[Supplementary File]
Personalized Subgraph Federated Learning

Organization This supplementary file is organized as follows. In Section A, we first describe the algorithms of our FEDerated Personalized sUBgraph learning (FED-PUB) framework, on both server- and client-sides. Then, we provide the detailed experimental setups including datasets, models and hyperparameters in Section B, as well as the additional experimental results in Section C. After that, we finally discuss the limitations and potential societal impacts of our work in Section D.

A Algorithms

In this section, we algorithmically describe where the proposed subgraph similarity estimation and adaptive weight masking are performed in our FED-PUB framework, which is shown in Algorithm 1 for the client and in Algorithm 2 for the server.

Algorithm 1 FED-PUB Client Algorithm

R: number of rounds, E: number of epochs, K: number of clients, $\mathcal{D}_k$: local data for client k, f_k: model function for client k, $\boldsymbol{\theta}_k$: model parameters for client k, $\boldsymbol{\mu}_k$: weight masking parameters for client k, $\mathcal{S}(\cdot)$: similarity matching function, τ: scaling factor for similarity matching.

Function RunClient($\bar{\boldsymbol{\theta}}_k$)
$\boldsymbol{\theta}_k \leftarrow \bar{\boldsymbol{\theta}}_k \otimes \boldsymbol{\mu}_k$
for each local epoch e from 1 to E **do**
  $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k - \eta \nabla \mathcal{L}(\mathcal{D}_k; \boldsymbol{\theta}_k, \boldsymbol{\mu}_k)$
end for
return $\boldsymbol{\theta}_k$

Algorithm 2 FED-PUB Server Algorithm

Function RunServer()
initialize $\bar{\boldsymbol{\theta}}^{(0)}$
for each round $r = 1, 2, \ldots, R$ **do**
  for $f_k \forall k$ **in parallel do**
    if $r = 1$ **then**
      $\boldsymbol{\theta}_k^{(r+1)} \leftarrow$ RunClient($\bar{\boldsymbol{\theta}}^{(r)}$)
    else
      $\bar{\boldsymbol{\theta}}_k^{(r)} \leftarrow \sum_{i=0}^{K} \frac{\exp(\tau \cdot S(k,i))}{\sum_{j=0}^{K} \exp(\tau \cdot S(k,j))} \boldsymbol{\theta}_i$
      $\boldsymbol{\theta}_k^{(r+1)} \leftarrow$ RunClient($\bar{\boldsymbol{\theta}}_k^{(r)}$)
    end if
  end for
end for

B Experimental Setups

In this section, we first provide the descriptions of six different benchmark datasets that we use, with their preprocessing setups and statistics in Subsection B.1. Then, we introduce the baselines and our proposed FED-PUB in detail in Subsection B.2. After that, we further describe the implementation details of the experiments on synthetic and real-world graphs in Subsection B.3.

B.1 Datasets

We report the statistics of six different benchmark datasets (i.e., Cora, CiteSeer, Pubmed, and ogbn-arxiv for citation graphs; Computer and Photo for amazon product graphs) [15, 6, 11, 16] that we use in our experiments for both the overlapping and non-overlapping node scenarios, in Table 1. Specifically, we report the number of nodes, edges, classes, and clustering coefficient for each subgraph, as well as the heterogeneity among all the subgraphs. Note that clustering coefficients, which are the measure of how much nodes tend to cluster together, are calculated by the average of clustering coefficients [17] of all nodes. On the other hand, we calculate the heterogeneity by measuring the median Jenson-Shannon divergence of label distributions between all pairs of subgraphs. For dataset splits, we randomly select 20% nodes for training, 35% for validation, and 35% for testing, for all datasets except for the arxiv dataset. As the arxiv dataset has relatively larger numbers of nodes as shown in Table 1, we randomly select 5% nodes for training, the remaining half of the nodes for validation, and the other nodes for testing.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Table 1: **Statistics of datasets**, where we report the number of nodes, edges, classes, clustering coefficient, and heterogeneity for both the original graph and its splitted subgraphs on overlapping and non-overlapping node scenarios. Ori denotes the original graph, and Cli denotes the number of clients.

