FedGraphNN: A Federated Learning Benchmark System for Graph Neural Networks

Chaoyang He & Keshav Balasubramanian & Emir Ceyani *
Viterbi School of Engineering
University of Southern California

Carl Yang & Han Xie
Department of Computer Science
Emory University

Lichao Sun & Lifang He
Department of Computer Science and Engineering
Lehigh University

Liangwei Yang & Philip S. Yu
Department of Computer Science
University of Illinois at Chicago

Yu Rong & Peilin Zhao & Junzhou Huang
Machine Learning Center
Tencent AI Lab

Murali Annavaram & Salman Avestimehr
Viterbi School of Engineering
University of Southern California

Abstract

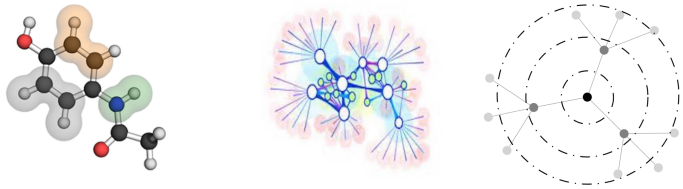
Graph Neural Network (GNN) research is rapidly growing thanks to the capacity of GNNs in learning distributed representations from graph-structured data. However, centralizing a massive amount of real-world graph data for GNN training is prohibitive due to privacy concerns, regulation restrictions, and commercial competitions. Federated learning (FL), a trending distributed learning paradigm, provides possibilities to solve this challenge while preserving data privacy. Despite recent advances in vision and language domains, there is no suitable platform for the FL of GNNs. To this end, we introduce `FedGraphNN`, an open FL benchmark system that can facilitate research on federated GNNs. `FedGraphNN` is built on a unified formulation of graph FL and contains a wide range of datasets from different domains, popular GNN models, and FL algorithms, with secure and efficient system support. Particularly for the datasets, we collect, preprocess, and partition 36 datasets from 7 domains, including both publicly available ones and specifically obtained ones such as `hERG` and `Tencent`. Our empirical analysis showcases the utility of our benchmark system, while exposing significant challenges in graph FL: federated GNNs perform worse in most datasets with a non-IID split than centralized GNNs; the GNN model that attains the best result in the centralized setting may not maintain its advantage in the FL setting. These results imply that more research efforts are needed to unravel the mystery behind federated GNNs. Moreover, our system performance analysis demonstrates that the `FedGraphNN` system is computationally efficient and secure to large-scale graphs datasets. We maintain the source code at `https://github.com/FedML-AI/FedGraphNN`.

*The first three authors contribute equally. Email: {chaoyang.he,keshavba,ceyani}@usc.edu.

This is a full version. Shorter versions are accepted to **ICLR 2021** Workshop on Distributed and Private Machine Learning (DPML) and **MLSys 2021** - GNNSys'21 - Workshop on Graph Neural Networks and Systems

1 Introduction

Graph Neural Networks (GNNs) are state-of-the-art models that learn representations from complex graph-structured data in various domains such as drug discovery [68, 75, 93], social network [26, 87, 29, 99], recommendation systems [85, 49, 22, 96], and traffic flow modeling [83, 12]. However, due to privacy concerns, regulatory restrictions, and commercial competition, there are widespread real-world cases in which graph data is decentralized. For example, in the AI-based drug discovery industry, pharmaceutical research institutions would significantly benefit from other institutions' data, but neither can afford to disclose their private data due to commercial reasons.



(a) Graph-level FL (b) Subgraph-level FL (c) Node-level FL

Figure 1: Three settings of graph federated learning.

Federated Learning (FL) is a distributed learning paradigm that addresses this data isolation problem. In FL, training is an act of collaboration between multiple clients without requiring centralized local data [54, 39]. Despite its successful application in domains like computer vision [32, 48, 37, 36] and natural language processing [27, 22, 46], FL has yet to be widely adopted in the domain of machine learning on graph data. There are multiple reasons for this:

1. There is a lack of unified formulation over the various graph FL settings and tasks in current literature, making it difficult for researchers who focus on SGD-based federated optimization algorithms to understand essential challenges in federated GNNs;
2. Existing FL libraries, as summarized by [33], do not support diverse datasets and learning tasks to benchmark different models and training algorithms. Given the complexity of graph data, the dynamics of training GNNs in an FL setting may be different from training vision or language models. A fair and easy-to-use benchmark with standardized open datasets and reference implementations is essential to the development of new graph FL models and algorithms;
3. The simulation-oriented federated training system is inefficient and unsecure for federated GNNs research on large-scale and private graph datasets in the cross-silo settings. Disruptive research ideas may be constrained by the lack of a modularized federated training system tailored for diverse GNN models and FL algorithms.

To address these issues, we present an open FL benchmark system for GNNs, called `FedGraphNN`, which contains a variety of graph datasets from different domains and eases the training and evaluation of various GNN models and FL algorithms. We first formulate graph FL to provide a unified framework for federated GNNs (Section 2). Under this formulation, we introduce the various graph datasets with synthesized partitions according to real-world application scenarios (Section 3). An efficient and secure FL system is designed and implemented to support popular GNN models and FL algorithms and provide low-level programmable APIs for customized research and industrial deployment (Section 4). Extensive empirical analysis demonstrates the utility and efficiency of our system and indicates the need of further research in graph FL (Section 5). Finally, we summarize the open challenges in graph FL based on emerging related works (Section 6) as well as future directions based on `FedGraphNN` (Section 7).

2 Federated Graph Neural Networks (`FedGraphNN`)

We consider a *distributed graph scenario* in which a single graph is partitioned or multiple graphs are dispersed over multiple edge servers that cannot be centralized for training due to privacy or regulatory restrictions. However, collaborative training over the dispersed data can aid the formulation of more powerful and generalizable graph models. In this work, we focus on graph neural networks (GNNs) as the graph models and extend the emerging studies on federated learning (FL) over neural network models in other domains to GNNs.

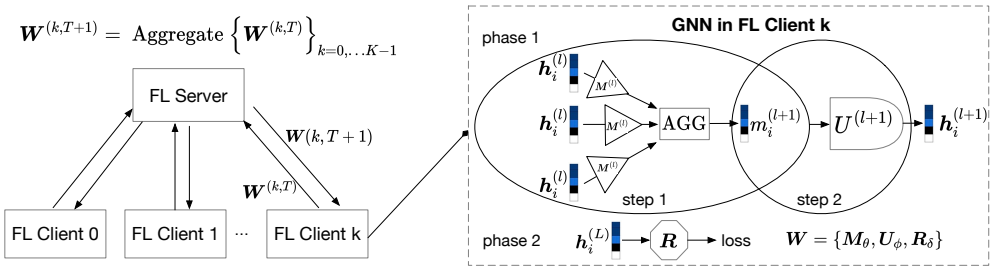


Figure 2: Formulation of FedGraphNN (Federated Graph Neural Network)

In our unified framework of FedGraphNN, we assume that there are K clients in the distributed graph scenario, and the k^{th} client has its own dataset $\mathcal{D}^{(k)} := (\mathcal{G}^{(k)}, \mathbf{Y}^{(k)})$, where $\mathcal{G}^{(k)} = (\mathcal{V}^{(k)}, \mathcal{E}^{(k)})$ is the graph(s) in $\mathcal{D}^{(k)}$ with vertex and edge feature sets $\boldsymbol{X}^{(k)} = \{\boldsymbol{x}_m^{(k)}\}_{m \in \mathcal{V}^{(k)}}$ and $\boldsymbol{Z}^{(k)} = \{\boldsymbol{e}_{m,n}^{(k)}\}_{m,n \in \mathcal{V}^{(k)}}$, $\mathbf{Y}^{(k)}$ is the label set of $\mathcal{G}^{(k)}$. Each client owns a GNN model to learn graph representations and make predictions. Multiple clients are interested in collaborating through a server to improve their GNN models without necessarily revealing their graph datasets.

We illustrate the formulation of FedGraphNN in Figure 2. Without loss of generality, we use a Message Passing Neural Network (MPNN) framework [24, 69]. Most of the spatial-based GNN models [41, 79, 26] can be unified into this framework, where the forward pass has two phases: a message-passing phase and a readout phase.

GNN phase 1: Message-passing (same for all tasks). The message-passing phase contains two steps: (1) the model gathers and transforms the neighbors' messages, and (2) the model uses aggregated messages to update the nodes' hidden states. Mathematically, for client k and layer indices $\ell = 0, \ldots, L-1$, an L-layer MPNN is formalized as follows:

$$\begin{aligned} \boldsymbol{m}_i^{(k,\ell+1)} &= \texttt{AGG}\left(\left\{\boldsymbol{M}_\theta^{(k,\ell+1)}\left(\boldsymbol{h}_i^{(k,\ell)}, \boldsymbol{h}_j^{(k,\ell)}, \boldsymbol{z}_{i,j}\right) \mid j \in \mathcal{N}_i\right\}\right), \\ \boldsymbol{h}_i^{(k,\ell+1)} &= \boldsymbol{U}_\phi^{(k,\ell+1)}\left(\boldsymbol{h}_i^{(k,\ell)}, \boldsymbol{m}_i^{(k,\ell+1)}\right), \end{aligned} \tag{1}$$

where $\boldsymbol{h}_i^{(k,0)} = \boldsymbol{x}_i^{(k)}$ is the k^{th} client's node features, ℓ is the layer index, AGG is the aggregation function (e.g., in the GCN model, the aggregation function is a simple SUM operation), $\mathcal{N}_i$ is the neighbor set of node i, and $\boldsymbol{M}_\theta^{(k,\ell+1)}(\cdot)$ is the message generation function which takes the hidden state of current node $\boldsymbol{h}_i$, the hidden state of the neighbor node $\boldsymbol{h}_j$ and the edge features $\boldsymbol{z}_{i,j}$ as inputs. $\boldsymbol{U}_\phi^{(k,\ell+1)}(\cdot)$ is the state update function receiving the aggregated feature $\boldsymbol{m}_i^{(k,\ell+1)}$.

GNN phase 2: Readout (different across tasks). After propagating through an L-layer MPNN, the readout phase computes feature vectors from the hidden states of the last MPNN layer and makes predictions for downstream tasks, that is

$$\hat{y}_S^{(k)} = \boldsymbol{R}_\delta\left(\left\{h_i^{(k,L)} \mid i \in \mathcal{V}_S^{(k)}\right\}\right). \tag{2}$$

Note that to handle different downstream tasks, S can be a single node (node classification), a node pair (link prediction), a node set (graph classification) and so forth, and $\boldsymbol{R}_\delta$ can be the concatenation function or a pooling function such as SUM plus a single- or multi-layer perceptron.

