A Novel Graph Federated Learning Framework in Hyperbolic Space

Haizhou Du[a], Conghao Liu[a], Haotian Liu[b], Angela Huo[b]

[a]*Shanghai University of Electric Power, China*
[b]*University of Technology Sydney, Australia*

Abstract

With the increasing number of graph data, Graph Federated Learning (GFL) has emerged and been used in medicine, chemistry, social networks and other fields. Consequently, the efficiency of graph classification has become a crucial issue in the GFL framework. However, due to the high distortion and redundancy in graph information, the existing works are troubled by the low accuracy of classification. In this paper, we propose a novel efficient GFL framework for graph classification, namely FedHGCN. FedHGCN has two novel features: 1) collaboratively train Graph Neural Network (GNN) in a high dimension space to capture the rich hierarchical feature of graphs. 2) build a strategy of node selection to remove the redundancy from the graph representation and highlight key nodes. Our extensive experiments show that FedHGCN outperforms the state-of-the-art approaches up to 15.6% by accuracy on four publicly available graph datasets. Furthermore, we prove that FedHGCN can efficiently deal with various poisoning attacks by experiments.

Keywords: Graph Federated Learning, Graph Classification, Hyperbolic Space, Node Selection.

1. Introduction

As more advanced techniques are developed for learning with graph data, using graphs to model and solve real-world problems becomes more popular. One important scenario of graph learning is graph classification [1, 2, 3], where models are used to complete molecular property predict for AI medicine [4] and bioinformatics in medical fields [5]. As the amount of graph data and the demand of medical institution increasing, graph classification federated learning is proposed to solve a data leak problem and train more powerful graph learning model which is needed to conduct effective inference over graph information [6, 7, 8, 9]. In particular, these methods can often be used in cross-silo scenarios of the medical field [10], which is shown in Figure 1. Medical institutes, such as hospitals

or research institutes, usually train healthcare datasets, including drug molecules, protein molecules and anticancer substances [11, 12]. It is rather difficult to let all medical institutions share their graph data with others to train the graph classification model due to conflicts of interests. GFL methods allow clients to upload model parameters instead of their graph data. Meanwhile, GFL avoid the privacy concerns associated with sharing models, which fits the needs of this scenario very well.

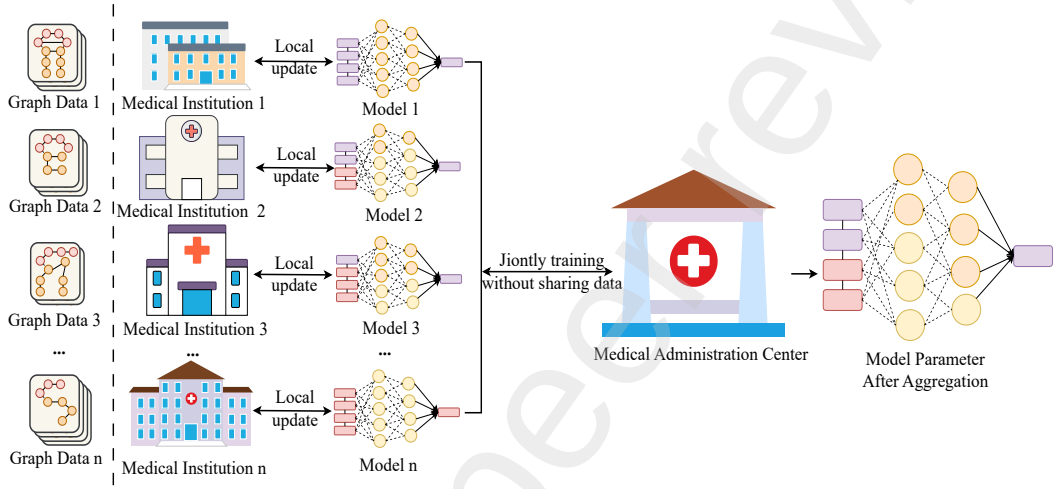


Figure 1: An example of GFL for graph classification in cross-silo scenes (**motivated scenario**): In this example, there are some medical institutions and an administration center. In the left part of the figure, each medical institutions has strong computing power and updates a local model parameter by training on the graph dataset it owns. The right part of the figure indicates our goal that the framework obtains a globally powerful graph model without data sharing.

However, existing GFL methods are still troubled by poor accuracy of classification. This is because this novel yet realistic setting brings two challenges:
Challenge 1: How to effectively learn important hierarchical information from multiple local graph datasets? Existing GFL methods are not suitable for the graph datasets on clients in cross-silo scenarios. Real-world graphs in such scenarios often exhibit scale-free or hierarchical structure [13]. Euclidean embeddings, which are commonly used in existing FL methods for graphs, suffer from high distortion when embedding graphs. This results in a significant drop in the accuracy of graph classification.
Solution 1: FedHGCN: Map graph information to hyperbolic space. To address the issue of high distortion, we propose a novel and accurate framework for GFL in hyperbolic space called FedHGCN, which is designed for graph classification. Hyperbolic geometry enables embedding with much smaller distortion

when embedding scale-free and hierarchical graphs. Additionally, hyperbolic embedding has been shown to capture the hierarchical structure in the graph [14], which can highlight important nodes.

Challange 2: How to address over-smoothing problem of GNNs caused by silver harmful nodes? Over-smoothing issue is a known issue in graph learning, particularly in graph learning with GCN. This issue arises due to excessive similarity of node representations [15], which can lead to misclassification.. In the federated learning framework for graph learning, if too many client models appear to be over-smoothing, the performance of the entire framework can be severely compromised.

Solution 2: FedHGCN with node selection: Keep important nodes in the graph and remove redundancy. To deal with the redundancy, we elaborately transmit the features of key nodes of FedHGCN. Specifically, we aim to simplify the graph representation of every client without altering the graph's structure.

In a nutshell, the **main contributions** of this paper can be summarized as follows:

- We propose FedHGCN as a novel federated graph learning framework and present the procedures of global parameter aggregation at the server and local parameter training at clients. To the best of our knowledge, this is the first work extending hyperbolic space to a federated learning architecture.
- On the client side, a lightweight node selection-based aggregating operation scheme in hyperbolic space is adopted to extract important nodes and improve graph representations. Meanwhile, some unimportant information will be ignored to reduce the redundancy of the graph information.
- We conduct extensive experiments to evaluate the performance of FedHGCN. Results show that the accuracy of FedHGCN significantly outperforms state-of-the-art studies by up to 15.6% on different real-world graph datasets. In addition, the evaluation proves that FedHGCN have the ability to prevent data-poisoning attacks and better scalability.