Overlapping node scenario

	Cora				**CiteSeer**				**Pubmed**			
	Ori	**10 Cli**	**30 Cli**	**50 Cli**	**Ori**	**10 Cli**	**30 Cli**	**50 Cli**	**Ori**	**10 Cli**	**30 Cli**	**50 Cli**
# Classes	7				6				3			
# Nodes	2,485	621	207	124	2,120	530	177	106	19,717	4,929	1,643	986
# Edges	10,138	1,249	379	215	7,358	889	293	170	88,648	10,675	3,374	1,903
Clustering Coefficient	0.238	0.133	0.129	0.125	0.170	0.088	0.087	0.096	0.060	0.035	0.034	0.035
Heterogeneity	-	0.297	0.567	0.613	-	0.278	0.494	0.547	-	0.210	0.383	0.394
	ogbn-arxiv				**Amazon-Computer**				**Amazon-Photo**			
	Ori	**10 Cli**	**30 Cli**	**50 Cli**	**Ori**	**10 Cli**	**30 Cli**	**50 Cli**	**Ori**	**10 Cli**	**30 Cli**	**50 Cli**
# Classes	40				10				8			
# Nodes	169,343	42,336	14,112	8,467	13,381	3,345	1,115	669	7,487	1,872	624	374
# Edges	2,315,598	282,083	83,770	44,712	491,556	59,236	16,684	8,969	238,086	29,223	8,735	4,840
Clustering Coefficient	0.351	0.337	0.348	0.359	0.410	0.380	0.391	0.410	0.226	0.177	0.185	0.191
Heterogeneity	-	0.327	0.577	0.614	-	0.306	0.696	0.684	-	0.315	0.606	0.615

Non-overlapping node scenario

	Cora				**CiteSeer**				**Pubmed**			
	Ori	**5 Cli**	**10 Cli**	**20 Cli**	**Ori**	**5 Cli**	**10 Cli**	**20 Cli**	**Ori**	**5 Cli**	**10 Cli**	**20 Cli**
# Classes	7				6				3			
# Nodes	2,485	497	249	124	2,120	424	212	106	19,717	3,943	1,972	986
# Edges	10,138	1,866	891	422	7,358	1,410	675	326	88,648	16,374	7,671	3,607
Clustering Coefficient	0.238	0.250	0.259	0.263	0.170	0.175	0.178	0.180	0.060	0.063	0.066	0.067
Heterogeneity	-	0.590	0.606	0.665	-	0.517	0.541	0.568	-	0.362	0.392	0.424
	ogbn-arxiv				**Amazon-Computer**				**Amazon-Photo**			
	Ori	**5 Cli**	**10 Cli**	**20 Cli**	**Ori**	**5 Cli**	**10 Cli**	**20 Cli**	**Ori**	**5 Cli**	**10 Cli**	**20 Cli**
# Classes	40				10				8			
# Nodes	169,343	33,869	16,934	8,467	13,381	2,676	1,338	669	7,487	1,497	749	374
# Edges	2,315,598	410,948	182,226	86,755	491,556	84,480	36,136	15,632	238,086	43,138	19,322	8,547
Clustering Coefficient	0.351	0.385	0.398	0.418	0.410	0.437	0.457	0.477	0.226	0.247	0.259	0.269
Heterogeneity	-	0.604	0.612	0.647	-	0.684	0.681	0.751	-	0.593	0.615	0.637

We then describe how to partition the original graph into multiple subgraphs, whose counts are the same as the number of clients. In general, we use the METIS graph partitioning algorithm [7] to divide the original graph into multiple subgraphs, which can control the number of disjoint subgraphs as parameters. Thus, in the non-overlapping node scenario, the disjoint subgraphs for each client are directly obtained by the output of the METIS algorithm (i.e., if we set the parameter value for METIS as 10, then we can obtain the 10 different disjoint subgraphs, each of which is given to each client). On the other hand, in the overlapping node scenario where nodes are duplicated across different subgraphs, we first divide the original graph into 2, 6, and 10 disjoint subgraphs for 10 clients, 30 clients, and 50 clients, respectively, with the METIS algorithm. After that, in the one disjoint subgraph, we randomly sample half of the nodes and their associated edges, and then use them as the subgraph for one particular client. This procedure is performed five times to generate five different yet overlapped subgraphs for five clients, per one split subgraph obtained from METIS.