GNN with FL. To formulate the FL setting, we define $\boldsymbol{W} = \{\boldsymbol{M}_\theta, \boldsymbol{U}_\phi, \boldsymbol{R}_\delta\}$ as the overall learnable weights in the GNN of client k. Consequently, we formulate FedGraphNN as a distributed optimization problem as follows:

$$\min_{\boldsymbol{W}} F(\boldsymbol{W}) \stackrel{\text{def}}{=} \min_{\boldsymbol{W}} \sum_{k=1}^{K} \frac{N^{(k)}}{N} \cdot f^{(k)}(\boldsymbol{W}), \tag{3}$$

where $f^{(k)}(\boldsymbol{W}) = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} \mathcal{L}(\boldsymbol{W}; x_i^{(k)}, z_i^{(k)}, y_i^{(k)})$ is the k^{th} client's local objective function that measures the local empirical risk over the graph dataset $\mathcal{D}^{(k)}$ with $N^{(k)}$ data samples. $\mathcal{L}$ is the loss function of the global GNN model. To solve this problem, the most straightforward algorithm

is FedAvg [54]. It is important to note here that in FedAvg, the aggregation function on the server merely averages model parameters. We use GNNs inductively (i.e., the model is independent of the structure of the graphs it is trained on). Thus, no topological information about graphs on any client is required on the server during parameter aggregation. Other advanced algorithms such as FedOPT [62], FedGKT [32], and Decentralized FL [28, 35] can also be applied.

Under the unified framework of `FedGraphNN`, we organize various distributed graph scenarios motivated by real-world applications into three settings based on how the graphs are distributed across silos, and provide support to the corresponding typical tasks in each setting.

- **Graph-level `FedGraphNN`:** Each client holds a set of graphs, where the typical task is graph classification/regression. Real-world scenarios include molecular trials [68], protein discovery [101] and so on, where each institute might hold a limited set of graphs with ground-truth labels due to expensive experiments.
- **Subgraph-level `FedGraphNN`:** Each client holds a subgraph of a larger global graph, where the typical task is node classification and link prediction. Real-world scenarios include recommendation systems [102], knowledge graph completion [10] and so forth, where each institute might hold a subset of user-item interaction data or entity/relation data.
- **Node-level `FedGraphNN`:** Each client holds the ego-networks of one or multiple nodes, where the typical task is node classification and link prediction. Real-world scenarios include social networks [108], sensor networks [103], etc., where each node only sees its k-hop neighbors and their connections in the large graph.

Supported GNN models and FL algorithms. `FedGraphNN`'s latest release supports the GNN models of GCN [41], GAT [79], GraphSage [26], SGC [86], and GIN [91], implemented via PyTorch Geometric [17]. For FL algorithms, aside from FedAvg [54], other advanced algorithms such as FedOPT [62] are also supported within FedML library [33]. We refer to the Appendix A for more details on the supported GNN baselines.

3 `FedGraphNN` Open Datasets

`FedGraphNN` is centered around three federated GNN settings based on the ways graph data can be distributed in real-world scenarios, which covers a broad range of domains, tasks and challenges of graph FL. Specifically, it includes 36 datasets from 7 domains, such as molecules, proteins, knowledge graphs, recommendation systems, citation networks and social networks. Here, to facilitate clear understanding over the various graph FL settings, we organize and introduce examples of real-world datasets in each of the three federated GNN settings. Exact sources and statistics of the datasets are provided in Table 1, while more details are provided in Appendix B.

- **Graph-level Setting:** In the real world, biomedical institutions might hold their own set of graphs such as molecules and proteins, and social network companies might hold their own set of community graphs. Such graphs may constitute large and diverse datasets for GNN traning, but they cannot be directly shared across silos. To simulate such scenarios, we utilize datasets from the domains of molecular machine learning [88], bioinformatics [8, 71, 26] and social computing [92], We also introduce a new large-scale dataset, called `hERG` [19] for federated drug discovery.
- **Subgraph-level Setting:** The first realistic scenario of subgraph-level FL is recommendation systems, where the users can interact with items owned by different shops or sectors, which makes each data owner only holding a part of the global user-item graph. To simulate such scenarios, we use recommendation datasets from both publicly available sources [77, 64] and internal sources [30], which have high-quality meta-data information. Another realistic scenario is knowledge graphs, where different organizations or departments might only have a subset of the entire knowledge, due to the focus in particular domains. We integrate the FB15k-237 [15], WN18RR [78] and YAGO3-10 [50] datasets, where subgraphs can be build based on relation types to distinguish specialized fields or communities to distinguish the entities of focus.
- **Node-level Setting:** In social networks, each user's personal data can be sensitive and only visible to his/her k-hop neighbors (e.g., in Instagram, $k = 1$ for contents and $k = 2$ for links, of private accounts). Thus, it is natural to consider node-level FL in social networks with clients

Table 1: Summary of open graph datasets from various domains contained in FedGraphNN.

Task-Level	Category	Datasets	# Graphs	Avg. # Nodes	Avg. # Edges	Avg. Degree	# Classes
Graph-Level	Molecules	BACE[74]	1513	34.12	36.89	2.16	2
		HIV[65]	41127	25.53	27.48	2.15	2
		MUV[66]	93087	24.23	26.28	2.17	17
		Clintox [21]	1478	26.13	27.86	2.13	2
		SIDER [42]	1427	33.64	35.36	2.10	27
		Toxcast[63]	8575	18.78	19.26	2.05	167
		Tox21 [1]	7831	18.51	25.94	2.80	12
		BBBP [52]	2039	24.05	25.94	2.16	2
		QM9 [18]	133885	8.8	27.6	6.27	1
		ESOL [14]	1128	13.29	40.65	6.11	1
		FreeSolv[56]	642	8.72	25.6	5.87	1
		Lipophilicity[18]	4200	27.04	86.04	6.36	1
		hERG [19]	10572	29.39	94.09	6.40	1
		MUTAG[13]	188	17.93	19.79	2.21	2
		NCI1[80]	4110	29.87	32.3	2.16	2
	Proteins	PROTEINS[8]	1113	39.06	72.82	3.73	2
		DDI [71]	1178	284.32	715.66	5.03	2
		PPI[26]	24	56,944	818,716	28.76	121
	Social networks	COLLAB[92]	5000	74.49	2457.78	65.99	3
		REDDIT-B[92]	2000	429.63	497.75	2.32	2
		REDDIT-M-5K[92]	4999	508.52	594.87	2.34	5
		IMDB-B[92]	1000	19.77	96.53	9.77	2
		IMDB-M[92]	1500	13	65.94	10.14	3
Subgraph-Level	Recomm. systems	Ciao [77]	28	5150.93	19280.93	3.74	5
		Epinions [64]	27	15824.22	66420.52	4.20	5
		Tencent [30]	1	709074	991713	2.80	2
	Knowledge graphs	FB15k-237 [15]	1	14505	212110	14.62	237
		WN18RR [78]	1	40559	71839	1.77	11
		YAGO3-10 [50]	1	123143	774182	6.29	37
Node-level	Publication networks	CORA [53]	1	2708	5429	2.00	7
		CORA-full [5]	1	19793	65311	3.30	70
		CITESEER [23]	1	4230	5358	1.27	6
		PUBMED [72]	1	19717	44338	2.25	3
		DBLP [76]	1	17716	105734	5.97	4
	Social networks	CS [73]	1	18333	81894	4.47	15
		Physics [73]	1	34493	247962	7.19	5

holding the user ego-networks. To simulate this scenario, we use the open social networks [73] and publication networks [53, 5, 23, 72, 76] and partition them into sets of ego-networks.

In terms of graph mining tasks, FedGraphNN supports all three common tasks of graph classification, node classification and link prediction. Some tasks are naturally important in certain graph FL settings while others are not, which we also clarify with real examples as follows

- **Graph Classification:** This task is to categorize different types of graphs based on their structure and overall information. Unlike other tasks, this requires to characterize the property of the entire input graph. This task is naturally important in graph-level FL, with real examples such as molecule property prediction, protein function prediction, and social community classification.
- **Link Prediction:** This task is to estimate the probability of links between any two nodes in a graph. It is important in the subgraph-level FL, for example, in recommendation systems and knowledge graphs, where link probabilities are predicted in the former, and relation types are predicted in the latter. It is less likely but still viable in the node-level setting, where friend suggestion and social relation profiling can be attempted in users' ego-networks, for example.
- **Node Classification:** This task is to predict the labels of individual nodes in graphs. It is more important in node-level FL, such as predicting the active research fields of an author based on his/her k-hop collaborators or habits of a user based on his/her k-hop friends. It might also be important in subgraph-level FL, such as the collaborative prediction of disease infections based on the patient networks dispersed in multiple healthcare facilities.

Data sources. We exhaustively examine and collecte 36 datasets from 7 domains. Among them, 34 are from publicly available sources such as MoleculeNet [88] and graph kernels datasets [8]. In addition, we introduce two new de-identified datasets based on our collaboration with Tencent: *hERG* [40, 20], a graph dataset for classifying protein molecules responsible for cardiac toxicity and *Tencent* [29], a large bipartite graph representing the relationships between users and groups. More details and their specific preprocessing procedures can be found in Appendices B.1 & B.2. We plan to continually enrich the available datasets in the future through active collection of open datasets and collaboration with industrial partners.

Generating Federated Learning Datasets . Unlike traditional ML banchmarking datasets, graph datasets and real-world graphs may exhibit non-I.I.D.-ness due to sources such as structure and feature heterogeneity [97, 98, 95]. Coupled with FL, multiple sources of non-I.I.D-ness are indistinguishable. Here, our focus is on how to produce *reproducible* and *statistical (sample-based)* non-I.I.D.ness. To generate *sample-based* non-I.I.D.ness, we use the unbalanced partition algorithm Latent Dirichlet Allocation (LDA) [33] to partition datasets, and this method can be applied regardless of data domain. In addition, researchers and practitioners can also synthesize non-I.I.D.-ness using our available additional meta-data. Figure 6 in the Appendix shows several datasets' non-I.I.D. distributions generated using the LDA method. The alpha values for LDA for representative datasets can be found in Tables 2,3,4 and 20 in the Appendix E.3. Yet, deeply decoupling and quantifying non-I.I.D.-ness in federated GNNs remains as an open problem [84, 89].

In summary, we provide a comprehensive study and solutions for several challenges in data collection and benchmark for graph FL: (1) Collecting, analyzing and categrozing a large number of public, real-world datasets into different federated GNN settings with corresponding typical tasks; (2) Standardizing the procedure to synthesize non-I.I.D. data distributions for all graph-structured datasets through providing a novel partition method and associative meta-data.

4 `FedGraphNN` Benchmark System: Efficient, Secure, and Modularized

We have released an open-source distributed training system for `FedGraphNN`. This system is tailored for benchmarking graph FL and promoting algorithmic innovations with three key advantageous designs in the context of FL that have not been supported by existing simulation-oriented benchmark and libraries [33].

