ASFGNN: Automated Separated-Federated Graph Neural Network

Longfei Zheng
Ant Group, Beijing, China
zlf206411@antgroup.com

Jun Zhou
Ant Group, Beijing, China
jun.zhoujun@antgroup.com

Chaochao Chen[*]
Ant Group, Hangzhou, China
chaochao.ccc@antgroup.com

Bingzhe Wu
Ant Group, Beijing, China
fengyuan.wbz@antgroup.com

Li Wang
Ant Group, Hangzhou, China
raymond.wangl@antgroup.com

Benyu Zhang
Ant Group, Sunnyvale, US
benyu.z@antgroup.com

November 9, 2020

ABSTRACT

Graph Neural Networks (GNNs) have achieved remarkable performance by taking advantage of graph data. The success of GNN models always depends on rich features and adjacent relationships. However, in practice, such data are usually isolated by different data owners (clients) and thus are likely to be Non-Independent and Identically Distributed (Non-IID). Meanwhile, considering the limited network status of data owners, hyper-parameters optimization for collaborative learning approaches is time-consuming in data isolation scenarios. To address these problems, we propose an Automated Separated-Federated Graph Neural Network (ASFGNN) learning paradigm. ASFGNN consists of two main components, i.e., the training of GNN and the tuning of hyper-parameters. Specifically, to solve the data Non-IID problem, we first propose a separated-federated GNN learning model, which decouples the training of GNN into two parts: the message passing part that is done by clients separately, and the loss computing part that is learnt by clients federally. To handle the time-consuming parameter tuning problem, we leverage Bayesian optimization technique to automatically tune the hyper-parameters of all the clients. We conduct experiments on benchmark datasets and the results demonstrate that ASFGNN significantly outperforms the naive federated GNN, in terms of both accuracy and parameter-tuning efficiency.

Keywords Graph neural network · Federated learning · Bayesian optimization · Privacy preserving

1 Introduction

Graph Neural Networks (GNNs) have achieved superior performance by taking advantage of embedding features via aggregating representations of nodes and their neighbors [1]. GNNs benefit a lot of applications across different tasks, such as computer vision [2], traffic prediction [3], recommend system [4] and risk control [5].

1.1 Existing problem

The factor that drives the success of GNN is the rapid growth of high-dimensional data and their adjacent information. However, existing GNN methods face two main challenges. First of all, with the increasing awareness of security and privacy, *data-isolation* problem is serious, which limits the data size of a single party (client) and further damage the performance of GNN. Furthermore, the isolated datasets in different clients are usually Non-Independent and Identically Distributed (Non-IID), due to the reasons that clients belong to diverse geographic locations or have different time

---

[*]Corresponding author.

windows of data collection. Therefore, it becomes more and more difficult to train a global GNN model with the Non-IID data in data isolation scenario.

Fig. 1 shows a typical example of the Non-IID graph data, where we assume there are I separated clients. These clients collect graph data from different sources with the same format. In other words, clients share the same feature domain, e.g., $\{f_1, f_2, f_3\}$, but differ in sample space, which are represented by colorful nodes. Meanwhile, clients may have diverse graph structures of nodes, i.e., heterogeneous graphs. Furthermore, data distributions are likely to be Non-IID, as is shown in Fig. 1.
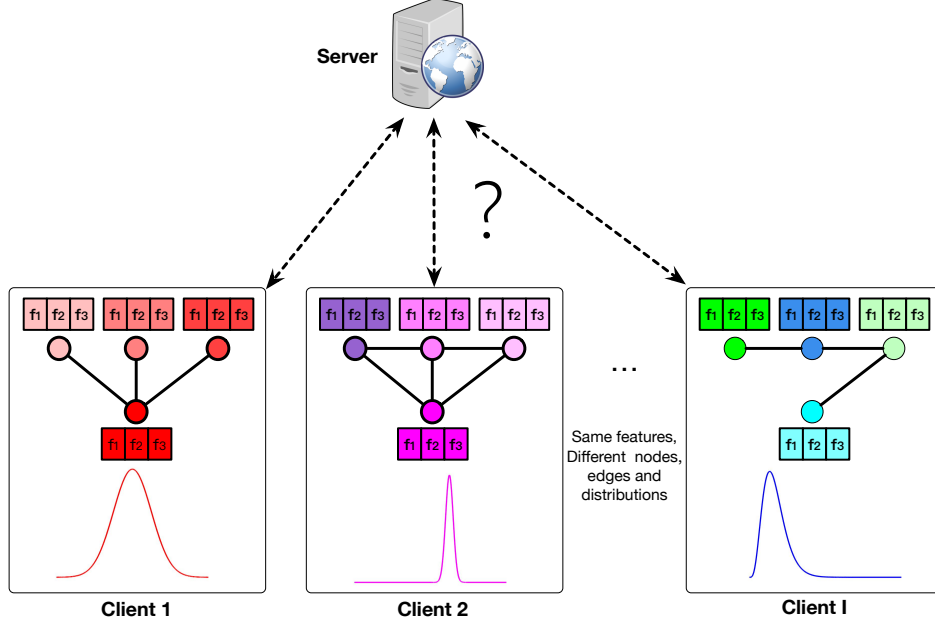