B.2 Baselines and Our Model

1. **FedAvg**: This method [12] is the FL baseline, where each client locally updates a model and then sends it to a server, while the server aggregates the locally updated models with respect to their number of training samples and then transmits the aggregated one back to the clients.
2. **FedProx**: This method [10] is the FL baseline, which regularizes the local model to not drift too much to the local data by minimizing the weight differences between local and global models.
3. **FedPer**: This method [1] is the personalized FL baseline, which shares only the base layers, while keeping the personalized classification layers in the local side.
4. **FedGNN**: This method [18] is the subgraph FL baseline, which expands the local subgraph by exactly augmenting the relevant nodes from the other clients. The relevant nodes are selected based on the similarity between the nodes in the local client and the other nodes in other clients.
5. **FedSage+**: This method [20] is the subgraph FL baseline, which expands the local subgraph by estimating the nodes from the local graph generator that is trained with the information of nodes in the other clients. Specifically, to train the graph generator, it first transmits the local node

representations to other clients, and then calculates the gradient of the distance between received node representations and local node representations, which are transmitted back to the local client.

6. **GCFL**: This method [19] is the graph FL baseline, which targets completely disjoint graphs (e.g., molecular graphs) as in image tasks. In particular, this method uses the bi-partitioning scheme, which divides a set of clients into two disjoint client groups based on their similarity of gradients. Then, the model weights are only shared between grouped clients having similar gradients, after partitioning. Note that this bi-partitioning scheme is similar to the scheme proposed in clustered-FL [14] for image classification, and we adopt this for our subgraph FL.
7. **Local**: This method is the non-FL baseline, which only locally trains the model for each client, and does not share weights between clients.
8. **FED-PUB**: This is our FEDerated Personalized sUBgraph learning (FED-PUB) framework, which not only estimates the similarity between client subgraphs with their models' functional embeddings for detecting subgraph community structures, but also adaptively masks received weights from the server for selecting only the subgraph-relevant parameters.

B.3 Implementation Details

Common Implementation Details For all experiments, we stack two layers of Graph Convolutional Network (GCN) [9] and one linear classifier layer. Also, the number of hidden dimensions is set to 128, the learning rate is set to 0.001, and all clients participate in federated training for every round. Then, all models are optimized with Adam optimizer [8]. To obtain functional embeddings in our FED-PER framework, we randomly generate a community graph from a stochastic block model [5]. Specifically, we sample five different graphs each of which has 100 nodes, where the probability of edges within the single graph is 0.1, while the probability of edges between the different graphs is 0.0. The node features for the sampled community graph are randomly initialized from the normal distribution. Note that the randomly sampled graphs are initialized at the server-side and the server distributes such graphs to all clients, while the client calculates its model's functional embedding and then transmits the obtained embedding to the server. Furthermore, for all experiments about our FED-PUB, we set the λ_1 and λ_2 for L_1 and L_2 losses for sparsity and proximal terms as 0.001. Notably, while we can tune such two scaling hyperparameters, we observe that those two values show satisfactory performances across all datasets without further specific tuning to each dataset.