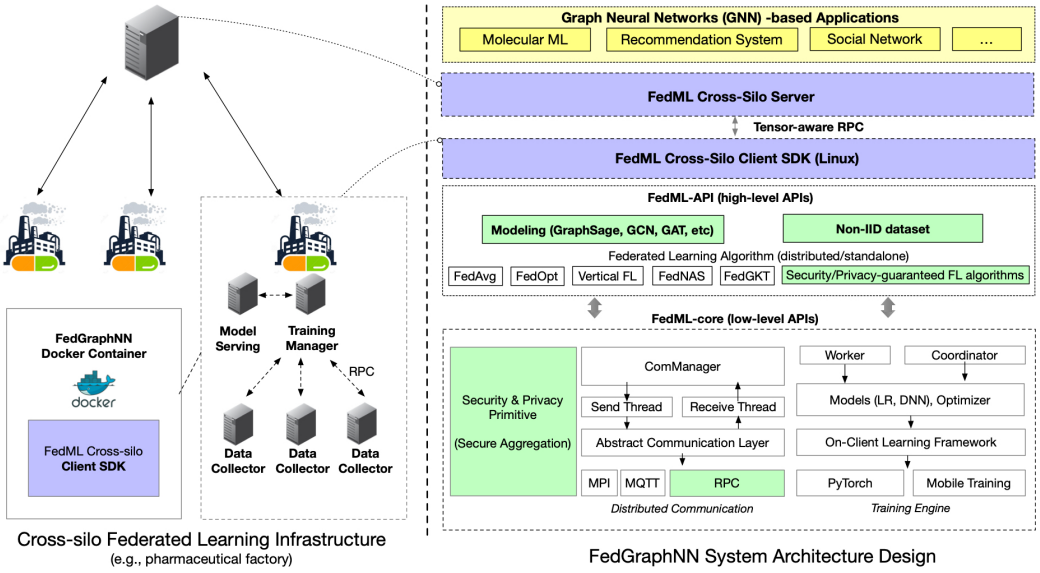


Figure 3: Overview of `FedGraphNN` System Architecture Design

Enhancing realistic evaluation with efficient and deployable distributed system design. We design the training system to support realistic distributed computing in multiple edge servers, given that `FedGraphNN` is mainly executed in the cross-silo settings where each FL client represents an edge server belonging to an organization rather than smartphone or IoT devices. The system architecture, shown in Figure 3, is composed of three layers: `FedML-core` layer, `FedML-API` layer, and `Application` layer. `FedML-core` layer supports both RPC (remote procedure call) and MPI (message passing interface), which enable communication among edge servers located at different data centers. More specially, the RPC API is tensor-oriented, meaning that its weight or gradient transmit among clients is much faster than naïve gRPC with GPU-direct communication (e.g., data owners from different AWS EC2 accounts can utilize this feature for faster training). The communication primitives are wrapped as abstract communication APIs (i.e., `ComManager` in Figure 3) to simplify the message definition and passing requested by different FL algorithms in `FedML-API` layer (see more details in Appendix C). In the deployment perspective, the FL client

library should be compatible with heterogeneous hardware and OS configuration. To meet this goal, we provide Docker containers to simplify the large-scale deployment for federated training.

With the help of the above design, researchers can run realistic evaluations in a parallel computing environment where multiple CPU/GPU servers are located in multiple organizations (e.g., edge servers in AWS EC2). As such, for medium and small-scale graph datasets, the training can be finished in only a few minutes. For large-scale graph datasets, researchers can also measure system-wise performance (communicational and computational costs) to have a tangible trade-off between accuracy and system efficiency. Scaling up to numerous edge servers (FL clients) is further simplified by Docker-based deployment.

Enabling secure benchmarking with lightweight secure aggregation. Researchers in the industry may also need to explore FL on their private customer datasets with other organizations. However, model weights from clients may still have the risk of privacy leakage [110]. As such, legal and regulatory departments normally do not permit FL research on private customer datasets when strong security is not guaranteed. To break this barrier, we integrate secure aggregation (SA) algorithms to the `FedGraphNN` system. ML researchers do not need to master security-related knowledge but also enjoy a secure distributed training environment. To be more specific, we support `FedGraphNN` with `LightSecAgg`, a state-of-the-art SA algorithm developed by our team (Appendix D.2). In high-level understanding, `LightSecAgg` protects the client model by generating a single random mask and allows their cancellation when aggregated at the server. Consequently, the server can only see the aggregated model and not the raw model from each client. The design and implementation of `LightSecAgg` spans across `FedML-core` and `FedML-API` in the system architecture, as shown in Figure 3. Baselines such as SecAgg [7] and SecAgg+ [3] are also supported.

Facilitating algorithmic innovations with diverse datasets, GNN models, and FL algorithms. `FedGraphNN` also aims to enable flexible customization for future algorithmic innovations. To support diverse datasets, GNN models, and FL algorithms, we have modularized the API and component design. All data loaders follow the same format of input and output arguments, which are compatible with different models and algorithms, and easy to support new datasets. The method of defining the model and related trainer is kept the same as in centralized training to reduce the difficulty of developing the distributed training framework. For new FL algorithm development, worker-oriented programming reduces the difficulty of message passing and definition (details are introduced in the Appendix C). Diverse algorithmic implementations serve as the reference source code for future algorithmic innovation. The user-oriented interface (main training script) is simplified as the example code shown in Figure 4 where a few lines of code can launch a federated training in a cross-silo cloud environment.

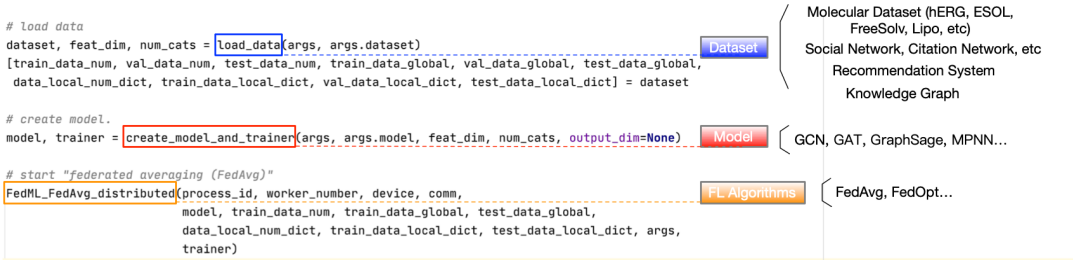


Figure 4: Example code for benchmark evaluation with `FedGraphNN`

5 `FedGraphNN` Empirical Analysis

5.1 Experimental Setup

Our experiments are conducted on multiple GPU servers each equipped with 8 NVIDIA Quadro RTX 5000 (16GB GPU memory). The hyper-parameters are selected via our built-in efficient parameter sweeping functionalities from the ranges listed in Appendix E.1. We present results on the ROC-AUC metric for graph classification and RMSE & MAE for graph regression, MAE, MSE, and RMSE for link prediction, and micro-F1 for node classification. More evaluation metrics supported by `FedGraphNN` are presented in Appendix E.2.

5.2 Baseline Performance Analysis

We report experimental results of several popular GNN models trained with the most widely used FL algorithm of FedAvg, to examplify the utility of `FedGraphNN`. More results with varying baselines, hyper-parameters, evaluation metrics and visualizations are being updated and presented in Appendix E.3. After hyper-parameter tuning, we present the main performance results as well as runtimes in Tables 2, 3 and 4.

Table 2: Performance of graph classification in the graph-level FL setting (#clients=4).

Metric	ROC-AUC					Training Time (sec.)				
Method	SIDER $\alpha=0.2$	BACE $\alpha=0.5$	CLINTOX $\alpha=0.5$	BBBP $\alpha=2$	TOX21 $\alpha=3$	SIDER	BACE	CLINTOX	BBBP	TOX21
MoleculeNet Results	0.6380	0.8060	0.8320	0.6900	0.8290	Not Published				
GCN (Centralized)	0.6476	0.7657	0.8914	0.8705	0.7800	458	545	686	532	1034
GCN (FedAvg)	0.6266	0.6594	0.8784	0.7629	0.7128	358	297	280	253	903
GAT (Centralized)	0.6639	0.9221	0.9573	0.8824	0.8144	739	603	678	533	2045
GAT (FedAvg)	0.6591	0.7714	0.9129	0.8746	0.7186	528	327	457	328	1549
GraphSAGE (Centralized)	0.6669	0.9266	0.9716	0.8930	0.8317	193	327	403	312	1132
GraphSAGE (FedAvg)	0.6700	0.8604	0.9246	0.8935	0.7801	127	238	282	206	771

Table 3: Performance of link prediction in the subgraph-level FL setting (#clients = 8).

Metric	MAE		MSE		RMSE		Training Time (sec.)	
DataSet	CIAO	EPINIONS	CIAO	EPINIONS	CIAO	EPINIONS	CIAO	EPINIONS
GCN (Centralized)	0.8167	0.8847	1.1184	1.3733	1.0575	1.1718	268	650
GCN (FedAvg)	0.7995	0.9033	1.0667	1.4378	1.0293	1.1924	352	717
GAT (Centralized)	0.8214	0.8934	1.1318	1.3873	1.0639	1.1767	329	720
GAT (FedAvg)	0.7987	0.9032	1.0682	1.4248	1.0311	1.1882	350	749
GraphSAGE (Centralized)	0.8231	1.0436	1.1541	1.8454	1.0742	1.3554	353	721
GraphSAGE (FedAvg)	0.8290	0.9816	1.1320	1.6136	1.0626	1.2625	551	810

Table 4: Performance of Node classification in the node-level FL setting (#clients = 10).

Metric	micro F1				Training Time (sec.)			
Method	CORA	CITESEER	PUBMED	DBLP	CORA	CITESEER	PUBMED	DBLP
GCN (Centralized)	0.8622	0.9820	0.9268	0.9294	1456	742	1071	1116
GCN (FedAvg)	0.8549	0.9743	0.9128	0.9088	833	622	654	653
GAT (Centralized)	diverge	0.9653	0.8621	0.8308	1206	1765	1305	957
GAT (FedAvg)		0.9610	0.8557	0.8201	871	652	682	712
GraphSAGE (Centralized)	0.9692	0.9897	0.9724	0.9798	1348	934	692	993
GraphSAGE (FedAvg)	0.9749	0.9854	0.9761	0.9749	774	562	622	592

Besides showcasing the utility of `FedGraphNN`, there are multiple takeaways from these results:

1. When the graph datasets are small, FL accuracy is often on par with centralized learning.
2. When dataset sizes grow, FL accuracy becomes significantly worse than centralized learning. We conjecture that it is the complicated non-I.I.D. nature of graphs that leads to the accuracy drop in larger datasets.
3. The dynamics of training GNNs in a federated setting are different from training federated vision or language models. Our findings show that the best model in the centralized setting may not necessarily be the best model in the FL setting.
4. Counterintuitive phenomenons (highlights in above tables) further add to the mystery of federated graph neural networks: in graph-level experiments, GAT suffers the most performance compromise on 5 out of 9 datasets; in both subgraph-level and node-level FL, results on some datasets (CIAO, CORA, PubMed) may even have slightly higher performance than centralized training; GAT cannot achieve reasonable accuracy in node-level FL (e.g., in CORA dataset), etc.

These results indicate the limitations of the baselines in `FedGraphNN` and motivate much further research in understanding the nuances and improving the training GNNs in the FL setting.

Evaluation on System Efficiency and Security. We provide additional results on system performance evaluation, where the results are summarized in Appendix D.1. Depending on the size of the graph data, `FedGraphNN` can complete the training efficiently. The training time ranges from a few minutes to about 1 hour even for large-scale graphs. In the security aspect, the main result of `LightSecAgg` is provided in Appendix D.2. The key benefit is that it not only obtains the same level of privacy guarantees as to the state-of-the-art (SecAgg [7] and SecAgg+ [3]) but also substantially reduces the aggregation complexity (hence much faster training).

