraphSAGE	46.62±0.35	49.18±0.00	49.18±0.00
GAT	46.83±0.31	49.18±0.00	49.18±0.00
GPR-GNN	47.73±0.75	49.24±0.08	49.21±0.07

Table 10: Results on graph classification dataset *HIV* with *instance_space_splitter*: Mean ROC-AUC scrore ± standard deviation.

	HIV (ROC-AUC)		
	Local	FGL	Global
GCN	0.6193±0.0319	0.6263±0.0332	0.6939±0.0165
GIN	0.6925±0.0354	0.7774±0.0195	0.7958±0.0200
GAT	0.6192±0.0101	0.6287±0.0197	0.7034±0.0201

A.3 Datasets description

As in Table 6, we provide a detailed description of datasets of FS-G with datasets and suggested *Splitter* accordingly. The datasets are collected from different domains, and the nodes and edges represent different meanings. We will support more datasets and provide more benchmarks in the future.