The rest of this paper is organized as follows. We review some necessary related work in Section 2. The FedHGCN framework design is presented in Section 4. Section 5 gives experimental settings and results. Finally, Section 6 concludes this paper.

2. Related Work

This section briefly reviews the related work on GFL framework and graph neural networks in hyperbolic space.

2.1. Graph Federated Learning

Recent researchers have made some progress in GFL [7, 6, 8]. He et al. [7] design an open FL benchmark system that can facilitate research on federated GNNs. Mei et al. [16] propose a similarity-based graph neural network model to precisely capture node structure information in node classification tasks. Zhu et al. [17] propose federated learning to obtain a generalized global model without access to private molecular data. Meng et al. [18] propose a cross-node federated graph neural network to encode the underlying graph structure using a GNN-based architecture under the constraint of cross-node federated learning. Zhang et al. [10] embed the GraphSAGE model into the FedAvg framework [19] and makes adaptive adjustments. Wu et al. [6] propose a federated framework for privacy-preserving GNN-based recommend systems. Han et al. [8] discover common patterns shared among graphs across datasets through experiments. However, none of these methods put forward effective information sharing scheme for the common patterns.

2.2. Hyperbolic Graph Neural Networks

Since the hyperbolic graph neural networks [20] were proposed, hyperbolic geometry has been applied to neural networks, to solve problems of computer vision, natural language processing and spatio-temporal forecasting [14, 13, 21, 22]. More recently, hyperbolic neural networks [23] was proposed, where core neural network operations are in hyperbolic space. Such as the Hyperbolic Graph Neural Network (HGNN) mainly aims to improve the performance of graph-structured data by embedding graphs to hyperbolic space [24]. Dmitri et al. [25] show that typical properties of complex networks, such as heterogeneous degree distributions and strong clustering, can be explained by assuming an underlying hyperbolic geometry and using these insights to develop a geometric graph model for real-world networks. Chami et al. [20] proposed a hyperbolic graph convolutional neural network to exploit the property of hyperbolic embeddings to embed tree-like graphs with low distortion. More recently, some of the shortcomings of hyperbolic embedding have been further optimized [26] and show better results, but Lorentz linear layer is not suitable for graph classification scenarios. However, due to the high redundancy in graph information, these methods can only achieve a low classification accuracy.

3. Preliminaries

This section first describes the basic arithmetic in hyperbolic space. We then introduce node embedding from Euclidean space to hyperbolic space. Finally, we introduce the rule of a simple Hyperbolic graph convolution network.

3.1. Hyperbolic geometry

Hyperbolic geometry offers an exciting alternative as it enables embeddings with much smaller distortion when embedding scale-free and hierarchical graphs [13]. Among the hyperbolic geometry, Poincaré embedding [27] can learn the hierarchical representation of symbol data by embedding symbol data into N-dimensional Poincaré balls. The focus of this work is to effectively model the link structure of symbolic data, that is, to find low-dimensional embedding by using the hierarchical structure of hyperbolic space. Presently, hyperbolic geometry has been applied to neural networks, to solve the problems of computer vision, natural language processing and spatio-temporal forecasting [21, 22].

3.2. Computing rules in hyperbolic space and hyperbolic embedding

The Poincaré ball model (D^n, g^D) is defined by the manifold $D_n = \{x \in R_n : \|x\| < 1\}$ equipped with Riemannian metric as follow:

$$g_x^D = \lambda_c^2 g^E, \tag{1}$$

where $\lambda_c := \frac{2}{1-\|c\|^2}$ and c is the curvature in hyperbolic space. Möbius addition of x and y is defined as Equation(2):

$$x \oplus_c y := \frac{(1 + 2c\langle x, y\rangle + c\|y\|^2)x + (1 - c\|x\|^2)y}{1 + 2c\langle x, y\rangle + c\|x\|^2\|y\|^2}, \tag{2}$$

when c equals to 0, this function represents the addition in Euclidean space. Möbius scalar multiplication of r and x is defined as Equation(3):

$$r \otimes_c x := tanh\big(r\, tanh^{-1}(c\|x\|)\big)\frac{x}{c\|x\|}. \tag{3}$$

To capture the hierarchical information in the graph through the hyperbolic space ,it is necessary to embed graph information between Euclidean and hyperbolic spaces by exponential mapping and logarithmic mapping. The exponential mapping is shown as Equation(4) :

$$x^H = exp_p^c(x) = p \oplus (tanh(c\frac{\lambda_{c,p}\|x\|}{2})\frac{x}{c\|x\|}), \tag{4}$$

p is the initial node of the mapping, when p = 0, the exponential map is defined as:

$$x^H = exp_0^c(x) = \frac{tanh(c\|x\|)x}{c\|x\|}. \tag{5}$$

The logarithmic mapping is shown as Equation (6):

$$x = log_q^c(x^H) = \frac{2}{c\lambda_{c,q}} tanh^{-1}(c\|x^H\|)\frac{x^H}{\|x^H\|}, \tag{6}$$

q is the initial node of the mapping, when q = 0, the logarithmic mapping is defined as:

$$x = log_0^c(x^H) = \frac{tanh^{-1}(c\|x\|)x}{c\|x\|}, \tag{7}$$

where x denotes the information in the Euclidean space and x^H denotes the information in the hyperbolic one. In addition, c is the curvature of the manifold. 0 means mapping from the original point of the coordinate axis.