6 Related Works and Open Challenges

`FedGraphNN` lies at the intersection of GNNs and FL. We first discuss related works under the umbrella of three different graph FL settings. (1) *Graph-level* (Figure 1(a)): we believe molecular machine learning is a paramount application in this setting, where many small graphs are distributed between multiple institutions, as demonstrated in [35, 89]. [89] proposes a clustered FL framework specifically for GNNs to deal with feature and structure heterogeneity. [35] develops a multi-task learning framework suitable to train federated graph-level GNNs without the need for a central server. (2) *Subgraph-level* (Figure 1(b)): this scenario typically pertains to the entire social networks, recommender networks or knowledge graphs that need to be partitioned into many smaller subgraphs due to data barriers between different departments in a giant company or data platforms with different domain focuses as demonstrated in [85, 105]. [85] proposes a federated recommendation system with GNNs, whereas [105] proposes FedSage, a subgraph-level federated GNN generating psuedo-neighbors utilizing variational graph autoencoder. (3) *Node-level* (Figure 1(c)): when the privacy of specific nodes in a graph is important, node-level graph FL is useful in practice. The IoT setting is a good example [107]; [81] uses a hybrid method of FL and meta-learning to solve the semi-supervised graph node classification problem in decentralized social network datasets; [55] attempts to protect the node-level privacy using an edge-cloud partitioned GNN model for spatio-temporal forecasting tasks using node-level traffic sensor datasets.

Before our unified system of `FedGraphNN`, there was a serious lack of standardized datasets and baselines for training graph-neural networks in a federated setting pertains. Previous platforms like LEAF [9], TensorFlow Federated [2, 6], PySyft [70], and FATE have no support on graph datasets and GNN models. Beyond the direct goals of `FedGraphNN`, many open algorithmic challenges in graph FL remain be studied. First, integrating both graph topology of GNNs and network topology of FL in a principled and efficient way is highly dataset- and application-specific. Second, the partitioning of a single graph into subgraphs or ego-networks for some tasks introduce dataset bias and information loss in terms of missing cross-subgraph links [105]. Third, decoupling and quantifying the multiple sources of non-I.I.D.-ness coming from both features and structures in graphs are crucial for the appropriate design of federated GNNs [89, 84]. Finally, in terms of communication efficiency and security, previous works utilize various methods including Secure Multi-Party Computation (SMPC) [109], Homomorphic Encryption (HE) [109], secure aggregation [38] and Shamir's secret sharing [107], but their comparison is missing.

7 Conclusions and Future Works

In this work, we design an FL system and benchmark for GNNs, named `FedGraphNN`, which includes open datasets, baseline implementations, programmable APIs, all integrated in a robust system affordable to most research labs. We hope `FedGraphNN` can serve as an easy-to-use research platform for researchers to explore vital problems at the intersection of FL and GNNs.

Here we highlight some future improvements and research directions based on our `FedGraphNN` system: 1. supporting more graph datasets and GNN models for diverse applications. Possible applications include and are not limited to sensor networks and spatio-temporal forecasting [47, 94]; 2. optimizing the system to further accelerate the training speed for large graphs [106, 44]; 3. designing advanced graph FL algorithms to mitigate the accuracy gap on datasets with non-I.I.D.ness, such as tailoring FedNAS [31, 34] to search for personalized GNN models for individual FL clients; 4. exploring label-efficient GNN models based on concepts such as meta-learning and self-supervision to exploit the graphs in each client and their collaboration [90]; 5. addressing challenges in security and privacy under the setting of Federated GNN [16, 57–59, 100, 11]; 6. proposing efficient compression algorithms that adapt to the level of compression to the available bandwidth of the users while preserving the privacy of users' local data; 7. organizing data competitions, themed workshops, special issues, etc., on the dissemination of `FedGraphNN`; 8. actively discussing ethics and societal impacts to avoid unwanted negative effects.

References

[1] Tox21 challenge. https://tripod.nih.gov/tox21/challenge/, 2017.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[3] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1253–1269, 2020.

[4] Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

[5] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1ZdKJ-0W.

[6] K Bonawitz, H Eichner, W Grieskamp, et al. Tensorflow federated: Machine learning on decentralized data. 2020.

[7] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

[8] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[9] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings, 2019.

[10] Chaochao Chen, Jamie Cui, Guanfeng Liu, Jia Wu, and Li Wang. Survey and open problems in privacy preserving knowledge graph: Merging, query, representation, completion and applications. *arXiv preprint arXiv:2011.10180*, 2020.

[11] Liang Chen, Jintang Li, Qibiao Peng, Yang Liu, Zibin Zheng, and Carl Yang. Understanding structural vulnerability in graph convolutional networks. In *IJCAI*, 2021.

[12] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, Ziyuan Pu, and Yinhai Wang. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, 2019.

[13] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

[14] John S Delaney. Esol: estimating aqueous solubility directly from molecular structure. *Journal of chemical information and computer sciences*, 44(3):1000–1005, 2004.

[15] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, pp. 1811–1818, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17366.

[16] Ahmed Roushdy Elkordy and A. Salman Avestimehr. Secure aggregation with heterogeneous quantization in federated learning. *preprint arXiv:2009.14388*, 2020.

[17] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019.

[18] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

[19] Anna Gaulton, Anne Hersey, Michał Nowotka, A. Patrícia Bento, Jon Chambers, David Mendez, Prudence Mutowo, Francis Atkinson, Louisa J. Bellis, Elena Cibrián-Uhalte, Mark Davies, Nathan Dedman, Anneli Karlsson, María Paula Magariños, John P. Overington, George Papadatos, Ines Smit, and Andrew R. Leach. The ChEMBL database in 2017. *Nucleic Acids Research*, 45(D1):D945–D954, 11 2016. ISSN 0305-1048. doi: 10.1093/nar/gkw1074. URL https://doi.org/10.1093/nar/gkw1074.

[20] Anna Gaulton, Anne Hersey, Michał Nowotka, A Patricia Bento, Jon Chambers, David Mendez, Prudence Mutowo, Francis Atkinson, Louisa J Bellis, Elena Cibrián-Uhalte, et al. The chembl database in 2017. *Nucleic acids research*, 45(D1):D945–D954, 2017.

[21] Kaitlyn M Gayvert, Neel S Madhukar, and Olivier Elemento. A data-driven approach to predicting successes and failures of clinical trials. *Cell chemical biology*, 23(10):1294–1301, 2016.

[22] Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Fedner: Privacy-preserving medical named entity recognition with federated learning. *arXiv e-prints*, pp. arXiv–2003, 2020.

[23] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, DL '98, pp. 89–98, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919653. doi: 10.1145/276675.276685. URL https://doi.org/10.1145/276675.276685.

[24] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017.

[25] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman (eds.), *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, Pasadena, CA USA, 2008.

[26] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017. URL http://arxiv.org/abs/1706.02216.

[27] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[28] Chaoyang He, Conghui Tan, Hanlin Tang, Shuang Qiu, and Ji Liu. Central server free federated learning over single-sided trust social networks. *arXiv preprint arXiv:1910.04956*, 2019.

[29] Chaoyang He, Tian Xie, Yu Rong, W. Huang, Junzhou Huang, X. Ren, and C. Shahabi. Cascade-bgnn: Toward efficient self-supervised representation learning on large-scale bipartite graphs. *arXiv preprint arXiv:1906.11994*, 2019.

[30] Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. Cascade-bgnn: Toward efficient self-supervised representation learning on large-scale bipartite graphs. *arXiv preprint arXiv:1906.11994*, 2019.

[31] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Fednas: Federated deep learning via neural architecture search. *arXiv preprint arXiv:2004.08546*, 2020.

[32] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33, 2020.

[33] 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. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[34] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11993–12002, 2020.

[35] Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. Spreadgnn: Serverless multi-task federated learning for graph neural networks, 2021.

[36] Chaoyang He, Alay Dilipbhai Shah, Zhenheng Tang, Di Fan, Adarshan Naiynar Sivashunmugam, Keerti Bhogaraju, Mita Shimpi, Li Shen, Xiaowen Chu, Mahdi Soltanolkotabi, et al. Fedcv: A federated learning framework for diverse computer vision tasks. 2021.

[37] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. *arXiv preprint arXiv:2003.08082*, 2020.

[38] Meng Jiang, Taeho Jung, Ryan Karl, and Tong Zhao. Federated dynamic gnn with secure aggregation. *arXiv preprint arXiv:2009.07351*, 2020.

[39] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[40] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. Pubchem in 2021: new data content and improved web interfaces. *Nucleic Acids Research*, 49(D1):D1388–D1395, 2021.

[41] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

[42] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079, 2016.

[43] Greg Landrum. Rdkit: Open-source cheminformatics, 2006. URL http://www.rdkit.org.

[44] S. Lee, Q. Kang, A. Agrawal, A. Choudhary, and W. k. Liao. Communication-efficient local stochastic gradient descent for scalable deep learning. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 718–727, 2020. doi: 10.1109/BigData50022.2020.9378178.

[45] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[46] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: A research platform for federated learning in natural language processing. *arXiv preprint arXiv:2104.08815*, 2021.

[47] Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Zhao Xu, Haiyang Yu, Jingtun Zhang, Yi Liu, et al. Dig: A turnkey library for diving into graph deep learning research. *arXiv preprint arXiv:2103.12608*, 2021.

[48] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. Fedvision: An online visual object detection platform powered by federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13172–13179, 2020.

[49] Zhiwei Liu, Mengting Wan, Stephen Guo, Kannan Achan, and Philip S Yu. Basconv: Aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 64–72. SIAM, 2020.

[50] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, Asilomar, United States, January 2013. URL https://hal-imt.archives-ouvertes.fr/hal-01699874.

[51] Elan Markowitz, Keshav Balasubramanian, Mehrnoosh Mirtaheri, Sami Abu-El-Haija, Bryan Perozzi, Greg Ver Steeg, and Aram Galstyan. Graph traversal with tensor functionals: A meta-algorithm for scalable learning, 2021.

[52] Ines Filipa Martins, Ana L Teixeira, Luis Pinheiro, and Andre O Falcao. A bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*, 52(6):1686–1697, 2012.

[53] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

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

[55] Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. Cross-node federated graph neural network for spatio-temporal data modeling, 2021. URL https://openreview.net/forum?id=HWX5j6Bv_ih.

[56] David L Mobley and J Peter Guthrie. Freesolv: a database of experimental and calculated hydration free energies, with input files. *Journal of computer-aided molecular design*, 28(7): 711–720, 2014.

[57] Saurav Prakash and Amir Salman Avestimehr. Mitigating byzantine attacks in federated learning. *arXiv preprint arXiv:2010.07541*, 2020.

[58] Saurav Prakash, Sagar Dhakal, Mustafa Riza Akdeniz, Yair Yona, Shilpa Talwar, Salman Avestimehr, and Nageen Himayat. Coded computing for low-latency federated learning over wireless edge networks. *IEEE Journal on Selected Areas in Communications*, 39(1):233–250, 2020.

[59] Saurav Prakash, Amirhossein Reisizadeh, Ramtin Pedarsani, and Amir Salman Avestimehr. Coded computing for distributed graph analytics. *IEEE Transactions on Information Theory*, 66(10):6534–6554, 2020.

[60] Dragomir R Radev, Hong Qi, Harris Wu, and Weiguo Fan. Evaluating web-based question answering systems. In *LREC*. Citeseer, 2002.

[61] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

[62] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[63] Ann M Richard, Richard S Judson, Keith A Houck, Christopher M Grulke, Patra Volarath, Inthirany Thillainadarajah, Chihae Yang, James Rathman, Matthew T Martin, John F Wambaugh, et al. Toxcast chemical landscape: paving the road to 21st century toxicology. *Chemical research in toxicology*, 29(8):1225–1251, 2016.