Figure 1: The data isolation problem with Non-IID graph data, assuming I clients with four nodes, three features and different label distributions.

Moreover, hyper-parameters are important for GNN learning algorithms. For example, activation function determines the output of layers, regularization parameter impacts the calculation of loss functions, and learning rate influences the update of model weights in the back-propagation process [6]. These hyper-parameters directly affect the training process of GNN models. Intuitively, in order to achieve the best model performance, clients with Non-IID graph data are likely to have individual hyper-parameter sets rather than a global hyper-parameter set [7]. Due to the huge search space and limited network status among clients, tuning of hyper-parameters is quite time-consuming. Therefore, it is important to design a proper distributed GNN model on Non-IID dataset with hyper-parameter optimization power.

Unfortunately, there is few literature on solving the above problem. Although directly applying federated learning to GNN seems a good choice, it has two main shortcomings [7]. Firstly, federated learning faces the statistical challenge. The original goal of federated learning, i.e., training a single global model on the union of clients' datasets, is no longer suitable for Non-IID graph data [8]. Secondly, communication of federated GNN learning is time-consuming. This is because, in order to achieve the best performance, models and hyper-parameters of clients are likely to be different. Comparing with the traditional neural network, GNN has extra individual hyper-parameters to represent graph information, which further increases the unbearable training time.

1.2 Our Solution

In order to bridge these gaps, we propose an Auto Separated-Federated GNN (ASFGNN) learning paradigm. As graph data is often owned by companies and governments, we focus on the cross-silo federated learning in which the clients are a limited number of organizations with powerful computing ability and reliable communications [7]. Our proposed ASFGNN consists of two steps, i.e., GNN training and hyper-parameters optimization.

In the first step, the Separated-Federated GNN learning framework decouples a GNN model into two parts: *message passing* sub-model that is conducted by clients separately and *loss computing* sub-model which is performed by clients federally. Specifically, clients first perform message passing, i.e., neighbor information aggregation, individually, and

get node embeddings. In the following step, clients take the embeddings as the input of the discrimination model to compute loss, then update both message passing sub-model and loss computing sub-model using backward propagation for the first time. After it, the server securely aggregates the local discrimination models using federated learning method and gets the global discrimination model. Finally, the global discrimination model is broadcast to clients to update the local discrimination models with the help of Jensen–Shannon divergence.

In the second step, we propose a *Bayesian optimization algorithm* to automatically optimize the hyper-parameters of Separated-Federated GNN model. That is, Bayesian optimization algorithm takes hyper-parameters as input and regards the average value of clients' evaluation metrics (e.g., precision) as output [9, 10], where these metrics are uploaded by clients and averaged by server in a secure manner. To this end, we get the hyper-parameters that achieve the best metric.

To verify the performance of our proposed ASFGNN, we empirically compare the accuracy of SFGNN and traditional federated GNN model, and analyze the efficiency of Bayesian optimization method and the traditional grid search method.

We summarize our main contributions as follows:

- We propose a novel Separated-Federated Graph Neural Network (SFGNN) learning framework, which can be used to learn any existing GNN models under privacy consideration.

- We propose to adopt Bayesian optimization to tune model parameters automatically, which significantly improves the efficiency of the SFGNN model.

- We conduct experiments on three benchmark datasets and the results demonstrate that our proposed SFGNN outperforms federated GNN in terms of accuracy, and ASFGNN significantly reduces the hyper-parameter tuning time of SFGNN comparing with grid search.

2 Related work

In this section, we briefly review the literature on federated learning and hyper-parameters optimization.