3.3. Hyperbolic Graph Convolution Network

Given a set of graph data $G = \{g_1, g_2, ..., g_n\}$, where the number of nodes and edges in each graph might be quite different. For an arbitrary graph $g_i = (v_i, \varepsilon_i, x_i)$, we have n_i and e_i to denote the number of nodes and edges. Let $A_i \in R^{n_i \times n_i}$ be the adjacent matrix describing its edge connection information and D is the diagonal degree matrix of A. $x^E \in R^{n_i \times f}$ represents the Euclidean node feature matrix, where f is the dimension of node attributes and $x^H \in H^{d,K}$ represents the hyperbolic node feature matrix, where c is the curvature in hyperbolic space. $x^\tau \in \tau_x H^{d,K}$ represents the (Euclidean) tangent space centered at point x . A simple HGCN [13] with message passing rule at layer l then consists of two process. HypLinear is the linear transform in hyperbolic space, which is defined as:

$$h^{H,l} = HypLinear(x^{H,l-1}) = W \otimes x^{H,l-1} \oplus b^l. \tag{8}$$

HypAggregate combines the characteristics of the tangent space with the aggregation process in Euclidean space. The aggregation process in the Euclidean space is defined as:

$$x = AGG(h) = \sum_{j \in n} A \cdot h. \tag{9}$$

Thus, the HypAggregate process is defined as:

$$x^{H,l} = HypAggregate(h^{H,l}) = exp_0^c(\sum_{j \in n} A \cdot log_0^c(h^{H,l})). \tag{10}$$

Based on this information, we propose a novel GFL framework that can capture the graph information from hyperbolic space and then highlight key nodes in the graph in this paper.

4. FedHGCN Framework Design

In this section, we first introduce the notions and the problem of GFL for graph classification in hyperbolic space. We then provide an overview of the FedHGCN framework, with details of its design presented at the end of this section.

4.1. Notations and Problem Formulation

Given a set of graph data $G = \{g_1, g_2, ..., g_n\}$, where the number of nodes and edges in each graph may differ significantly. For an arbitrary graph $g_i = (v_i, \varepsilon_i, x_i)$, we use n_i and e_i to denote the number of nodes and edges, respectively. Each client can determine which label each graph x_i belongs to via its graph neural network, and obtain the label $\tilde{Y}_i$. Label matrix $Y \in R^{n \times c}$ indicates the associated labels for each graph and we obtain the classified graph labels $\tilde{Y}$. The objective function of each client is defined as the cross-entropy of predictions over the labels. Therefore, it can be simplified as follows:

$$J_m(w) = \frac{1}{n}\sum_{i=1}^{n} Y_i log(\tilde{Y}_i). \tag{11}$$

We then calculate the average loss of all clients as our final optimization function in Equation 12:

$$minJ = \sum_{m=1}^{M} J_m(w). \tag{12}$$

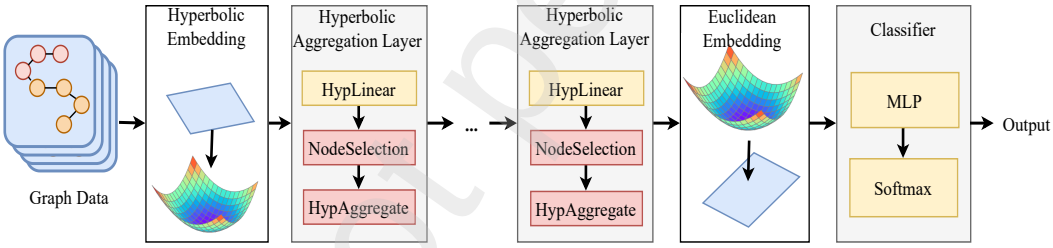


Figure 2: The workflow of FedHGCN at each client

4.2. Client model in FedHGCN Design

In this subsection, we describe the model used by FedHGCN in each client. As illustrated in Figure 2, FedHGCN client includes four key layers, namely the hyperbolic mapping layer, hyperbolic aggregation layer, euclidean mapping layer and graph classification layer. The hyperbolic aggregation layer is divided into two parts: node selection process and hypaggregation layer with selection matrix.

4.2.1. Hyperbolic Embedding and HypLinear Layer

At the beginning of the workflow, we first use exponential mapping to map Euclidean input features x_{i-1} into the hyperbolic manifold.This yields features x_{i-1}^H in hyperbolic space, as given by Equation 5:

$$x_{i-1}^H = exp_c^0(x_{i-1}). \tag{13}$$

We then utilize a linear transformation in the hyperbolic space to improve graph learning to enable learnable convolution. Thus, the hyperbolic linear layer is defined as:

$$h_i^H = HypLinear(x_{i-1}^H) = W \otimes x_{i-1}^H \oplus b^l. \tag{14}$$

4.2.2. Node Selection Proces

To address the over-smoothing problem caused by the redundancy in graph neural networks, many researchers have focused on improving the methods of extracting structural features of graphs. In addition, poisoning attacks are also a very troublesome problem in the field of graph learning. Among them, adversarial attacks often generate redundant nodes, which has a significant impact on the structure of the graph [28]. Since this kind of attack changes the structure of the graph, it seriously affects the accuracy of graph learning. Therefore, adversarial attacks not only affect the result of node classification, but also lead to the error of graph classification.

Inspired by [1, 15], we adopt a method of node selection to address these issues. The process of reducing nodes involves identifying a subset of informative nodes to form a new graph. This method significantly reduces the influence of unimportant nodes on graph learning and reduces the redundancy of graph information. Moreover, this method can remove some noise nodes and further ensure the stability of the graph structure.

We first map the nodes from hyperbolic space to the tangent space because aggregating in tangent space can reduce unnecessary resource consumption in the process of aggregation. More importantly, this mapping process does not affect graph learning [13]. The mapping process is in Equation 15:

$$h_{i-1}^{\tau} = log_c^{h_i^H}(h_i^H). \tag{15}$$

On this basis, we introduce a criterion named node information score p to evaluate the information that each node contains given its neighborhood. Here, we formally define the node information score in Equation 16 as the Manhattan distance [29] between the node representation itself and the one constructed from its neighbors, because it represents a common similarity measure that is especially convenient for high dimensional vectors:

$$s = \|(I - (D)^{-1}A)h_i^{\tau}\|_1, \tag{16}$$

where A and D are the adjacent and diagonal degree matrix of A. I is the identity matrix. Thus we will have a constant set p as the information score of each node in the graph.