[64] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the semantic web. In *International semantic Web conference*, pp. 351–368. Springer, 2003.

[65] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 287–297. Springer, 2008.

[66] Sebastian G Rohrer and Knut Baumann. Maximum unbiased validation (muv) data sets for virtual screening based on pubchem bioactivity data. *Journal of chemical information and modeling*, 49(2):169–184, 2009.

[67] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data, 2020.

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

[69] Yu Rong, Tingyang Xu, Junzhou Huang, Wenbing Huang, Hong Cheng, Yao Ma, Yiqi Wang, Tyler Derr, Lingfei Wu, and Tengfei Ma. Deep graph learning: Foundations, advances and applications. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge DiscoverY; Data Mining*, KDD '20, pp. 3555–3556, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3406474. URL https://doi.org/10.1145/3394486.3406474.

[70] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning, 2018.

[71] Isabel Segura Bedmar, Paloma Martínez, and María Herrero Zazo. Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). Association for Computational Linguistics, 2013.

[72] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[73] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation, 2019.

[74] Govindan Subramanian, Bharath Ramsundar, Vijay Pande, and Rajiah Aldrin Denny. Computational modeling of β-secretase 1 (bace-1) inhibitors using ligand based approaches. *Journal of chemical information and modeling*, 56(10):1936–1949, 2016.

[75] Mengying Sun, Sendong Zhao, Coryandar Gilvary, Olivier Elemento, Jiayu Zhou, and Fei Wang. Graph convolutional networks for computational drug development and discovery. *Briefings in Bioinformatics*, 21(3):919–935, 06 2019. ISSN 1477-4054. doi: 10.1093/bib/bbz042. URL https://doi.org/10.1093/bib/bbz042.

[76] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 990–998, 2008.

[77] Jiliang Tang, Huiji Gao, and Huan Liu. mtrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pp. 93–102, 2012.

[78] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *CVSCW*, 07 2015. doi: 10.18653/v1/W15-4007.

[79] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[80] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

[81] Binghui Wang, Ang Li, Hai Li, and Yiran Chen. Graphfl: A federated learning framework for semi-supervised node classification on graphs. *arXiv preprint arXiv:2012.04187*, 2020.

[82] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging, 2020.

[83] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. *Traffic Flow Prediction via Spatial Temporal Graph Neural Network*, pp. 1082–1092. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450370233. URL https://doi.org/10.1145/3366423.3380186.

[84] Yiqi Wang, Yao Ma, Charu Aggarwal, and Jiliang Tang. Non-iid graph neural networks, 2020.

[85] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925*, 2021.

[86] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr. au2, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks, 2019.

[87] Le Wu, Peijie Sun, Richang Hong, Yanjie Fu, Xiting Wang, and Meng Wang. Socialgcn: An efficient graph convolutional network based model for social recommendation. *CoRR*, abs/1811.02815, 2018. URL http://arxiv.org/abs/1811.02815.

[88] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[89] Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs, 2021.

[90] Yaochen Xie, Zhao Xu, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *arXiv preprint arXiv:2102.10757*, 2021.

[91] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

[92] Pinar Yanardag and S.V.N. Vishwanathan. *Deep Graph Kernels*, pp. 1365–1374. Association for Computing Machinery, New York, NY, USA, 2015. ISBN 9781450336642. URL https://doi.org/10.1145/2783258.2783417.

[93] Carl Yang, Mengxiong Liu, Vincent W Zheng, and Jiawei Han. Node, motif and subgraph: Leveraging network functional blocks through structural convolution. In *ASONAM*, 2018.

[94] Carl Yang, Chao Zhang, Xuewen Chen, Jieping Ye, and Jiawei Han. Did you enjoy the ride? understanding passenger experience via heterogeneous network embedding. In *ICDE*, 2018.

[95] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. In *NIPS*, 2019.

[96] Carl Yang, Aditya Pal, Andrew Zhai, Nikil Pancha, Jiawei Han, Chuck Rosenberg, and Jure Leskovec. Multisage: Empowering graphsage with contextualized multi-embedding on web-scale multipartite networks. In *KDD*, 2020.

[97] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous network representation learning: A unified framework with survey and benchmark. In *TKDE*, 2020.

[98] Carl Yang, Jieyu Zhang, and Jiawei Han. Co-embedding network nodes and hierarchical labels with taxonomy based generative adversarial nets. In *ICDM*, 2020.

[99] Carl Yang, Jieyu Zhang, Haonan Wang, Sha Li, Myungwan Kim, Matt Walker, Yiou Xiao, and Jiawei Han. Relation learning on social networks with multi-modal graph edge variational autoencoders. In *WSDM*, 2020.

[100] Carl Yang, Haonan Wang, Ke Zhang, Liang Chen, and Lichao Sun. Secure deep graph generation with link differential privacy. In *IJCAI*, 2021.

[101] Kevin K Yang, Zachary Wu, Claire N Bedbrook, and Frances H Arnold. Learned protein embeddings for machine learning. *Bioinformatics*, 34(15):2642–2648, 2018.

[102] Liangwei Yang, Zhiwei Liu, Yingtong Dou, Jing Ma, and Philip S Yu. Consisrec: Enhancing gnn for social recommendation via consistent neighbor aggregation. *arXiv preprint arXiv:2105.02254*, 2021.

[103] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

[104] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks, 2019.

[105] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation, 2021.

[106] Da Zheng, Chao Ma, Minjie Wang, Jinjing Zhou, Qidong Su, Xiang Song, Quan Gan, Zheng Zhang, and George Karypis. Distdgl: Distributed graph neural network training for billion-scale graphs. *arXiv preprint arXiv:2010.05337*, 2020.

[107] Longfei Zheng, Jun Zhou, Chaochao Chen, Bingzhe Wu, Li Wang, and Benyu Zhang. Asfgnn: Automated separated-federated graph neural network. *arXiv preprint arXiv:2011.03248*, 2020.

[108] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM Sigkdd Explorations Newsletter*, 10(2): 12–22, 2008.

[109] Jun Zhou, Chaochao Chen, Longfei Zheng, Xiaolin Zheng, Bingzhe Wu, Ziqi Liu, and Li Wang. Privacy-preserving graph neural network for node classification. *arXiv preprint arXiv:2005.11903*, 2020.

[110] Ligeng Zhu and Song Han. Deep leakage from gradients. In *Federated learning*, pp. 17–31. Springer, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A More Details of the Supported Graph Neural Network Architectures

- **Graph Convolutional Networks** [41] is a GNN model which is a 1^{st} order approximation to spectral GNN models. [51]
- **GraphSAGE** [26] is a general inductive GNN framework capable of generating node-level representations for unseen data.
- **Graph Attention Networks** [79] is the first attention-based GNN model. Attention is computed in a message-passing fashion.
- **Simplifying Graph Convolutional Networks** [86] is the first attention-based GNN model. Attention is computed in a message-passing fashion.
- **Graph Isomorphism Networks** [91] is the SotA method on graph classification showing that GNNs are strong at most a 1-Weisfeiler Lehman test.

B More Details of the Open Datasets

B.1 Data Sources

The details of each dataset are listed below:

Datasets for Graph-level `FedGraphNN`

- `BBBP` [52] involves records of whether a compound carries the permeability property of penetrating the blood-brain barrier.
- `SIDER` [42], or Side Effect Resource, the dataset consists of marketed drugs with their adverse drug reactions. The available
- `ClinTox` [21] includes qualitative data of drugs both approved by the FDA and rejected due to the toxicity shown during clinical trials.
- `BACE` [74] is collected for recording compounds that could act as the inhibitors of human β-secretase 1 (BACE-1) in the past few years.
- `Tox21`[1] is a dataset which records the toxicity of compounds.
- `hERG`[40, 20] is a dataset that records the gene (KCNH2) that codes for a protein known as Kv11.1 responsible for its contribution to the electrical activity of the heart to help the coordination of the heart's beating.
- `QM9` [61] is a subset of GDB-13, which records the computed atomization energies of stable and synthetically accessible organic molecules, such as HOMO/LUMO, atomization energy, etc. It contains various molecular structures such as triple bonds, cycles, amide, and epoxy.
- `ESOL` [14] is a small dataset documenting the water solubility(log solubility in mols per litre) for common organic small molecules.
- `Lipophilicity` [18] which records the experimental results of octanol/water distribution coefficient for compounds.
- `FreeSolv` [56] contains the experimental results of hydration-free energy of small molecules in water.
- `MUTAG` [13] is a collection of nitroaromatic molecules and the aim is to classify their mutagenicity on Salmonella typhimurium. Input graphs are used to represent chemical compounds, where vertices stand for atoms and are labeled by the atom type (represented by one-hot encoding), while edges between vertices represent bonds between the corresponding atoms. It includes 188 samples of chemical compounds with 7 discrete node labels.
- `PROTEINS`[8] is a dataset of protein molecules and goal is to classify whether a protein is an enzyme or not. Nodes represent the amino acids and two nodes are connected by an edge if they are less than 6 Angstroms apart.
- `NCI1`[80] is a dataset where each input graph represents a chemical compound in which each vertex is an atom of the molecule(encoded via 1-hot vector), and edges between vertices represent bonds between atoms. This dataset is used to detect whether a chemical is responsible for cell lung cancer or not.

- DDI[71] is a drug-drug interactions as well as documents describing drug-drug interactions from the DrugBank database.
- COLLAB[92] is a scientific collaboration dataset consisting of researchers' ego networks, the graph representation of a researcher and his/her collaborators' collaborations. These graphs have three possible labels to distinguish researchers' field: High Energy Physics, Condensed Matter Physics, and Astro Physics.
- IMDB-B & IMDB-M[92] are movie collaboration datasets consisting of ego-networks of 1,000 actors/actresses featured in IMDB. In each graph, nodes represent actors/actresses, and there is an edge between them if they appear in the same movie. The main difference is that the latter one has more than 2 categories.
- REDDIT-B & REDDIT-M-5K[92] are relational datasets of graphs describing online discussions on Reddit where nodes represent users, and there is an edge between them if at least one of them respond to the other's comment. The only difference is that graphs in REDDIT-B labeled according to whether it belongs to a question/answer-based community or not. There are multiple categories inside REDDIT-B.

Datasets for Subgraph-level `FedGraphNN` Our focus is for recommendation systems. A node in the graph represent a user or an item while the edge weight represents the rating score from user to item. Items are also assigned to different categories based on their characteristics. Each item belongs to at least one category.

- Ciao [77] dataset contains rating information of users given to items, and also contain item category information.
- Epinions [64] dataset is trust network dataset containing profile, ratings and trust relations of a user as triplet for each user in the network. For each rating, it has the product name and its category, the rating score, the time point when the rating is created, and the helpfulness of this rating.
- Tencent [30] dataset is a large bipartite graph representing the relation between users and groups. An edge indicates the user belongs to the connected group. The data also contains all the node features and node labels.
- FB15k-237 [15] contains knowledge base relation triples and textual mentions of Freebase entity pairs.
- WN18RR [78] is a link prediction dataset created from WordNet containing 93,003 triplets with 11 different relations of 40,943 entries.
- YAGO3-10 [50] or known as Yet Another Great Ontology 3-10, is a subset of YAGO, well-known benchmark dataset for knowledge base completion. Contains entities related to at least ten different relations. Triplets describe atributes like citizenship, profession, salary.