2.1 Federated learning

Federated learning model is prevailing privacy-preserving approach via model or gradient aggregation rather than data aggregation [11]. However, the accuracy of federated learning would drop significantly with Non-IID datasets[7]. Existing works propose different strategies to resolve the statistical challenge of federated learning. One natural approach is to create a small shared dataset which makes the data across clients more similar [12]. For some applications, the contributions of clients to the global model are bounded according to the dataset characteristics [13]. Furthermore, model-agnostic meta-learning has been developed to meta-learn a global model, which can be used as a starting point for learning a good model of Non-IID data in each client [14]. These methods modify federated learning model with Non-IID datasets, which can not be applied in GNN model directly. As GNN model includes two parts as shown in preliminary, among which the message passing part owns personal information which should be learned individually.

Besides the federated learning, Split Learning (SL) is another decentralized method which trains the local models separately and sends hidden layers to server [15]. The separated local models represent the personality of clients with Non-IID datasets [16]. However, it is obviously that the hidden layers leak privacy information and the deep local layers decrease the accuracy seriously [17]. In this paper, we combine the advantages of federated learning and split learning, and propose a novel Separated-Federated Graph Neural Network learning framework.

2.2 Hyper-parameters optimization

Recently, there has been an increasing literature on hyper-parameters optimization [10]. Grid search is the most traditional way of hyper-parameters tuning, which enumerates every possible configuration in the search space. Random search is better than naive grid search, which samples configurations randomly. Moreover, Evolutionary Algorithm (EA) and Reinforcement Learning (RL) methods are used to generate a new population (a bunch of configurations). Another conventional solution resorts to formalizing machine learning process as a black-box optimization task, reference [18] finds the optimal of black-box objectives with the method of Bayesian Optimization (BO). Comparing with EA and RL, BO is more efficient than these methods and shows promising results in hyper-parameters optimization [10]. In this paper, we propose to apply BO as a prevailing approach to find the proper hyper-parameters in our proposed model.

3 Preliminaries

In this section, we present some preliminary techniques and methods of our proposal, including Graph Neural Network (GNN), federated learning, secret sharing, Jensen-Shannon divergence, and Bayesian optimization.

3.1 Graph neural network

GNN learns node embeddings by aggregating features of node and its neighbors. The node embeddings are regarded as the new node representations which are fed to downstream machine learning tasks. The process of GNN training includes two steps: message passing and loss computing. The first step is the difference between GNN and other neural network models, which uses a generation function to infer node embeddings. Numbers of message passing functions have been proposed, e.g., random walk statistics based, attention based, similarity based, and convolution based [19, 20, 21, 3]. In this work, we select GraphSAGE as the node embeddings generation function, which aggregates the embeddings from a node's local neighborhood in a inductive way [22]. The message passing process is described in Equation (1), where $k \in \{1, 2, ..., K\}$ denotes the depth of neighborhood aggregation, $\mathbf{h}_{v,k}$ denotes the embedding of node v during k-th aggregation step, $\mathbf{x}_v$ denotes features of node v and $\mathcal{N}(v)$ denotes the neighbors of node v in the graph [23].

$$
\begin{aligned}
\mathbf{h}_{\mathcal{N}(v),k} &\leftarrow \mathrm{AGG}_k(\{\mathbf{h}_{u,k-1}, \forall u \in \mathcal{N}(v)\}), \\
\mathbf{h}_{v,k} &\leftarrow \left(\mathbf{W}_k \cdot \mathrm{CONCAT}\left(\mathbf{h}_{v,k-1}, \mathbf{h}_{\mathcal{N}(v),k}\right)\right).
\end{aligned}
\tag{1}
$$

where AGG_k is the aggregation function in k-th step, such as Mean, LSTM, and Pooling methods [22].

3.2 Federated learning

Federated Learning (FL) was first proposed by Google [24], which builds distributed machine learning models while keeping personal data on clients. In other words, federated learning models are trained via model aggregation rather than data aggregation. We suppose that I clients have their own datasets $\{D_1, D_2, ..., D_I\}$ which are collected from different sources with the same feature domain. Private raw dataset D_i is preserved locally, client i uses forward and backward propagations to update its own model M_i individually, which has the identical neural network architecture with other clients. Then clients upload the encrypted weights to the server with the help of secret sharing or homomorphic encryption [25, 26, 27, 28]. The server averages the uploaded model parameters to update the global federated model M_s, which will be sent back to client i to replace the local model M_i.