Implementation Details on Synthetic Graphs We perform two experiments on synthetic graphs, which are shown in Figure 1 and Figure 3 of the main paper. In particular, in the experiment of Figure 1, there are three communities that have different label distributions (e.g., the nodes in the first community have label 0, whereas the nodes in the last community have label 2), and each community has 5/5/40 non-overlapped subgraphs. Also, there are 50 clients, and each client has one of 50 subgraphs. Each subgraph consists of 30 nodes, and the edges between two nodes are sampled from the probability of 0.5. Also, in the experiment of Figure 3, there are two communities that have different label distributions, and each community has 5/15 non-overlapped subgraphs. Also, there are 20 clients, and each client has one of 20 subgraphs. Each subgraph consists of 30 nodes, and the edges between two subgraphs within the same community are sampled from the probability of 0.7, whereas the edges between two subgraphs from different communities are sampled from the probability of 0.01. For all experiments, the number of local epochs is set to 3, and the number of total rounds is set to 100. In our FED-PER including its variants of using parameter and gradient, the scaling hyperparameter for calculating the similarity in equation 4 (i.e., τ) is set to 10.

Implementation Details on Real-World Benchmark Graphs For small datasets, namely Cora, CiteSeer and PubMed, we set the number of local training epoch as 1, and the number of total rounds as 100. Also, for larger datasets, such as Computer, Photo and arxiv, we set the number of total rounds as 200, while the number of local epochs is set to 2 for Photo and arxiv, and set to 3 for Computer. In the overlapping node scenario, we set the similarity scaling hyperparameter (i.e., τ) as 5 for all our models. Meanwhile, we set the similarity scaling hyperparameter (i.e., τ) as 3 in the non-overlapping node scenario for all our models. We generally observe that, the larger τ value works better for the overlapping node scenario, in which different subgraphs are easily grouped together, compared to the non-overlapping node scenario. Finally, we report the test performance of all models at the best validation epoch, and the performance is measured by the node classification accuracy.

λ_1	λ_2	Accuracy [%]	Sparsity [%]
3e-1	1e-3	79.62 ± 0.23	28.93 ± 0.52
5e-1	1e-3	79.42 ± 0.37	42.38 ± 0.35
7e-1	1e-3	78.68 ± 0.59	56.94 ± 0.29
9e-1	1e-3	77.36 ± 0.99	74.87 ± 0.34

λ_1	λ_2	Accuracy [%]	Sparsity [%]
7e-1	1e-3	78.68 ± 0.59	56.94 ± 0.29
7e-1	1e-2	78.56 ± 0.05	56.61 ± 0.32
7e-1	1e-1	79.46 ± 0.41	57.41 ± 1.33
7e-1	1e-0	79.31 ± 0.45	57.28 ± 0.16

Figure 1: **Analysis on hyperparameters λ_1 and λ_2**, with corresponding model sparsity and performance.

Computing Resources For all experiments, we use PyTorch [13] and PyTorch Geometric [4] as deep learning libraries. We use two types of GPUs: GeForce RTX 2080 Ti and TITAN XP for training each model. The runtime of our framework depends on the number of clients, and also workers for processing clients' jobs, as well as the number of local epochs and rounds. In general, we use 10 or 20 workers (i.e., simultaneously training 10/20 local models for 10/20 clients), and the single run of our algorithm for 50 clients with 1 local epoch and 100 total rounds takes less than 2 hours.

C Additional Experimental Results

In this section, we provide additional experimental results on varying hyperparameters, varying graph partitioning schemes, varying random graph inputs, and varying similarity calculation schemes, as well as analyses on the distributional shifts, the relations between graph size and heterogeneity, and the impact of missing edges to task performances.

C.1 Results on Varying λ_1 and λ_2 Values

In Figure 1, we further explore the effects of hyperparameters λ_1 and λ_2 on the Cora dataset with the overlapping node scenario, where the number of local epochs is set as 2 and the number of clients is set as 10. In particular, λ_1 value can control the degree of the model sparsity, thus, to see its efficacy, we fix λ_2 value while varying λ_1, and then measure both the model sparsity and performance. As shown in Figure 1 left, higher λ_1 values significantly increase the model sparsity, meanwhile, the model performance is slightly decreased. The results indicate that we should consider the trade-off between the sparsity and the model performance, when choosing the λ_1 value. On the other hand, λ_2 value is designed to prevent the excessive knowledge drift to the local subgraph distributions, and, to verify its effectiveness, we fix λ_1 value while varying λ_2. As shown in Figure 1 right, too small lambda values lead to the performance degeneration, thus choosing the sufficiently large λ_2 values (e.g., 1e-1) would lead to the performance improvement. Also, we further observe that the sparsity does not depend on λ_2 value, thus the effects of λ_1 and λ_2 are orthogonal and complementary.