Datasets for Node-level `FedGraphNN`

- CORA [53] dataset is a citation network formed of 2708 scientific publications with 5429 links classified into one of seven classes. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words.
- CORA-full [5] dataset is an additional version of cora [53] which is extracted from the original entire network. It consists of 19,793 scientific publications with 65,311 links classified into one of 70 classes. The node features are represented by one-hot vectors indicating the absence/presence of words.
- CITESEER [23] is a citation network formed from 3312 scientific publications with 4732 links classified into one of six classes. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words.
- PUBMED [72] dataset consists of 19717 scientific publications with 44338 links from PubMed database pertaining to diabetes classified into one of three classes. Each publication in the dataset

is described by a TF/IDF weighted word vector from a dictionary which consists of 500 unique words.

- `DBLP` [76] is a citation network dataset where it is extracted from DBLP, ACM, MAG, and other sources. The first version contains 629,814 papers and 632,752 citations. Each paper is associated with abstract, authors, year, venue, and title. The data set can be used for clustering with network and side information, studying influence in the citation network, finding the most influential papers, topic modeling analysis, etc.
- `Coauthor-CS` [73] is a coauthor network focusing on the area of computer science, in which nodes (18,333) represent authors and links (81, 894) indicate that the connected authors present on the same paper. The features of nodes represent paper keywords for each author's papers, and the labels of nodes (15) indicate the most active study area of authors.
- `Coauthor-Physics` [73] is a coauthor network focusing on the area of physics which has the same representations of nodes, links, and labels as `Coauthor-CS`. It consists of 34,493 nodes and 247,962 links, and nodes are classified into one of 5 classes.

B.2 Data Preprocessing

Dataset Splitting. Before generating non-I.I.D.'ness for our datasets, we partition all datasets such that 80% training, 10% validation, and 10% test. This ratio can be modified and as future work, we plan to support domain-specific splits such as scaffold splitting [4] for molecular machine learning datasets.

Molecular Datasets The feature extraction process is in two steps:

1. Atom-level feature extraction and Molecule object construction using RDKit [43].
2. Constructing graphs from molecule objects using NetworkX [25].

Atom features, shown in Table 5, are the atom features we used exactly the same as in [67].

Table 5: Atom features

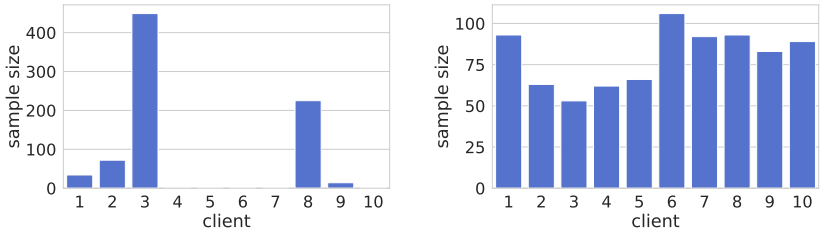
Features	Size	Description
atom type	100	Representation of atom (e.g., C, N, O), by its atomic number
formal charge	5	An integer electronic charge assigned to atom
number of bonds	6	Number of bonds the atom is involved in
chirality	5	Number of bonded hydrogen atoms
number of H	5	Number of bonded hydrogen atoms
atomic mass	1	Mass of the atom, divided by 100
aromaticity	1	Whether this atom is part of an aromatic system
hybridization	5	SP, SP2, SP3, SP3D, or SP3D2

Recommendation Systems/Knowledge Graph Datasets For recommendation systems, the data pre-processing step is partitioning the bipartite graph into subgraphs by item categories. Items belong to the same category and related users form a subgraph. Subgraphs are combined if one client holds more than one category. In knowledge graph, the pre-processing includes two steps. We first build subgraphs by relation types or node community, and then partition the data to multiple clients by a specific manner (uniform or non-I.I.D. partitioning with LDA).

Citation/Coauthor Datasets The main data pre-processing step for citation/coauthor networks in the node-level setting is to sample central nodes (egos) and build the k-hop neighborhoods correspondingly (k-hop ego-networks). For each dataset with a big entire graph, we randomly sample a number of egos (e.g. 1000) and construct k (e.g. 2)-hop egonetworks for them. These ego-networks are then partitioned by a specific manner (e.g. LDA non-IID partition) and distributed to multiple clients.

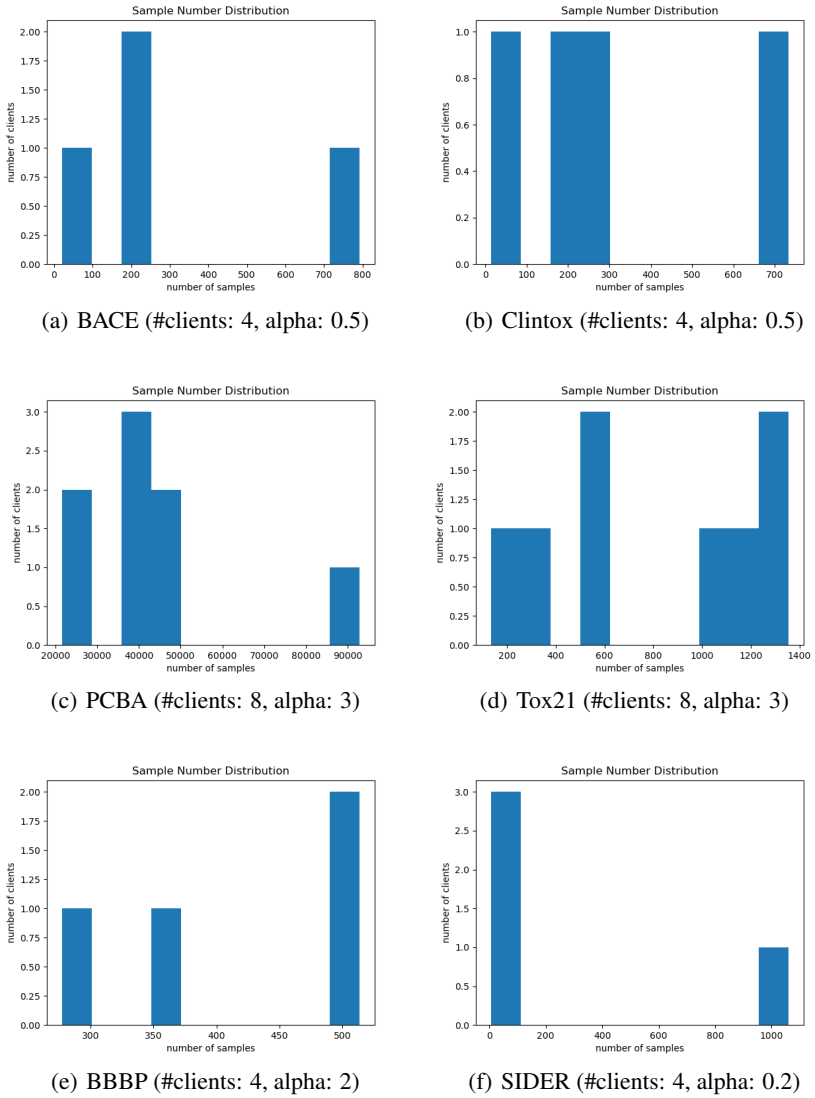
B.3 Non-I.I.D. Partitioning

Latent Dirichlet Allocation(LDA)-Based According to [104, 82], we generate a heterogeneous partition into J clients by sampling $p_k \sim \mathrm{Dir}_J(\alpha)$ and allocating a $p_{k,j}$ proportion of the training instances of class k to local client.



(a) $\alpha = 0.1$ (b) $\alpha = 10.0$

Figure 5: Unbalanced sample distribution (non-I.I.D.) for citation networks.

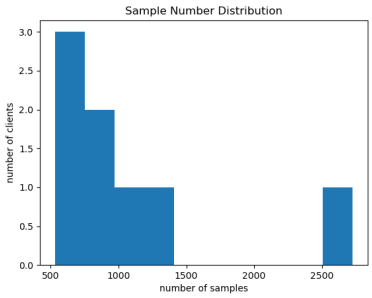


(a) BACE (#clients: 4, alpha: 0.5) (b) Clintox (#clients: 4, alpha: 0.5)

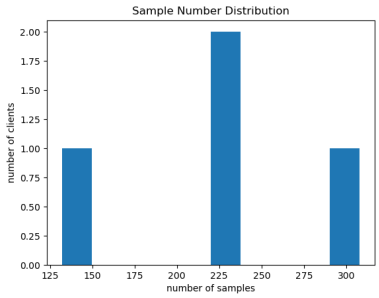
(c) PCBA (#clients: 8, alpha: 3) (d) Tox21 (#clients: 8, alpha: 3)

(e) BBBP (#clients: 4, alpha: 2) (f) SIDER (#clients: 4, alpha: 0.2)

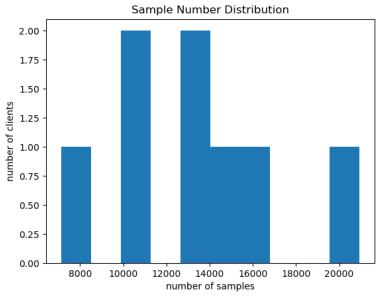
Figure 6: Unbalanced Sample Distribution (Non-I.I.D.) for Molecular Graph Classification Datasets



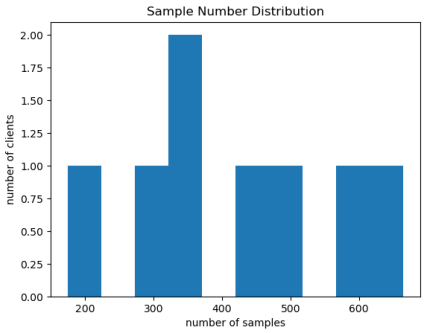
(a) hERG (#clients: 4, alpha: 3)



(b) ESOL (#clients: 4, alpha: 2)



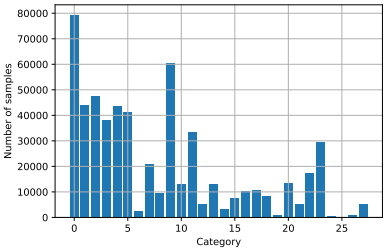
(c) QM9 (#clients: 8, alpha: 3)



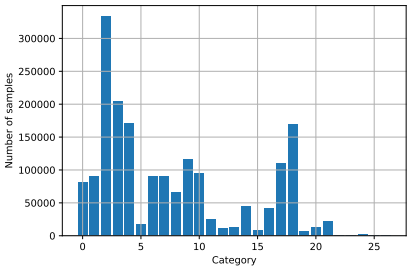
(d) LIPO (#clients: 8, alpha: 2)

Figure 7: Example (Non-I.I.D.) Sample Distributions for Molecular Property Prediction Datasets

Based on the meta-data of dataset Data can be partitioned to clients by meta-data information. Meta-data can show the intrinsic data partition in real life scenes. For example, in recommendation system, user's behaviour is different for items from different categories. Fig. 8 shows the non-I.I.D. of user's rating number on item from different categories.