3.3 Jensen-Shannon divergence

The Jensen–Shannon divergence (JS) is popularly used to evaluate the dissimilarity between two probability distributions [29]. JS has a finite value range from 0 to 1 for two probability distributions. Motivated by [30], JS can be used to indicate the dissimilarity between two None-IID datasets. Considering two probability distributions P and Q, the JS between P and Q is defined in Equation (2).

$$
\begin{aligned}
\mathbf{JS}\left(P||Q\right) &\leftarrow \frac{1}{2}\mathbf{KL}\left(P||\frac{P+Q}{2}\right) + \frac{1}{2}\mathbf{KL}\left(Q||\frac{P+Q}{2}\right), \\
\mathbf{KL}\left(P_1||P_2\right) &\leftarrow \sum_{x \in X} P_1(x) \log \frac{P_1(x)}{P_2(x)}.
\end{aligned}
\tag{2}
$$

As the machine learning model is built to represents the trained dataset, the difference between the aggregated model in server and the local model in client can be simulated by the distribution similarity between the participated data and the client data.

3.4 Secret sharing

Our proposal depends on Shamir's t-out-of-n threshold secret sharing algorithm [25]. Typically, we use n-out-of-n additive secret sharing to recover the privacy in this paper. For example, we suppose that there is an ℓ-bit value a of client i, $i \in \mathcal{P}$ with $\mathcal{P} = \{1, ..., I\}$, which will be shared among all the participant clients. Firstly, in order to encrypt ($\mathbf{Shr}(\cdot)$) the value a of client i, client i generates a random number a_j, $\{a_j \in \mathcal{Z}_{2^\ell}, j \in \mathcal{P}, j \neq i\}$, which will be distributed to client j, $\{j \in \mathcal{P}, j \neq i\}$. Then client i calculates $a_i = a - \sum_j a_j \bmod 2^\ell$ which will be kept locally. For simplification, We use $\langle a \rangle_k$ to denote the share of a in client k, $\forall k \in \mathcal{P}$. To decrypt ($\mathbf{Rec}(\cdot)$) the shared value a, client k ($\forall k \in \mathcal{P}$) sends the encrypted value $\langle a \rangle_k$ to the server. The server aggregates $\sum_k \langle a \rangle_k \bmod 2^\ell$, $k \in \mathcal{P}$, and gets the value a of client i.

(a) Separated federated graph neural network (SFGNN) model
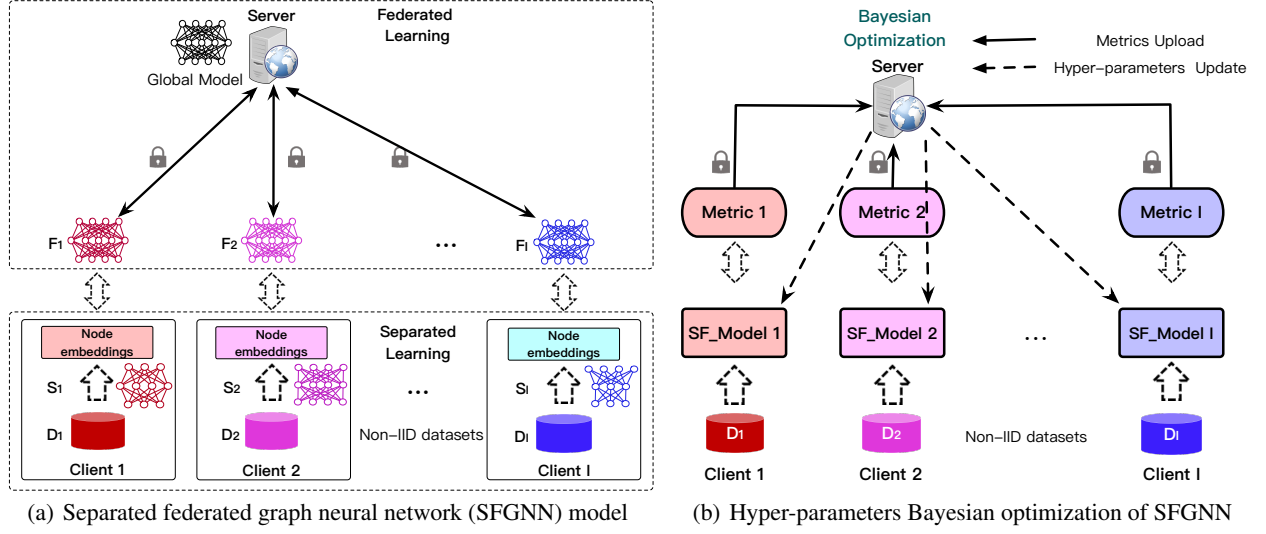
(b) Hyper-parameters Bayesian optimization of SFGNN

Figure 2: Our proposed automated separated-federated graph neural network model.