Then we use the $topk$ method to reserve the nodes with higher scores in Equation

17 and get a baseline rating r to build a selection matrix ω to participate in the Equation 18:

$$r = min(topk(s)), \tag{17}$$

$$\omega = \{ \begin{matrix} 1, s > r \\ 0, otherwise \end{matrix} . \tag{18}$$

4.2.3. HypAggregation Layer with Selection Matrix

Once the selection matrix ω is learned, the model learns the embedding feature while also being able to cancel part of the node aggregation. Therefore, we gain a weight matrix α for aggregation by concatenating the aggregated feature matrices before and after screening matrices in Equation 19:

$$\alpha = \sigma(log_c^{h^H}(HypLinear(C))), \tag{19}$$

where $C = exp_c^{h^H}(\sigma(AGG(h_{i-1}^\tau)) \| \sigma(AGG(\omega \cdot h_{i-1}^\tau)))$ is the concat sum of h_{i-1}^τ.

At last, we allow the layer to aggregate the embedding information only from its selected neighbors in Equation 20:

$$z = AGG(\alpha \cdot \omega \cdot h_i^\tau), \tag{20}$$

Therefore, the final aggregate function is as shown in Equation 21:

$$x_i^\tau = \sigma(z_i) + \sigma(h_i^\tau), \tag{21}$$

where z_i is the embedding information from selected nodes and h_{i-1}^τ is the learned graph feature matrix in the hyperbolic space. At last, we map the most novel graph feature matrix x_i^τ into hyperbolic space via Equation 22:

$$x_i^H = exp_c^{h^H}(x_i^\tau). \tag{22}$$

4.2.4. Euclidean Mapping Layer

After the aggregation, we map the graph feature matrix back to Euclidean space via logarithmic mapping in Equation 23:

$$x_i^E = log_c^o(x_i^H). \tag{23}$$

4.3. The Architecture of FedHGCN Framework

In the client side of the FedHGCN, each client maintains local graph data and learns the graph representation with the user embeddings from its graph data. During each epoch of local training, every client first computes $P_m \leftarrow P - \eta J_m(w_{m,t}, Y_i)$, where $w_{m,t}$ contains all the weights and biases in the m-th local HGCN for epoch t, and η is the learning rate, The central server then collects the

latest $\{P_m|m \in M\}$ and aggregates the parameters received from the Equation 24:

$$P_t = \frac{1}{M}\sum P_{m,t}. \tag{24}$$

To evaluate the effectiveness of the FedHGCN framework, the following section will assess its performance. Next, the central server sets P_t as the averaged value by averaging over $\{P_m|m \in M\}$. Finally, the central server broadcasts P_t to data owners and finishes one round of training. Both the weights and biases in the hyperbolic aggregation layer and the classifier are Euclidean [13].

To verify the effectiveness of FedHGCN , the following section will evaluate the performance of FedHGCN framework.

5. Evaluation

This section presents the experimental analysis of FedHGCN framework. We first introduce graph-based benchmark datasets for graph classification and baselines in subsection 5.1 and subsection 5.2, respectively. We then evaluate the graph classification performance of FedHGCN in subsection 5.3. In subsection 5.4, we design scalability experiments to assess the framework's performance as the number of clients increases. Finally, through ablation experiments detailed in subsection 5.5, we explore how FedHGCN enhances accuracy and ensures robustness. In addition, in subsection 5.6 we explore what percentage of node selection is optimal in different scenarios.

Our experiments are conducted on a CPU/GPU cluster equipped with 14 Tesla T4 GPUs and 1024GB 3200MHz memory. We utilize Python 3.7, PyTorch 1.11, and the corresponding torch-geometric 2.0.4 environment to implement FedHGCN. All the following experiments in this paper use this federated learning architecture.

5.1. Datasets

We evaluate our model using four real-world datasets from Tudataset [30], which are summarized in Table 1 with the following specifications:

- PROTEINS [31] is a protein graph dataset that has received a lot of attention in the field of graph classification.
- D&D [31] contains graphs of protein structures. A node represents an amino acid. A label denotes whether a protein is an enzyme or a non-enzyme.
- NCI1 [32] is a biological dataset used for anticancer activity classification. In the dataset, each graph represents a chemical compound, with nodes and edges representing atoms and chemical bonds, respectively.

- Mutagenicity [33] is a chemical compound dataset of drugs, which can be categorized into two classes: mutagen and non-mutagen.

To ensure each client can complete the training task, we first assign 10 graphs from the dataset to each client. Then, we assign the rest of the dataset to each client using the Dirichlet distribution $Dirlet$. Within each client, we randomly split each sub-dataset into three parts: 80% as a training set, 10% as a validation set, and the remaining 10% as a test set.

Table 1: Statistics of the Datasets.

Datasets	Number of Graphs	Avg. of Nodes	Avg. of Edges	Classes
PROTEINS	1,113	39.06	72.82	2
D&D	1,178	284.32	715.66	2
NCI1	4,100	29.87	32.30	2
Mutagenicity	4,337	30.32	30.77	2

5.2. Baselines and Experiment settings

We compare the performance of FedHGCN with several methods that have proven excellent graph learning abilities based on decentralized storage of user data and several privacy-preserving ones based on federated learning. These methods include:

- FedAvg[19]: FedAvg is a classical FL framework. It is typically used to run some image classification tasks. Therefore, due to better learning of the graph information, we set GCN[34] as the model in the framework.
- FedGraphNN[7]: FedGraphNN is an open FL benchmark system that facilitates research on federated GNNs.
- FedSAGE[10]: FedSAGE is a FL framework that trains several GraphSage[35] models based on FedAvg to integrate graph tasks on multiple local subgraphs.

To complete the graph classification task, we set the client model of each framework into three parts: three layers of graph neural network with the hidden size of 64, a global-add-pool layer and a softmax classifier with two fully connected layers with the hidden size of 64 for every framework. We use a batch size of 64, and an Adam [36] optimizer with a learning rate of 0.001, dropout of

0.01 and weight decay $5e^{-4}$. Moreover, in FedHGCN, we set the node selection ratio to 0.8. When this parameter is set to 1.0, all nodes are preserved. We adopt 500 iterations in each training session and take the result of 50 iterations after finding the iteration with the lowest loss value in each training session epoch.

To measure the performance of FedHGCN and the baselines, we design three different experiments, which are accuracy experiments, scalability experiments and ablation experiments.