C.2 Results on Another Graph Partitioning Algorithm

Table 2: Results on experimental settings of Louvain graph partitioning algorithms, following Zhang et al. [20].

Methods	Cora	CiteSeer	PubMed
Local	78.56 ± 0.27	64.06 ± 0.09	84.07 ± 0.17
FedAvg	71.83 ± 0.40	69.23 ± 0.71	82.47 ± 0.32
FedProx	72.09 ± 0.29	67.66 ± 0.97	82.68 ± 0.34
FedPer	80.13 ± 0.50	66.28 ± 1.22	85.02 ± 0.23
FedGNN	76.59 ± 0.66	61.21 ± 1.46	82.67 ± 0.26
FedSage+	72.20 ± 0.60	68.40 ± 0.61	82.76 ± 0.09
GCFL	78.55 ± 0.38	64.20 ± 0.31	84.62 ± 0.31
FED-PUB (Ours)	**82.68** ± 0.13	**69.45** ± 0.75	**86.20** ± 0.11

To verify the effectiveness of our FED-PUB on different graph partitioning settings, we use another experimental setup from Zhang et al. [20], which uses the Louvain algorithm [2] for partitioning the entire graph into several subgraphs for clients. Note that, before explaining experimental results, we would like to point out that there is a drawback when using the Louvain algorithm presented in Zhang et al. [20], rather than using the METIS algorithm [7] as ours, for subgraph FL scenarios. Specifically, since the Louvain algorithm cannot specify the number of graph partitions, the number of subgraphs on the CiteSeer dataset is 38, where three of them have less than ten nodes. Then, based on those 38 disjoint subgraphs, to generate the particular number of clients (e.g., 10), Zhang et al. [20] randomly merge the different subgraphs without considering their graph properties. Therefore, even though each partitioned subgraph has its unique structural role/characteristic, the reconstructed 10 subgraphs from the original 38 subgraphs have mixed properties (i.e., two incompatible subgraphs could be merged), which is suboptimal. However, as described in the Datasets paragraph of Subsection 5.1, the METIS that we use can specify the number of partitions, thus more appropriate for making experimental settings for subgraph FL.

Moreover, we further conduct experiments with the Louvain graph partitioning algorithm [2, 20], on Cora, CiteSeer, and PubMed datasets with the number of clients as 10, and then report the results in

Table 2. Then, the results show that our FED-PUB consistently outperforms all the other baselines on another graph partitioning setting, thus the effectiveness of our FED-PUB becomes more obvious.

C.3 Results on Random Graph Partitioning

Since one might be curious about the results on uniform partitions of graphs, rather than splitting the graph with its partitioning algorithms (e.g., METIS and Louvain algorithms), in this subsection, we explain why this random partitioning setting is unrealistic, and then show the performances on this setting as well. Specifically, when we partition the entire graph of the CiteSeer dataset into different subgraphs uniformly at random, the number of nodes per subgraph will be larger than the number of edges (e.g., 211 nodes yet 72 edges per subgraph, thus some nodes do not have any edges), which is uncommon in practice. However, to compare the performances of different models, we further perform experiments on the random split setting with 10 different clients on the CiteSeer dataset, and then report the results in Table 3. As shown in Table 3, the gap between baselines and our model is reduced compared to the non-overlapping and overlapping scenarios in Table 1 and Table 2 of the main paper. This is because there is no specific community structure in this random setting; however, our FED-PUB still meaningfully outperforms all the other baselines under this setting as well.