3.5 Bayesian optimization

Bayesian Optimization (BO) is an effective method to solve the black-box parameter optimization problem [10]. In our paper, we care about the hyper-parameter optimization in the training of GNN model, where we try to find the optimal hyper-parameter setting that maximizes the utility function:

$$\theta^* = \underset{\theta \in \Theta}{\arg\max} f(\theta), \tag{3}$$

where θ denotes the hyper-parameters, such as learning rate and dimension of hidden units. The Θ denotes the search space and f is the utility function which is measured by certain model metrics, such as model accuracy and the Area Under Curve (AUC) score. Typically, the evaluation of f is expensive and we cannot obtain its closed-form solution. Therefore, we treat Equation (3) as a black-box optimization and adopt BO to solve this problem. The key ingredients of BO include a surrogate model to "imitate" f and an acquisition function to decide the next trial based on historical trails (i.e., hyper-parameters). In our paper, we use Gaussian process (GP) as our surrogate model and use the Expected Improvement (EI) function as the acquisition function [31].

4 The proposed method

In this section, we first give an overview of the proposed Automated Separated-Federated Graph Neural Network (AS-FGNN) learning paradigm. We then present its three main components, i.e., separated learning for message passing on clients, federated learning for loss computing with Jensen–Shannon divergence, and hyper-parameters optimization with Bayesian optimization. Finally, we summarize the whole algorithm.

4.1 Overview

We first give an overview of the proposed ASFGNN learning framework. We focus on horizontally split datasets in this paper.

Our design of ASFGNN consists of two steps. First, we need to design a privacy preserving GNN learning model, which can solve the Non-IID problem and reduce the communication cost as much as possible. Second, since GNN has many hyper-parameters, we need to design a strategy to automatically optimize hyper-parameters to reduce the training time.

The first step is to design a practical GNN learning paradigm without leaking the private plaintext data of clients. Inspired by existing works [32, 33], we propose a Separated-Federated GNN (**SFGNN**) learning framework. The main idea is decoupling the computation module of GNN into two sub-modules, i.e., the Separated GNN learning (**SGNN**) model and the Federated GNN learning (**FGNN**) model, as shown in Fig. 2 (a). The former performs message passing and obtains the node embeddings as inputs of the latter one. As clients have Non-IID datasets, node embeddings

5

Table 1: Notations and descriptions.

Notation	Description	Notation	Description
I	total number of clients	$\mathcal{P}$	union set of I clients
G^i	graph data of client i	V^i	nodes data of client i
E^i	edges data of client i	$\mathcal{N}^i(v)$	neighbour function of client i
$\mathbf{x}_v^i$	features of node v in client i	J	total number of categories
$W_{k,t}^i$	weights of k-th step in SGNN for client i during t-th epoch	K	depth of neighbor aggregation in SGNN
$h_{v,k}^i$	intermediate node embeddings of node v in client i during k-th step	$H_{v,t}^i$	the final node embeddings of node v in client i during t-th epoch
Q_t^i	probability density of label in client i during t-th epoch	Q_t^s	probability density of label in server during t-th epoch
N_t^i	sample numbers of different categories in client i during t-th epoch	N_t^s	sample numbers of different categories in server during t-th epoch
$n_{j,t}^i$	sample numbers of category j in client i during t-th epoch	n_t^i	total number of samples in client i during t-th epoch
y_t^i	labels in client i	$\hat{y}_t^i$	labels prediction in client i
$W_{l,t}^i$	intermediate weights of l-th layer in FGNN of client i during t-th epoch	$\overline{W}_{l,t}^i$	weights of l-th layer in FGNN of client i during t-th epoch
$\overline{W}_{l,t}^s$	weights of l-th layer in FGNN model of server during t-th epoch	js_t^i	JS divergence between dataset of client i and dataset of server during t-th epoch
$\mathbf{Shr}(\cdot)$	additively secret sharing encrypt	$\mathbf{Rec}(\cdot)$	additively secret sharing decrypt
$\langle\cdot\rangle$	encryption using secret sharing	L	number of layers in FGNN model
M_t	average of metrics during t-th epoch	M_t^i	metric of client i during t-th epoch
lr_n^i	learning rate of client i in n-th BO round	$l2_n^i$	L2 regularization of client i in n-th BO round
θ_n	hyper-parameters set in n-th BO round	$M(\cdot)$	black-box function of hyper-parameters optimization

are generated separately with individual network architecture and hyper-parameters. After the generation of node embeddings with SGNN, FGNN trains the discrimination neural network taking advantage of federated learning algorithm.