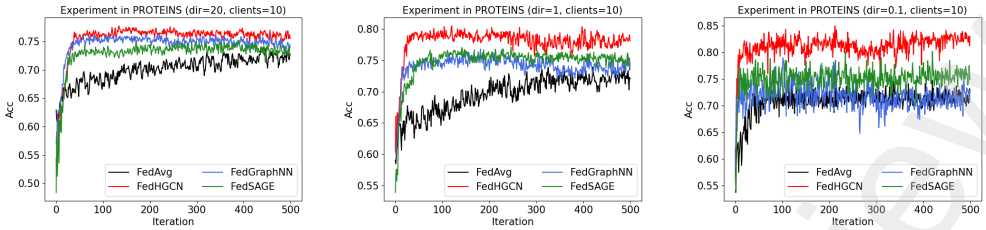
5.3. Performance of Accuracy Experiment

In this subsection, we deploy FedHGCN on 10 computing clients in two different experimental settings. First, We distribute the dataset equally for 10 computing clients and test the accuracy trends in an Independent and Identically Distribution (IID) dataset distribution scenario. We then find that the distribution of data in real life is mostly Non-IID. Thus, we use the Dirichlet distribution [37] to create disjointed Non-IID client training data. We consider two data distribution heterogeneous settings, which respectively follow Dirichlet distribution $Dirlet$, where $Dirlet$ is set to 1 and 0.1. If the parameter of Dirichlet distribution is close to 0, the distribution of the dataset in different clients is close to Non-IID.

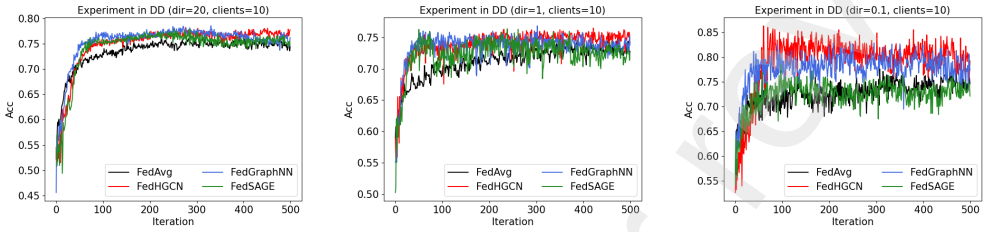
Table 2: The accuracy results of graph classification on four datasets in IID and Non-IID scenarios.

	PROTEINS	D&D	NCI1	Mutagenicity
FedAvg [19] (IID)	72.48%	75.25%	70.42%	76.27%
FedGraphNN [7] (IID)	75.51%	77.49%	74.11%	**81.21%**
FedSAGE [10] (IID)	74.11%	76.66%	74.75%	77.26
FedHGCN (IID)	**76.94% (↑6.2%)**	**77.72% (↑3.2%)**	**75.86% (↑7.7%)**	80.23% (↑5.2%)
FedAvg [19] ($Dirlet$=1)	72.32%	73.39%	73.02%	76.78%
FedGraphNN [7] ($Dirlet$=1)	75.35%	74.54%	77.02%	81.80%
FedSAGE [10] ($Dirlet$=1)	76.13%	73.99%	74.79%	79.62
FedHGCN ($Dirlet$=1)	**79.06% (↑9.3%)**	**75.36% (↑2.7%)**	**77.91% (6.6%↑)**	**82.60% (↑7.5%)**
FedAvg [19] ($Dirlet$=0.1)	71.52%	75.09%	73.60%	76.08%
FedGraphNN [7] ($Dirlet$=0.1)	74.00%	79.41%	75.21%	78.85%
FedSAGE [10] ($Dirlet$=0.1)	76.14%	74.24%	73.47%	78.30
FedHGCN ($Dirlet$=0.1)	**83.02% (↑15.6%)**	**81.93% (↑9.1%)**	**76.88% (↑4.5%)**	**79.66% (↑4.7%)**

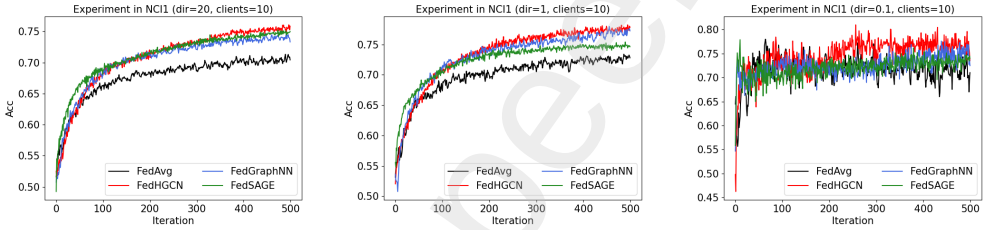
As seen in Table 2, FedHGCN has better performance in different experimental settings. The most important observation emerging from the results is that the accuracy of FedHGCN reach 76.94%, 79.06% and 83.02% in three different data distribution settings in the PROTEINS experiment, which are 6.2%, 9.3% and



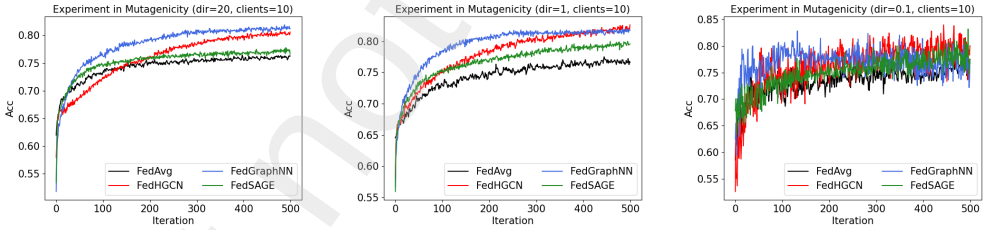
(a) PROTEINS



(b) DD



(c) NCI1



(d) Mutagenicity

Figure 3: Accuracy trends of different data sets and different data distributions: The left figure is the accuracy trends in IID scenario, the middle figure is the Non-IID scenario with $Dirlet$=1, and the $Dirlet$ in the right figure is set to 0.1.

15.6% better than that of FedAvg. The accuracy of FedHGCN reaches 77.72%, 75.36% and 81.93% in three different data distribution settings in the D&D experiment. These performances are also the best in all the framework experiments. Moreover, as shown in Figure 3a and Figure 3b, the accuracy of FedHGCN and that of other architectures reach a high level at 50 to 100 iterations. This phe-

nomenon illustrates that FedHGCN maintains high classification accuracy without significantly compromising the convergence rate. Taking the above two points into consideration, FedHGCN has better performances than that of other frameworks for distributed graph classification when using protein-type datasets.