Table 3: Results on experimental settings of random graph partitioning.

Methods	CiteSeer with 10 Clients
Local	44.27 ± 1.05
FedAvg	60.84 ± 0.80
FedProx	59.38 ± 1.66
FedPer	60.04 ± 0.93
FedGNN	54.64 ± 1.67
FedSage+	61.03 ± 0.11
GCFL	53.15 ± 1.82
FED-PUB (Ours)	**63.63** ± 0.86

C.4 Distribution Shifts between Subgraphs

In this subsection, we measure the label differences (i.e., distributional shifts) between subgraphs with the Jenson-Shannon divergence, in which the minimum and maximum values are 0 and 2, respectively, on the Cora dataset with 20 different clients over the overlapping and the non-overlapping scenarios. Then, the results show that the distance (i.e., divergence value) among the subgraphs within the same community is 0.384 while the distance among the subgraphs belonging to different communities is 0.639 for the non-overlapping node scenario. On the other hand, the distance among the subgraphs within the same community is 0.047 while the distance among the subgraphs belonging to different communities is 0.528 for the overlapping node scenario.

Then, from the results above: heterogeneity of subgraphs within the same community is extremely larger in the non-overlapping setting (0.384) compared to the overlapping setting (0.047), we can further confirm that personalized weight aggregation might not be enough in disjoint subgraph FL problems, since one particular subgraph might not get meaningful weights from the completely heterogeneous subgraphs. On the other hand, in this extremely heterogeneous case, a personalized weight masking scheme is clearly helpful, since it can filter out irrelevant information transmitted from the other heterogeneous subgraphs, while allowing the model to maintain the locally helpful information in its parameters. This result is also aligned with the results in Figure 7 (Ablation studies) of the main paper that, the personalized weight masking scheme brings huge performance improvements in the non-overlapping setting with high heterogeneity, whereas the personalized weight aggregation scheme is helpful in the overlapping setting with low heterogeneity.

C.5 Additional Analyses on Local Graph Size vs Heterogeneity

To see how much heterogeneity issues are severe in terms of the number of clients, we first analyze the exact amount of heterogeneities by varying the number of clients, and then provide the results on the small number of clients (i.e., the heterogeneity issue is less severe). At first, following the reported statistics in Table 1, when we increase the number of clients on the non-overlapping node scenario, the heterogeneity across subgraphs becomes more severe and problematic for personalized subgraph FL, and thus becomes an important issue to tackle.

Table 4: Results on Cora, CiteSeer, and PubMed with the number of clients as 3 over the non-overlapping scenario.

Methods	Cora	CiteSeer	PubMed
Local	81.73 ± 0.44	68.16 ± 0.25	84.81 ± 0.40
FedAvg	78.77 ± 0.13	69.34 ± 0.23	85.29 ± 0.20
FedProx	78.91 ± 0.21	69.54 ± 0.27	85.59 ± 0.18
FedPer	82.29 ± 0.13	69.80 ± 0.33	85.34 ± 0.16
FedGNN	82.36 ± 0.62	67.79 ± 0.49	85.57 ± 0.13
FedSage+	77.79 ± 1.96	69.35 ± 0.12	85.63 ± 0.22
GCFL	82.67 ± 0.74	68.85 ± 0.58	86.20 ± 0.15
FED-PUB (Ours)	**84.45** ± 0.23	**70.66** ± 0.34	**86.74** ± 0.16

While we already provide the results that our FED-PUB is effective when the number of clients is large (i.e., heterogeneous issues are severe) in Table 1 and Table 2 of the main paper, one might be curious about whether our FED-PUB is still effective, when the heterogeneity issue is less significant.