(a) Ciao



(b) Epinions

Figure 8: Example (Non-I.I.D.) Sample Distributions for Recommendation Systems Datasets

C More Details of FedGraphNN System Design

Data Collector and Manager is a distributed computing system that can collect scattered datasets or features from multiple servers to Training Manager. Such collection can also keep the raw data in the original server with RPCs, which can only access the data during training. After obtaining all necessary datasets for federated training, Training Manager will start federated training using algorithms supported by FedML-API. Once training has been completed, Model Serving can request the trained model to deploy for inference. Under this SDK abstraction, we plan to address the challenges mentioned above (1) and (2) within the Data Collector and Manager. As for challenge (3), we plan to make FedML Client SDK compatible with any operating systems (Linux, Android, iOS) with a cross-platform abstraction interface design. In essence, the three modules inside FedML Client SDK builds up a pipeline that manages a model's life cycle, from federated training to personalized model serving (inference). Unifying three modules of a pipeline into a single SDK can simplify the system design. Any subsystem in an institute can integrate FedML Client SDK with a host process, which can be the backend service or desktop application. Overall, we hope FedML Client SDK could be a lightweight and easy-to-use SDK for federated learning among diverse cross-silo institutes.

D More Results of System Efficiency and Security

D.1 Evaluation on System Efficiency

Table 6: System-Level Performance Metrics for Graph-Level FedGraphNN tasks with FedAvg (Hardware: 8 x NVIDIA Quadro RTX 5000 GPU (16GB/GPU); RAM: 512G; CPU: Intel Xeon Gold 5220R 2.20GHz).

		SIDER	**BACE**	**Clintox**	**BBBP**	**Tox21**	**FreeSolv**	**ESOL**	**Lipo**	**hERG**	**QM9**
Wall-clock Time	GCN	5m 58s	4m 57s	4m 40s	4m 13s	15m 3s	4m 12s	5m 25s	16m 14s	35m 30s	6h 48m
	GAT	8m 48s	5m 27s	7m 37s	5m 28s	25m 49s	6m 24s	8m 36s	25m 28s	58m 14s	9h 21m
	GraphSAGE	2m 7s	3m 58s	4m 42s	3m 26s	14m 31s	5m 53s	6m 54s	15m 28s	32m 57s	5h 33m
Average FLOP	GCN	697.3K	605.1K	466.2K	427.2K	345.8K	142.6K	231.6K	480.6K	516.6K	153.9K
	GAT	703.4K	612.1K	470.2K	431K	347.8K	142.5K	232.6K	485K	521.3K	154.3K
	GraphSAGE	846K	758.6K	1.1M	980K	760.6K	326.9K	531.1K	1.5M	1.184M	338.2K
Parameters	GCN	15.1K	13.5K	13.6K	13.5K	14.2K	13.5K	13.5K	13.5K	13.5K	14.2K
	GAT	20.2K	18.5K	18.6K	18.5K	19.2K	18.5K	18.5K	18.5K	18.5K	19.2K
	GraphSAGE	10.6K	8.9K	18.2K	18.1K	18.8K	18.1K	18.1K	269K	18.1K	18.8K

*Note that we use the distributed training paradigm where each client's local training uses one GPU. Please refer to our code for details.

Table 7: System-level Performance Metrics for Subgraph-Level FedGraphNN tasks with FedAvg (Hardware: 8 x NVIDIA Quadro RTX 5000 GPU (16GB/GPU); RAM: 512G; CPU: Intel Xeon Gold 5220R 2.20GHz).

		Ciao	**Epinions**
Wall-clock Time	GCN	352s	717s
	GAT	350s	749s
	GraphSAGE	551s	810s
Average FLOP	GCN	697.3K	605.1K
	GAT	703.4K	612.1K
	GraphSAGE	846K	758.6K
Parameters	GCN	15.1K	13.5K
	GAT	20.2K	18.5K
	GraphSAGE	10.6K	8.9K

*Note that we use the distributed training paradigm where each client's local training uses one GPU. Please refer to our code for details.

The training time using RPC is also evaluated; and results are similar to that of using MPI. Note that RPC is useful for realistic deployment when GPU/CPU-based edge devices can only be accessed via public IP addresses due to locating in different data centers. We will provide detailed test results in such a scenario in our future work.

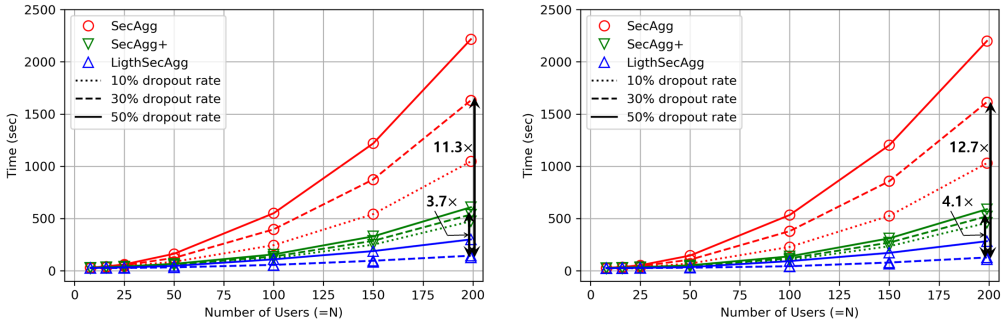
Table 8: System-level Performance Metrics for Node-Level `FedGraphNN` tasks with FedAvg (Hardware: 8 x NVIDIA Quadro RTX 5000 GPU (16GB/GPU); RAM: 512G; CPU: Intel Xeon Gold 5220R 2.20GHz).

		CORA	Citeseer	DBLP	PubMed
Wall-clock Time	GCN	833s	622s	654s	653s
	GAT	871s	652s	682s	712s
	GraphSAGE	774s	562s	622s	592s
Average FLOP	GCN	44.9M	739.4K	8.8M	1.9M
	GAT	47.8M	845.3K	9.5M	2.3M
	GraphSAGE	45.3M	817K	9.2M	2.1M
Parameters	GCN	282.1K	20.5K	53.7K	17.2K
	GAT	285.1K	23.7K	56.5K	19.3K
	GraphSAGE	283K	21.6K	54.7K	18.2K

*Note that we use the distributed training paradigm where each client's local training uses one GPU. Please refer to our code for details.

D.2 Evaluation on Security (`LightSecAgg`)

`LightSecAgg`, a FL security algorithm developed by our team [2], provides model privacy guarantees as the state-of-the-art (SecAgg [7] and SecAgg+ [3]) while substantially reducing the aggregation (hence run-time) complexity (Figure 9). The main idea of `LightSecAgg` is that each user protects its local model using a locally generated random mask. This mask is then encoded and shared to other users, in such a way that the aggregate mask of any sufficiently large set of surviving users can be directly reconstructed at the server. Our main effort in `FedGraphNN` is integrating `LightSecAgg`, optimizing its system performance, and designing user-friendly APIs to make it compatible with various models and FL algorithms.



(a) Non-overlapped (b) Overlapped

Figure 9: `LightSecAgg`: Total running time of `LightSecAgg` versus the state-of-the-art protocols (SecAgg [7] and SecAgg+ [3]) to train neural networks with 1.2 M parameters (*more than all GNN models used in this work*), as the number of users increases, for various dropout rates.

[2] `LightSecAgg` is under submission and patent application process when we publish `FedGraphNN`, so we cannot provide a full paper reference now. At the current stage, to understand secure aggregation, we refer readers to the baseline SecAgg [7].

E More Details of the Empirical Analysis

E.1 Hyper-parameters

For each task, we utilize grid search to find the best results. Table 9 & **??** list all the hyper-parameters ranges used in our experiments. All hyper-parameter tuning is run on a single GPU. The best hyperparameters for each dataset and model are listed in Table 13,14,15, & 16 For molecule tasks ,batch-size is kept fixed since the molecule-level task requires us to have mini-batch is equal to 1. Also, number of GNN layers were fixed to 2 because having too many GNN layers result in over-smoothing phenomenon as shown in [45]. For all experiments, we used Adam optimizer.

Table 9: Hyper-parameter Range for Graph-Level Centralized Training(classification & regression)

hyper-parameter	Description	Range
learning rate	Rate of speed at which the model learns.	$[0.00015, 0.0015, 0.015, 0.15]$
dropout rate	Dropout ratio	$[0.2, 0.3, 0.5, 0.6]$
node embedding dimension	Dimensionality of the node embedding	$[16, 32, 64, 128, 256]$
hidden layer dimension	Hidden layer dimensionality	$[16, 32, 64, 128, 256]$
readout embedding dimension	Dimensionality of the readout embedding	$[16, 32, 64, 128256]$
graph embedding dimension	Dimensionality of the graph embedding	$[16, 32, 64, 128, 256]$
attention heads	Number of attention heads required for GAT	1-7
alpha	LeakyRELU parameter used in GAT model	0.2

Table 10: Hyper-parameter Range for Graph-Level Federated Learning(classification & regression)

hyper-parameter	Description	Range
learning rate	Rate of speed at which the model learns.	$[0.00015, 0.0015, 0.015, 0.15]$
dropout rate	Dropout ratio	$[0.3, 0.5, 0.6]$
node embedding dimension	Dimensionality of the node embedding	64
hidden layer dimension	Hidden layer dimensionality	64
readout embedding dimension	Dimensionality of the readout embedding	64
graph embedding dimension	Dimensionality of the graph embedding	64
attention heads	Number of attention heads required for GAT	1-7
alpha	LeakyRELU parameter used in GAT model	0.2
rounds	Number of federating learning rounds	[10, 50, 100]
epoch	Epoch of clients	1
number of clients	Number of users in a federated learning round	4-10

Table 11: Hyper-parameter Range for Subgraph-Level Federated Learning

hyper-parameter	Description	Range
learning rate	Rate of speed at which the model learns.	$[0.0001, 0.001, 0.01, 0.1]$
node embedding dimension	Dimensionality of the node embedding	64
hidden layer dimension	Hidden layer dimensionality	$[64]$
rounds	Number of federating learning rounds	$[1, 10, 20, 50, 100]$
local epoch	Epoch of clients	$[1, 2, 5]$
number of clients	Number of users in a federated learning round	4-10

Table 12: Hyper-parameter Range for Node-Level Federated Learning

hyper-parameter	Description	Range
learning rate	Rate of speed at which the model learns.	$[0.1, 0.01, 0.001, 0.0001]$
dropout rate	Dropout ratio	$[0.3, 0.5, 0.6]$
hidden layer dimension	Hidden layer dimensionality	[32, 64, 128]
alpha	LeakyRELU parameter used in GAT model	[0.1, 10]
rounds	Number of federating learning rounds	100
epoch	Epoch of clients	[1, 3, 5]
number of clients	Number of users in a federated learning round	10

Table 13: Hyperparameters for Graph-Level Molecular Classification Task

Dataset	**Score & Parameters**	**GCN**	**GAT**	**GraphSAGE**
BBBP	ROC-AUC Score	0.8705	0.8824	**0.8930**
	learning rate	0.0015	0.015	0.01
	dropout rate	0.2	0.5	0.2
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
BACE	ROC-AUC Score	0.9221	0.7657	**0.9266**
	learning rate	0.0015	0.001	0.0015
	dropout rate	0.3	0.3	0.3
	node embedding dimension	64	64	16
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
Tox21	ROC-AUC Score	0.7800	0.8144	**0.8317**
	learning rate	0.0015	0.00015	0.00015
	dropout rate	0.4	0.3	0.3
	node embedding dimension	64	128	256
	hidden layer dimension	64	64	128
	readout embedding dimension	64	128	256
	graph embedding dimension	64	64	128
	attention heads	None	2	None
	alpha	None	0.2	None
SIDER	ROC-AUC Score	0.6476	0.6639	**0.6669**
	learning rate	0.0015	0.0015	0.0015
	dropout rate	0.3	0.3	0.6
	node embedding dimension	64	64	16
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
ClinTox	ROC-AUC Score	0.8914	0.9573	**0.9716**
	learning rate	0.0015	0.0015	0.0015
	dropout rate	0.3	0.3	0.3
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None