In the experiment of training molecular-type data set, such as NCI1, the accuracy of FedHGCN reached 75.86%, 77.91% and 76.88% in three different data distribution settings, which respectively have 7.7% 6.6% and 4.5% improvements compared to FedAvg. This improvement is less pronounced than that in the PROTEINS experiments. Moreover, as shown in Figure 3d, FedHGCN compromises the convergence rate compared to FedGraphNN and FedSAGE, especially in IID scenarios. But these compromises are not obvious in the non-IID scenarios. This is because the node densities of these datasets are usually small, so the node selection we used may affect the learning of graph structure, resulting in a decrease in the accuracy of graph classification. Therefore, the degree of node selection and when to use it is open for discussion.

Table 3: Scalability experiments on PROTEINS datasets.

	10 clients	30 clients	50 clients
FedAvg (IID)	72.48%	71.68%	71.62%
FedGraphNN (IID)	75.51%	74.36%	72.92%
FedSAGE (IID)	74.11%	67.39%	67.39%
FedHGCN (IID)	**76.94% (↑6.2%)**	**76.17% (↑6.2%)**	**73.86% (↑3.1%)**
FedAvg ($Dirlet$=1)	72.32%	70.29%	72.24%
FedGraphNN ($Dirlet$=1)	75.35%	75.08%	74.30%
FedSAGE ($Dirlet$=1)	76.13%	67.95%	67.95%
FedHGCN ($Dirlet$=1)	**79.06% (↑9.3%)**	**78.46% (↑11.6%)**	**75.62% (↑4.6%)**
FedAvg ($Dirlet$=0.1)	71.79%	71.58%	70.26%
FedGraphNN ($Dirlet$=0.1)	74.00%	74.47%	71.12%
FedSAGE ($Dirlet$=0.1)	76.14%	75.52%	68.12%
FedHGCN ($Dirlet$=0.1)	**83.02% (↑15.6%)**	**79.93% (↑11.6%)**	**72.63% (↑3.4%)**

5.4. Results of Scalability Experiments

Considering the increasingly large amount of data and a growing number of participants, the scalability of FedHGCN also needs to be evaluated. Thus, we conduct scalability experiments on PROTEINS and NCI1 by changing the number of clients from 10 to 30 and 50. As shown in Table 3 and Table 4, the most

Table 4: Scalability experiments on NCI1 datasets..

	10 clients	30 clients	50 clients
FedAvg (IID)	70.42%	68.52%	68.67%
FedGraphNN (IID)	74.11%	**71.69%**	**71.93%**
FedSAGE (IID)	74.75%	69.98%	69.46%
FedHGCN (IID)	**75.86% (↑7.7%)**	70.15% (↑2.3%)	69.86% (↑1.7%)
FedAvg ($Dirlet$=1)	73.02%	71.04%	68.76%
FedGraphNN ($Dirlet$=1)	77.02%	**72.96%**	**71.81%**
FedSAGE ($Dirlet$=1)	74.79%	71.09%	70.58%
FedHGCN ($Dirlet$=1)	**77.91% (↑6.6%)**	72.73% (↑6.6%)	71.71% (↑6.6%)
FedAvg ($Dirlet$)	73.60%	66.91%	68.76%
FedGraphNN ($Dirlet$=0.1)	75.21%	70.43%	71.81%
FedSAGE ($Dirlet$=0.1)	73.47%	70.34%	70.58%
FedHGCN ($Dirlet$=0.1)	**76.88% (↑4.5%)**	**73.17% (↑9.3%)**	**72.93% (↑6.1%)**

important observation emerging from the results is that, for FedHGCN, the accuracy of PROTEINS and NCI1 respectively dropped to 79.93% and 73.17%, which is still 11.6% and 9.3% higher than the performance for FedAvg, when the Dirichlet distribution parameter $Dirlet$ is set to 0.1. Meanwhile, when the number of clients sets to 50 on the PROTEINS dataset, the performance of FedHGCN suffers a significant decline, especially when $Dirlet$ is set to 0.1. The accuracy of FedHGCN drops to 72.63%, which is 3.4% higher than that of FedAvg. We believe that there are two reasons for this result. First, in order to ensure that all 50 clients can complete the training task, we extract 500 graphs and evenly divided them among the 50 clients. This results in our experiment being more similar to the IID scenario. Second, the datasets obtained by all clients decreased significantly, resulting in inadequate training. In the experiment of NCI1, although the performance of FedHGCN in the IID scenario is slightly worse than that of FedGraphNN, the performance of FedHGCN in the non-IID scenario is more stable and more than 6% higher than that of FadAvg. This proves that FedHGCN has a better ability to handle more clients participating in training when part of the client's data set is sufficient.

5.5. Results of Ablation Experiments and Data-poisoning Attack Experiments

In this subsection, we analyze the performance of FedHGCN with and without node selection to prove the effectiveness of node selection. To demonstrate

the robustness, of FedHGCN we then conduct experiments on the PROTEINS and Mutagenicity datasets to simulate data-poisoning attacks The attackers can inject noisy or original data into training sets of a subset of clients. To simulate data-poisoning attacks in real-world scenarios, We deploy FedHGCN on 10 computing clients and poison the data of those clients. We introduce two parameters: the percentage of poison size and the percentage of poison rounds. Specifically, the percentage of poison clients indicates the number of clients controlled to participate in data-poisoning attacks, which is set as 25% 50% and 70%. The percentage of poison sizes represents the amount of noise injected into each graph data compared to the original graph data, which is set to 25%.

As shown in Figure 4, the accuracy of FedHGCN without node selection only reaches 76.25%, 72.64% and 76.16% in three different data splitting scenarios when using PROTEINS. Meanwhile, the performance of FedHGCN with node selection reaches 79.94%, 79.06% and 83.02%, which increases by 0.9%, 8.8% and 9.0% in the same scenarios. In experiments using the Mutagenicity dataset, the highest improvement reaches 9.2%. This phenomenon proves that node selection can enhance the graph learning ability by removing the redundancy of graph information without affecting the graph structure.