Secondly, hyper-parameters of SFGNN, such as learning rate, regularization factor, network structures etc., explode with the increasing number of clients. We adopt Bayesian Optimization method to solve this black-box optimization problem, in which we regard the hyper-parameters of model as inputs and the average of clients' metrics as outputs, as shown in Fig. 2 (b). The metrics of SFGNN model in clients are securely aggregated in server. Then the server optimizes the hyper-parameters and sends the hyper-parameters back to clients to finish another training epoch of SFGNN. To the end, the whole parameter-tuning time is greatly decreased, as the searching round of hyper-parameters is highly reduced.

In summary, we leverage Bayesian optimization technique to automatically tune the hyper-parameters of SFGNN model, combining SGNN with FGNN. **Notations.** Before presenting our model in details, we first describe the notations. Considering there are many notations, for clarity, we summarize the notations used in this paper in Table 1.

4.2 Separated GNN learning (SGNN)

We summarize how to generate initial node embeddings for client $i(i \in \mathcal{P})$ using GraphSAGE method [22] in SGNN Algorithm 1, where the entire graph $G^i = (V^i, E^i)$, features for all nodes $\mathbf{x}_v^i$ $(\forall v \in V^i)$ are provided as inputs. The weight matrix $W_k^i, \forall k \in \{1, ..., K\}$ are used to propagate information of message passing layers. The first step is generating initial node embeddings using nodes' private features, e.g., user features in social networks (line 2). In the next step, clients generate local node embeddings by aggregating multi-hop neighbors' information using GraphSAGE method [22] for the FGNN computations as shown in line 4-15 in Algorithm 1.

Algorithm 1: SGNN (Separated GNN learning on client)

Input: Graph $G(V^i, E^i)$ and node features $\{\mathbf{x}_v^i, \forall v \in V^i\}$ on data holder $i, i \in \mathcal{P}$; depth K; non-linearity function
$\quad\quad\sigma$; neighborhood functions $\mathcal{N}^i(v) : v \to 2^{V^i}, \forall i \in \mathcal{P}$

Output: Node embeddings:$H_{v,t}^i, \forall v \in V^i$ on client i during $t-th$ training round

1 # Calculate the initial node embeddings
2 $h_{v,0}^i \leftarrow \mathbf{x}_v^i \cdot W_0^i, \forall i \in \mathcal{P}, \forall v \in V^i$
3 # Generate local node embeddings
4 **for** *each round $t = 1, 2, ..., T$* **do**
5 $\quad$ **for** $i \in \mathcal{P}$ *in parallel* **do**
6 $\quad\quad$ **for** $k = 1$ *to K* **do**
7 $\quad\quad\quad$ **for** $v \in V$ **do**
8 $\quad\quad\quad\quad$ client i: calculates $h_{v,k}^{\mathcal{N}^i(v)} \leftarrow \text{Mean}\left(\left\{h_{u,k-1}^i, \forall u \in \mathcal{N}^i(v)\right\}\right)$
9 $\quad\quad\quad$ **end**
10 $\quad\quad\quad$ client i: calculates $h_{v,k}^i \leftarrow \sigma\left(W_{k,t}^i \cdot CONCAT\left(h_{v,k-1}^i, h_{v,k}^{\mathcal{N}^i(v)}\right)\right)$
11 $\quad\quad$ **end**
12 $\quad\quad$ **Client** i: calculates $H_{v,t}^i \leftarrow h_{v,K}^i / ||h_{v,K}^i||_2, \forall v \in V^i$
13 $\quad$ **end**
14 $\quad$ **return** Node embeddings $H_{v,t}^i, \forall i \in \mathcal{P}$
15 **end**

4.3 <mark>Federated GNN learning (FGNN)</mark>

First of all, client i ($\forall i \in \mathcal{P}$) randomly initializes weights of Federated GNN Learning model $\overline{W}_{l,0}^i, l \in \{1, ..., L\}$ with L denoting the max layer. Client i gets the label distribution Q_t^i ($Q_t^i = \{q_{t,1}^i, q_{t,2}^i, ..., q_{t,J}^i\}$) in the current batch during training epoch t with n_t^i samples, where J denotes the label classification as shown in FGNN Algorithm 2. Then client i counts sample numbers of different categories $N_t^i = \{n_{t,1}^i, n_{t,2}^i, ..., n_{t,J}^i\}$, where $\sum_{j=1}^J n_{t,j}^i = n_t^i$ (line 4). Meanwhile client i updates local FGNN model's weights $W_{k,t}^i$ and $W_{l,t}^i$ using forward and backward propagation with their own embeddings $H_{v,t}^i$ generated by Algorithm 1 (lines 5-11). Loss function $L(\hat{y_t}^i, y_t^i)$ is defined by different tasks, e.g., cross entropy loss for classification task and mean absolute loss for regression task. In this paper, we choose classification task for example, the loss of which is defined in Equation (4).