To see the results for this, we further conduct the experiment in the setting where the number of clients is smaller than 5 (i.e., 3) on the Cora, CiteSeer, and PubMed datasets of the non-overlapping node scenario. As shown in Table 4, compared to the results in Table 2 of the main paper with numbers of clients as 5, 10, and 20, the performance gaps between our FED-PUB and baselines are much reduced. However, we can clearly observe that our FED-PUB meaningfully (i.e., statistically) outperforms all the other baselines even when the number of clients is small, since there still exists incompatible knowledge across clients, which our FED-PUB effectively tackles with personalized weight aggregation and local weight masking schemes.

C.6 Results on Varying the Random Graph Inputs for Functional Embeddings

As described in Section B.3 of the supplementary file, to calculate each client model's functional embedding, we use the same random graph for all clients, which is initialized by a stochastic block model [5] with node features initialized by the normal distribution. Since such randomness does not yield any bias on the functional space, unlike existing node features for the certain subgraph, we expect our random graphs are helpful for effectively capturing the similarities among subgraphs.

Table 5: Results on varying the random graph inputs for functional embeddings, over overlapping and non-overlapping node scenarios with 20 clients on Cora.

Random Graph	Overlapping	Non-Overlapping
SBM	0.937	0.810
ER	0.920	0.712
One	0.822	0.656
Feature	0.897	0.632

In this subsection, to experimentally validate this statement, we compare various schemes used for calculating the functional embeddings: 1) SBM denotes the random graph generated from the SBM model like ours; 2) ER denotes the random graph generated from the Erdos-Renyi model [3]; 3) One denotes the random graph having only one node; 4) Feature denotes the graph where nodes are initialized by the node features in the client. We then measure the performances of those four schemes by calculating the correlation coefficients between label distributions and generated similarities of subgraphs (i.e., the high value means that the similarities from the functional embeddings are similar to the label distributions) on the Cora dataset of non-overlapping and overlapping node scenarios with 20 clients, which are reported in Table 5. As shown in Table 5, compared to the One scheme that uses only one node for calculating the functional embeddings, SBM and ER schemes that use more large numbers of randomly initialized nodes can accurately capture the similarities between subgraphs. This result confirms that a sufficient amount of randomness is required to identify the model's functional space. Also, compared to the Feature scheme that uses existing node representations for calculating the functional embeddings, SBM and ER random models show superiority in capturing similarities among subgraphs, which verifies that randomness might help obtain accurate functional embeddings of the models without incurring bias.

C.7 Results on Varying the Similarity Calculation Schemes

As shown in Figure 3 of the main paper, compared against the parameter and gradient similarities, our functional embeddings are not only effective but also efficient in capturing similarities between subgraphs. Also, using the label distributions – locally stored in the client – as the proxy for similarity calculation might violate the privacy of the user's data. However, to see their performance differences in the real-world dataset, we additionally conduct experiments on the parameter, gradient, and label similarities, on the Cora dataset of the overlapping node scenario with the number of clients as 30, and then report the results on 20, 40, 60, and 80 epochs in Table 6.

Table 6: Results on varying the similarity calculation schemes: parameter, gradient, label, and our functional embedding, on the overlapping node scenario with 30 clients of the Cora dataset.

Model	20	40	60	80
FedAvg	29.94	32.69	47.84	52.42
Parameter	29.94	35.89	47.03	52.28
Gradient	33.93	51.09	52.77	58.14
Label	65.97	74.31	76.50	76.82
Function (FED-PUB)	67.82	73.51	74.66	75.90

As shown in Table 6, we can observe that the models, which utilize the parameter and gradient for calculating the similarities between subgraphs, perform similarly to the FedAvg model and perform worse than our functional and label similarity schemes. However, even though the label similarity model uses privacy-sensitive local information (i.e., label distributions of every client), the performance of our FED-PUB that utilizes the functional embeddings is similar to the performance of the label model. Therefore, along with the results in Figure 6 of the main paper, this comparison results on similarity schemes further verify the effectiveness of our functional embedding scheme in capturing the similarities among subgraphs.