As shown in Figure 5, when the poison clients account for 25% of the dataset, the accuracy of FedHGCN reaches 77.93%, which is the highest among all baselines in the NCI1 dataset. The accuracy drops to 69.87% from 77.93% as the percentage of poison clients increases. For the Mutagencity dataset, the accuracy of FedHGCN reaches 71.87% when the poison clients account for 25% of the dataset. The accuracy drops to 70.2% from 71.87% as the percentage of poison clients increases. The decrease of 1.67% is the smallest among all baselines. These results demonstrate that FedHGCN has the ability to withstand data-poisoning attacks as the percentage of poison clients increases.

To sum up, node selection plays a key role in FedHGCN. Meanwhile, this method can prevent some bad information from poisoning graph data, which let FedHGCN has the ability to withstand data-poisoning at- tacks as the percentage of poison clients increases.

5.6. Hyperparameter Study and Visualization

In this subsection, we discuss the influence of node selection rate on the accuracy of FedHGCN for graph classification under different experimental settings. We set up two scenarios: (1) 10 clients training the PROTEINS dataset. (2) 50 clients training NCI1 dataset. The data distributions in both scenarios are non-IID with(the Dirichlet parameter $Dirlet$ set to 0.1). We respectively set the ratio of node selection to 0.5, 0.8, and 1.0. When this ratio is set to 1.0, all nodes in the graph are preserved.

As shown in Figure 6, in the experiments of PROTEINS, the best accuracy

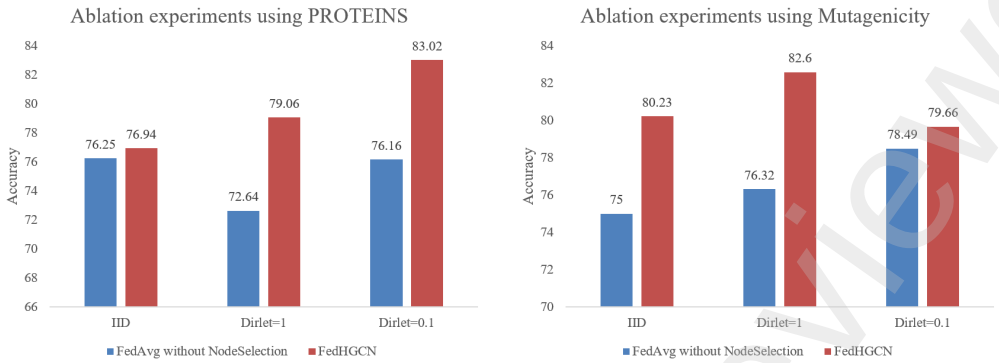


Figure 4: Ablation experiments using PROTEINS and Mutagenicity.

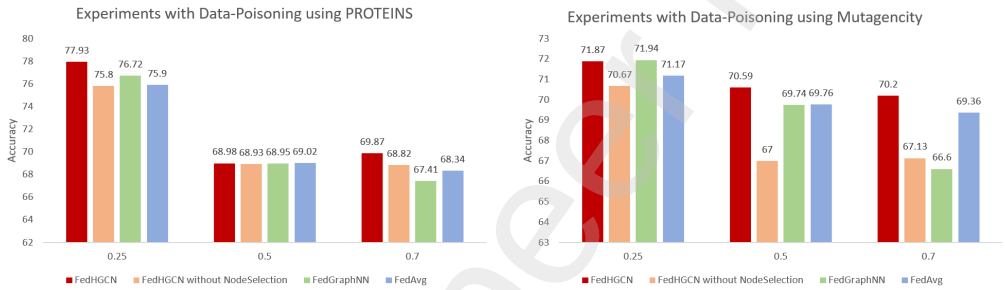


Figure 5: Ablation experiment of data-poisoning attack

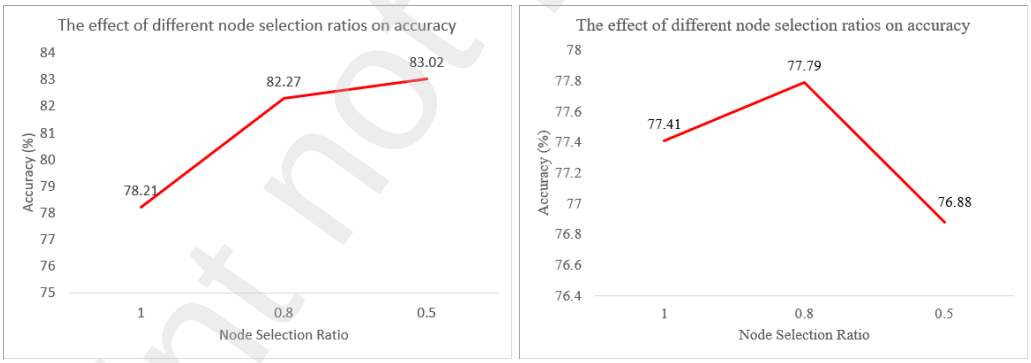


Figure 6: Hyperparameter experiment: The left figure shows the results of the experiment with PROTEINS. The right figure shows the results of the experiment with NCI1.

performance reaches 83.02%, when the node selection ratio is set to 0.5. In the experiments of NCI1, the best performance of accuracy reaches 77.79%, when the node selection ratio is set to 0.8. This suggests that datasets of protein types are more worthy of selecting important nodes. However, in the NCI1 experiments, a

high ratio of node selection does not work well, possibly because the node density of this type of dataset is small and more nodes are affecting the structure of the graph.

6. Conclusion

In this paper, we propose FedHGCN as a novel federated graph classification framework. Through the FedHGCN, each client can more effectively learn about deep hierarchical structure information, increasing the accuracy of graph classification. We employ the node selection and the hyperbolic aggregation techniques to address the problem of decreasing accuracy due to redundancy in graph data. The extensive experimental results show that FedHGCN is more advantageous on both IID and Non-IID experiments. In particular, the accuracy of FedHGCN improved by 15.6% compared to FedAvg when using PROTEINS as the dataset. Moreover, we have also experimented with more clients, and proved that FedHGCN has a certain competitiveness in scalability experiments. Finally, We demonstrate through the ablation experiments that node selection makes FedHGCN more robust.