Table 14: Hyperparameters for Graph-Level Federated Molecular Classification Task

Dataset	Score & Parameters	GCN + FedAvg	GAT + FedAvg	GraphSAGE + FedAvg
BBBP	ROC-AUC Score	0.7629	0.8746	**0.8935**
	number of clients	4	4	4
	learning rate	0.0015	0.0015	0.015
	dropout rate	0.3	0.3	0.6
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
BACE	ROC-AUC Score	0.6594	0.7714	**0.8604**
	number of clients	4	4	4
	learning rate	0.0015	0.0015	0.0015
	dropout rate	0.5	0.3	0.5
	node embedding dimension	64	64	16
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
Tox21	ROC-AUC Score	0.7128	0.7171	**0.7801**
	number of clients	4	4	4
	learning rate	0.0015	0.0015	0.00015
	dropout rate	0.6	0.3	0.3
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
SIDER	ROC-AUC Score	0.6266	0.6591	**0.67**
	number of clients	4	4	4
	learning rate	0.0015	0.0015	0.0015
	dropout rate	0.6	0.3	0.6
	node embedding dimension	64	64	16
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
ClinTox	ROC-AUC Score	0.8784	0.9160	**0.9246**
	number of clients	4	4	4
	learning rate	0.0015	0.0015	0.015
	dropout rate	0.5	0.6	0.3
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None

Table 15: Hyperparameters for Graph-Level Molecular Regression Task

Dataset	Score &Parameters	GCN	GAT	GraphSAGE
Freesolv	RMSE Score	0.8705	0.8824	**0.8930**
	learning rate	0.0015	0.015	0.01
	dropout rate	0.2	0.5	0.2
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
ESOL	RMSE Score	0.8705	0.8824	**0.8930**
	learning rate	0.0015	0.015	0.01
	dropout rate	0.2	0.5	0.2
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
Lipophilicity	RMSE Score	0.8521	0.7415	**0.7078**
	learning rate	0.0015	0.001	0.001
	dropout rate	0.3	0.3	0.3
	node embedding dimension	128	128	128
	hidden layer dimension	64	64	64
	readout embedding dimension	128	128	128
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
hERG	RMSE Score	0.7257	**0.6271**	0.7132
	learning rate	0.001	0.001	0.005
	dropout rate	0.3	0.5	0.3
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
QM9	RMSE Score	14.78	**12.44**	13.06
	learning rate	0.0015	0.015	0.01
	dropout rate	0.2	0.5	0.2
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None

Table 16: Hyperparameters for Graph-Level Federated Molecular Regression Task

Dataset	Parameters	GCN + FedAvg	GAT + FedAvg	GraphSAGE + FedAvg
FreeSolv	RMSE Score	2.747	3.108	**1.641**
	number of clients	4	8	4
	learning rate	0.0015	0.00015	0.015
	dropout rate	0.6	0.5	0.6
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
ESOL	RMSE Score	1.435	**1.028**	1.185
	number of clients	4	4	4
	learning rate	0.0015	0.0015	0.0015
	dropout rate	0.5	0.3	0.3
	node embedding dimension	64	256	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
Lipophilicity	RMSE Score	1.146	1.004	**0.7788**
	number of clients	4	4	4
	learning rate	0.0015	0.0015	0.0015
	dropout rate	0.3	0.3	0.3
	node embedding dimension	64	64	256
	hidden layer dimension	64	64	256
	readout embedding dimension	64	64	256
	graph embedding dimension	64	64	256
	attention heads	None	2	None
	alpha	None	0.2	None
hERG	RMSE Score	0.7944	0.7322	**0.7265**
	number of clients	8	8	8
	learning rate	0.0015	0.0015	0.0015
	dropout rate	0.3	0.3	0.6
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64
	readout embedding dimension	64	64	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None
QM9	MAE Score	21.075	23.173	**19.167**
	number of clients	8	8	8
	learning rate	0.0015	0.00015	0.15
	dropout rate	0.2	0.5	0.3
	node embedding dimension	64	256	64
	hidden layer dimension	64	128	64
	readout embedding dimension	64	256	64
	graph embedding dimension	64	64	64
	attention heads	None	2	None
	alpha	None	0.2	None

Table 17: Hyperparameters for Subgraph-Level Centralized Link Prediction Task

Dataset	Score & Parameters	GCN	GAT	GraphSAGE
Ciao	mean absolute error	**0.8167**	0.8214	0.8231
	mean squared error	**1.1184**	1.1318	1.1541
	root mean squared error	**1.0575**	1.0639	1.0742
	communication round	100	100	50
	learning rate	0.01	0.001	0.01
	node embedding dimension	64	64	64
	hidden layer dimension	32	32	16
Epinions	mean absolute error	**0.8847**	0.8934	1.0436
	mean squared error	**1.3733**	1.3873	1.8454
	root mean squared error	**1.1718**	1.1767	1.3554
	communication round	50	50	100
	learning rate	0.01	0.01	0.01
	node embedding dimension	64	64	64
	hidden layer dimension	32	32	64

Table 18: Hyperparameters for Subgraph-Level Federated Link Prediction Task

Dataset	Score & Parameters	GCN + FedAvg	GAT + FedAvg	GraphSAGE + FedAvg
Ciao	mean absolute error	0.7995	**0.7987**	0.8290
	mean squared error	**1.0667**	1.0682	1.1320
	root mean squared error	**1.0293**	1.0311	1.0626
	communication round	100	100	50
	local epochs	5	2	5
	number of clients	8	8	8
	learning rate	0.01	0.001	0.01
	node embedding dimension	64	64	64
	hidden layer dimension	32	32	32
Epinions	mean absolute error	0.9033	**0.9032**	0.9816
	mean squared error	1.4378	**1.4248**	1.6136
	root mean squared error	1.1924	**1.1882**	1.2625
	communication round	100	50	100
	local epochs	2	1	2
	number of clients	8	8	8
	learning rate	0.01	0.001	0.01
	node embedding dimension	64	64	64
	hidden layer dimension	64	64	64

Table 19: Hyperparameters for Node-Level Federated Node Classification Task

Dataset	Parameters	GCN + FedAvg	GAT + FedAvg	GraphSAGE + FedAvg
CORA	Micro F1 score	0.8549	diverge	**0.9746**
	number of clients	10	10	10
	learning rate	0.001	0.001	0.001
	dropout rate	0.5	0.5	0.5
	local epochs	5	5	5
	hidden layer dimension	128	128	128
Citeseer	Micro F1 score	0.9743	0.9610	**0.9854**
	learning rate	0.001	0.01	0.001
	dropout rate	0.5	0.5	0.5
	local epochs	5	3	5
	hidden layer dimension	128	128	128
PubMed	Micro F1 score	0.9191	0.8557	**0.9761**
	number of clients	10	10	10
	learning rate	0.001	0.001	0.001
	dropout rate	0.5	0.5	0.5
	local epochs	5	5	5
	hidden layer dimension	128	128	128
DBLP	Micro F1 score	0.9088	0.8201	**0.9749**
	number of clients	10	10	10
	learning rate	0.001	0.001	0.001
	dropout rate	0.5	0.5	0.5
	local epochs	5	5	5
	hidden layer dimension	128	128	128

E.2 Evaluation Metrics

Current metrics supported in `FedGraphNN` include:

- **Graph Classification**: ROC-AUC (Area Under the Curve - Receiver Operating Characteristics) is a well-used classification metric to evaluate the performance at various thresholds.
- **Graph Regression**: For this task, we chose RMSE (Root Mean Squared Error). However, when train and test distributions differ, it is more suitable to use metrics such as MAPE (Mean Absolute Percentage Error).
- **Node Classification**: In ego-network node-level FL task, we use F1 score as the metric because F1 score is better than the accuracy metric when imbalanced class distribution exists.
- **Link prediction (Recommendation Systems)**: Specifically for recommendation systems, we can treat it as a regression problem. Thus, it is possible to use well-known metrics such as MAE (Mean Absolute Error), MSE (Mean Squared Error), RMSE (Root Mean Squared Error). Widely used ranking based metrics can also be applied such as DCG (Discounted Cumulative Gain) and NDCG (Normalized Discounted Cumulative Gain),.
- **Relation Prediction (Knowledge Graphs)**: Besides accuracy based metrics such as precision, recall and f1 score, ranking based metrics are also applied to relation type prediction on knowledge graphs such as MRR (Mean Reciprocal Rank) [60] and HR (Hit Ratio).

In addition to the metrics described, `FedGraphNN` users can add their custom metrics as well. As a future work, we plan to analyze these metrics' representative capacity on the FL performance.

E.3 More Experimental Results

Aside from additional results for graph-level FL, we will further provide more live results and reports in our project website `https://FedML.ai/FedGraphNN`. We hope these visualized training results can be a useful reference for future research exploration.

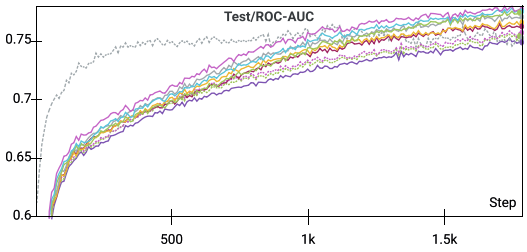


Figure 10: Tox21: test score during sweeping

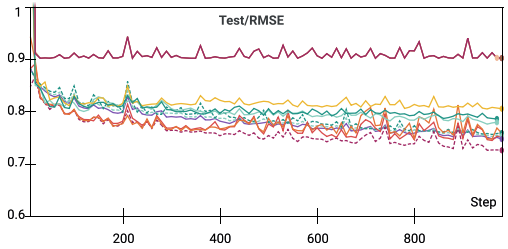


Figure 11: hERG: test score during sweeping

Table 20: Performance of graph regression in the graph-level FL setting (#clients=4, MAE for QM9).

Metric	RMSE				
Method	FreeSolv $\alpha = 0.2$	ESOL $\alpha = 0.5$	Lipo $\alpha = 0.5$	hERG $\alpha = 3$	QM9 $\alpha = 3$
MoleculeNet Results	1.40±0.16	0.97±0.01	0.655±0.036	DNE	2.35
GCN (centralized)	1.5787	1.0190	0.8518	0.7257	14.78
GCN (FedAvg)	2.7470	1.4350	1.1460	0.7944	21.075
GAT (Centralized)	1.2175	0.9358	0.7465	0.6271	12.44
GAT (FedAvg)	1.3130	0.9643	0.8537	0.7322	23.173
GraphSAGE (centralized)	1.3630	0.8890	0.7078	0.7132	13.06
GraphSAGE (FedAvg)	1.6410	1.1860	0.7788	0.7265	19.167