$$L(\hat{y_t}^i, y_t^i) = -\frac{1}{n_t^i}\sum_{j=1}^J \hat{y}_{j,t}^i \log^{y_{j,t}^i} + l_2{_n^i} \cdot \left(\sum_{k=0}^K ||W_{k,t}^i||_2 + \sum_{l=1}^L ||\overline{W}_{l,t}^i||_2\right). \tag{4}$$

After it, $W_{l,t}^i$, N_t^i, and M_t^i ($i \in \mathcal{P}, l \in \{1, ..., L\}$) of clients are uploaded to server with the help of secret sharing (**Shr**$(\cdot)$), supposing all clients participate in the federated learning, as shown in line 13. The server aggregates the global FGNN model $\overline{W}_{l,t}^s$ by averaging the sum of $W_{l,t}^i$, and gets the global label distribution Q_t^s, sample numbers N_t^s of a training batch and average of metrics M_t, all of which are regarded as outputs of FGNN model, as shown in Algorithm 2 line 16-19. Then $\overline{W}_{l,t}^s$ and Q_t^s are sent back to clients. To the end, client i calculates js_t^i with the help of Q_t^i and Q_t^s, then the local FGNN model is updated by combining $\overline{W}_{l,t}^s$ and $W_{l,t}^i$ (line 24). <mark>js_t^i controls the percent of the client local model in update process. The more Non-IID clients datasets are, the bigger priority of client model is. In a world, the addition of JS contributes to the accuracy of client model in Non-IID federated learning.</mark>

4.4 <mark>Hyper-parameters optimization</mark>

We employ Bayesian optimization in tuning hyper-parameters, where we treat the hyper-parameter search process as a black-box optimization, as shown in Equation (3). Specifically, the hyper-parameter set θ_n includes dropout rate, L2 regularization, propagation depth, learning rate, and dimension of hidden units. The utility function f is set to be the average of clients' accuracy. The high-level optimization process is shown in Algorithm 3. Firstly, we update the posterior probability distribution on f using all the hyper-parameters sets(line 5). Then we calculate the maximize point of the EI acquisition function as the next hyper-parameters groups and observe the value of utility function (line 6 - line 7). The hyper-parameter tuning time is measured by $T = n * t$, where t denotes the running time of one set of

---

Algorithm 2: FGNN (Federated GNN learning)

Input: Node embeddings $H_{v,t}^i, \forall v \in V, \forall i \in \mathcal{P}$; hyper-parameters set θ_n

Output: M_t

1   # Client model update

2   Randomly initialization $\overline{W}_{l,0}^i, \forall i \in \mathcal{P}, \forall l \in \{1, ..., L\}$

3   **for** *each round $t = 1, 2, ..., T$* **do**

4   # Updates local FGNN model's weights and sends to server

5   **for** $i \in \mathcal{P}$ `in parallel` **do**

6   # Get: Q_t^i, N_t^i

7         $H_{v,t}^i \leftarrow \textbf{SGNN}\left(G(V^i, E^i), \mathcal{N}^i(v), \mathbf{x}_v^i\right)$

8         $\hat{y}_t^i \leftarrow \sigma\left(H_{v,t}^i \cdot \overline{W}_{l,t}^i\right)$

9   # Get: $L(\hat{y}_t^i, y_t^i), M_t^i$

10        $W_{k,t}^i \leftarrow W_{k,t-1}^i - lr_m^i \nabla L\left(\hat{y}_t^{\,i}, y_t^i\right)$

11        $W_{l,t}^i \leftarrow \overline{W}_{l,t-1}^i - lr_m^i \nabla L\left(\hat{y}_t^{\,i}, y_t^i\right)$

12   # Upload privacy information using secret sharing:

13        $\langle W_{l,t}^i \rangle, \langle N_t^i \rangle, \langle M_t^i \rangle \leftarrow \textbf{Shr}(W_{l,t}^i), \textbf{Shr}(N_t^i), \textbf{Shr}(M_t^i)$, upload to server

14   **end**

15   # Secure aggregation in server:

16      $\overline{W}_{l,t}^s \leftarrow \frac{1}{I}\left(\sum_{i=1}^I \langle W_{l,t}^i \rangle\right)$