C.8 Impacts of Missing Edges to Performance Degeneration

In this section, we show that, due to the problem of missing edges, all the FL methods, which observe edges only within each subgraph, show inferior performances than the Oracle method, which trains on the entire graph including missing edges. In other words, we train the Oracle model on the connected global graph, and then evaluate it on disjoint subgraphs over all clients, on the Cora dataset of both Non-overlapping and Overlapping node scenarios with varying client numbers. Then, as shown in Table 7, the Oracle model outperforms all the other methods, while our FED-PUB achieves the closest performance to the Oracle model. The above results bring us to the following points. First, due to the problem of missing edges, all the FL methods, which observe edges only within each subgraph, perform poorly than the Oracle method. Also, the missing edge problem negatively affects the incompatible knowledge issue: since all client models are trained by the partial subgraphs, which are parts of the larger global graph, the trained parameters in the client and the aggregated parameters in the server might not capture globally meaningful knowledge or the knowledge that is helpful to the other clients, which explains why the Oracle performs the best.

Table 7: Results on Non-Overlapping and Overlapping node scenarios with varying the number of clients on Cora, including the Oracle model that is not comparable.

Model	NonOverlapping-5	NonOverlapping-20	Overlapping-10	Overlapping-50
Oracle	85.07	85.47	85.08	85.28
Local	81.30	80.30	73.98	76.63
FedAvg	74.45	69.50	76.48	53.99
FedGNN	81.51	70.10	70.63	56.91
FedSage+	72.97	57.97	77.52	55.48
FED-PUB (Ours)	83.70	81.75	79.60	77.84

D Limitations and Potential Societal Impacts

In this section, we discuss the limitations and potential societal impacts of our work.

Limitations While our personalized subgraph FL framework, namely FED-PUB, is generally applicable regardless of the types of subgraphs (e.g., unipartite graphs or bipartite graphs), our experiments are mainly done with unipartite graphs which are the most popular graph representation learning settings. However, the behaviors of our FED-PUB on the other types of graphs, such as bipartite graphs, would be interesting to further see, which have but not been explored so far, and we leave this as future work.

Potential Societal Impacts The FL scheme is important for preserving users' privacy, and, while this scheme is actively studied in the image and language domains, it gets little attention for graphs. In particular, the subgraph FL, which we mainly target, has unique challenges on missing nodes, edges, and their community structures, and we believe which are sufficiently tackled in our work.

Then, the potential positive impact of our work on society is that, our method affects various domains that use graphs, such as social networks, recommendation networks, and patient networks, to name a few. Note that we would like to emphasize the importance of our subgraph FL scheme, especially in social and recommendation networks. In the current real-world applications, all the user's interactions with other users for social networks and with other products for recommendation networks may be stored in the server. However, this may not preserve the user's privacy, but also has potential risks of a leak of user data from the server, such that storing the user's data in the server is not recommended, for example, from the data protection regularizations such as GDPR [1]. Then, by applying our subgraph FL framework to this domain, we expect such problems could be resolved by not storing user's interaction data to the server, but only sharing the locally trained models with clients.

However, the transmitted model parameters from the client to the server may hold the privacy-sensitive information, and, while this is not the main focus of this work (i.e., we assume that model parameters are transmittable without compromising privacy as in many FL works [12, 10, 1]), the research community may need to put further effort on whether the model parameters are safe, and how to make them safe if the model parameters contain privacy-sensitive information.

[1] https://gdpr-info.eu/

References

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

[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[3] P. Erdős and A Rényi. On the evolution of random graphs. In *PUBLICATION OF THE MATHEMATICAL INSTITUTE OF THE HUNGARIAN ACADEMY OF SCIENCES*, pages 17–61, 1960.

[4] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[5] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983. ISSN 0378-8733.

[6] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[7] George Karypis and Vipin Kumar. Metis – unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, 1995.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[9] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[10] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020.

[11] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

[12] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

[15] 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.

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

[17] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

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

[19] Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs. In *Advances in Neural Information Processing Systems*, volume 34, pages 18839–18852. Curran Associates, Inc., 2021.

[20] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 6671–6682. Curran Associates, Inc., 2021.