References

[1] C. W. Zhen Zhang; Jiajun Bu; Martin Ester; Jianfeng Zhang; Chengwei Yao; Zhi Yu, Zhi Yu1, Hierarchical graph pooling with structure learning, in: Lecture Notes in Computer Science, 2019, pp. 287–297.

[2] X. Han, Z. Jiang, N. Liu, X. Hu, G-mixup: Graph data augmentation for graph classification, in: International Conference on Machine Learning, PMLR, 2022, pp. 8230–8248.

[3] H. Guo, Y. Mao, Interpolating graph pair to regularize graph classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, pp. 7766–7774.

[4] P. Rajpurkar, E. Chen, O. Banerjee, E. J. Topol, Ai in health and medicine, Nature medicine 28 (1) (2022) 31–38.

[5] H.-C. Yi, Z.-H. You, D.-S. Huang, C. K. Kwoh, Graph representation learning in bioinformatics: trends, methods and applications, Briefings in Bioinformatics 23 (1) (2022) bbab340.

[6] C. Wu, F. Wu, Y. Cao, Y. Huang, X. Xie, Fedgnn: Federated graph neural network for privacy-preserving recommendation, arXiv preprint arXiv:2102.04925.

[7] C. He, K. Balasubramanian, E. Ceyani, C. Yang, H. Xie, L. Sun, L. He, L. Yang, S. Y. Philip, Y. Rong, et al., Fedgraphnn: A federated learning benchmark system for graph neural networks, arXiv preprint arXiv:2104.07145.

[8] L. X. C. Y. Han Xie, Jing Ma, Federated graph classification over non-iid graphs, in: In NeurIPS, 2021.

[9] B. Wang, A. Li, M. Pang, H. Li, Y. Chen, Graphfl: A federated learning framework for semi-supervised node classification on graphs, in: 2022 IEEE International Conference on Data Mining (ICDM), IEEE Computer Society, 2022, pp. 498–507.

[10] K. Zhang, C. Yang, X. Li, L. Sun, S. M. Yiu, Subgraph federated learning with missing neighbor generation, Advances in Neural Information Processing Systems 34 (2021) 6671–6682.

[11] A. Clauset, C. Moore, M. E. Newman, Hierarchical structure and the prediction of missing links in networks, Nature 453 (7191) (2008) 98–101.

[12] M. Zitnik, R. Sosič, M. W. Feldman, J. Leskovec, Evolution of resilience in protein interactomes across the tree of life, Proceedings of the National Academy of Sciences 116 (10) (2019) 4426–4433.

[13] I. C. R. Y. C. Ré, J. Leskovec, Hyperbolic graph convolutional neural networks, in: In Conference and Workshop on Neural Information Processing Systems, 2019.

[14] G. B. Octavian-Eugen Ganea, T. Hofmann., Hyperbolic neural networks., in: In NeurIPS, 2018.

[15] F. J. R. C. M. Steph-Yves Louis, Alireza Nasiri, J. Hu, Node-select: A graph neural network based on a selective propagation technique, in: Lecture Notes in Computer Science, 2021, pp. 287–297.

[16] G. Mei, Z. Guo, S. Liu, L. Pan, Sgnn: A graph neural network based federated learning approach by hiding structure, in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 2560–2568. doi:10.1109/BigData47090.2019.9005983.

[17] W. Zhu, J. Luo, A. D. White, Federated learning of molecular properties with graph neural networks in a heterogeneous setting, Patterns (2022) 100521.

[18] C. Meng, S. Rambhatla, Y. Liu, Cross-node federated graph neural network for spatio-temporal data modeling, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1202–1211.

[19] D. R. S. H. B. A. H. Brendan McMahan, Eider Moore, y Arcas, Communication-efficient learning of deep networks from decentralized data, In AISTATS.

[20] M. N. Qi Liu, D. Kiela., Hyperbolic graph neural networks., in: In NeurIPS, 2019.

[21] E. U. I. O. Valentin Khrulkov, Leyla Mirvakhabova, V. Lempitsky., Hyperbolic image embeddings., in: In CVPR, 2020.

[22] Y. Z. Haizhou Du, Nostradamus: A novel event propagation prediction approach with spatio-temporal characteristics in non-euclidean space., in: In NN, 2022.

[23] O. Ganea, G. Bécigneul, T. Hofmann, Hyperbolic neural networks, Advances in neural information processing systems 31.

[24] T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, ArXiv abs/1609.02907.

[25] M. K. A. V. Dmitri Krioukov, Fragkiskos Papadopoulos, M. Boguná., Hyperbolic geometry of complex networks., in: In Physical Review E, 2010.

[26] Y. L. H. Z. Z. L. P. L. M. S. J. z. Weize Chen, Xu Han, Fully hyperbolic neural networks., in: In ACL, 2022.

[27] M. Nickel, D. Kiela, Poincaré embeddings for learning hierarchical representations, Advances in neural information processing systems.

[28] J. Fox, S. Rajamanickam, How robust are graph neural networks to structural noise?, arXiv preprint arXiv:1912.10206.

[29] H. Gao, X.; Xiong, P. Frossard, ipool–information-based pooling in hierarchical graph neural networks., 2019.

[30] C. M. N. M. K. F. B. K. K. P. Mutzel, M. Neumann, Tudataset: A collection of benchmark datasets for learning with graphs, in: In ICML 2020 Workshop on Graph Representation Learning and Beyond, 2020.

[31] D. P. D., D. A. J., Distinguishing enzyme structures from nonenzymes without alignments, in: Journal of molecular biology, 2003, pp. 771–783.

[32] N. Wale, I. A. Watson, G. Karypis, Comparison of descriptor spaces for chemical compound retrieval and classification, Knowledge and Information Systems 14 (2008) 347–375.

[33] R. Kazius, J.; McGuire, R. Bursi, Derivation and validation of toxicophores for mutagenicity prediction, in: Journal of medicinal chemistry, 2005, pp. 312–320.

[34] K. N, M. Welling, Semi-supervised classification with graph convolutional networks, in: ICLR, 2017.

[35] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30.

[36] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[37] A. M. G. S. G. K. H. N. Yurochkin, M., Y. Khazaeni, Bayesian nonparametric federated learning of neural networks, in: In International Conference on Machine Learning, pp, 2019, p. 7252–7261.