17      $N_t^s \leftarrow \sum_{i=1}^I \langle N_t^i \rangle$

18      $Q_t^s \leftarrow \frac{N_t^s}{\sum_J N_t^s}$

19      $M_t = \frac{1}{I}\sum_{i=1}^I \langle M_t^i \rangle$

20   # Send $\overline{W}_{l,t}^s, Q_t^s$ to client $i, \forall i \in \mathcal{P}$

21   # Update $\overline{W}_{l,t}^i$ in client:

22   **for** $i \in \mathcal{P}$ `in parallel` **do**

23         $js_t^i \leftarrow JS\left(Q_t^i \| Q_t^s\right)$

24         $\overline{W}_{l,t}^i \leftarrow js_t^i \cdot \overline{W}_{l,t}^i + \left(1 - js_t^i\right) \cdot W_{l,t}^s$

25   **end**

26   **return** M_t

27   **end**

---

hyper-parameters, n denotes the number of hyper-parameter combinations, and T is the total hyper-parameter tuning time. Bayesian optimization optimizes the hyper-parameter tuning time by narrowing down the number of combinations n greatly.

4.5 Putting all together

To sum up, we conclude the ASFGNN framework in the Algorithm 4. Before the training process, we initialize the hyper-parameters set of clients and server as θ_0. First of all, we get the node embeddings $H_{v,t}^i$ for each client i using Algorithm 1 (SGNN) with the relevant hyper-parameters set θ_n (line 5). Secondly, we start the training of FGNN model using node embeddings as the inputs and get the average of accuracy (M_t) in each training round (line 7). The max of M_t is marked as $M(\theta_n)$ (line 9), which is regarded as outputs of black-box. Then, the following input θ_{n+1} is updated by Bayesian optimization. Finally, we get the best hyper-parameters set θ_N and the relevant $M(\theta_N)$.

5 Experiment

In this section, we empirically compare the performance of our proposed ASFGNN model with the GraphSAGE of Centralized Model (CM) which is trained using all the data, the traditional Federated Learning model (FL) and the Separated model (SP) in which clients can only use their own data without any communications. We aim to answer the following questions.

Algorithm 3: Hyper-parameters optimization

---
1 Place a Gaussian process prior on f
2 Observe f at n_0 hyper-parameters groups according to an initial space-filling experimental design
3 Set $n = n_0$
4 while $n \leq N$ **do**
5   Update the posterior probability distribution on f using all available data
6   Let θ_n be a maximize point of the EI acquisition function over Θ, where the acquisition function is computed
   using the current posterior distribution
7   Observe $y_n = f(\theta_n)$.
8   Increment n
9 end

Algorithm 4: ASFGNN (Automated separated-federated graph neural network learning)

---
1 Initialization of hyper-parameters set: θ_0
2 for *each BO round* $n = 0, 1, ..., N$ **do**
3   **for** *each FGNN round* $t = 1, 2, ..., T$ **do**
4   **for** *each client* $i \in \mathcal{P}$ **do**
5 $H^i_{v,t} \leftarrow SGNN(V^i, E^i, \theta_n)$
6   **end**
7 $M_t \leftarrow FGNN(H^i_{v,t}, \theta_n)$
8   **end**
9 $M(\theta_n) \leftarrow \text{MAX}(M_t)$
10   Update $\theta_{n+1} \leftarrow \arg\max_{\theta \in \Theta} M(\theta)$
11 end
12 return $\theta_N, M(\theta_N)$

- **Q1:** whether our model (SFGNN) outperforms the CM model, FL model and SP model that is trained on the isolated Non-IID data, including Non-IID label and Non-IID graph?

- **Q2:** how the distribution parameter influences the performance of our model?

- **Q3:** how the number of clients influences the performance of our model?

- **Q4:** how the JS divergence influences the performance of our model?

- **Q5:** how does Bayesian optimization affect the efficiency of parameter tuning comparing with grid search?

5.1 Experimental settings

5.1.1 Framework

We construct our experiment on the popular TensorFlow framework [34]. All the experiments were performed on a Macbook Pro laptop with 2.3GHz 4-core Intel Core i5 processor. For simplification, we ignore the communication cost and focus on the performance and computation efficiency.

5.1.2 Datasets

To test the effectiveness of our proposed model, we choose three benchmark citation datasets, i.e., Cora, Pubmed, and Citeseer. For simplification, we assume there are only two clients ($\mathcal{A}$ and $\mathcal{B}$) who split datasets according to label classes and number of neighbours in graph. We use N_1 and N_2 to denote the number of samples in each part. We divide Cora dataset into Co_1 and Co_2. The first part Co_1 has four label categories (theory, reinforcement learning, genetic algorithms, and probabilistic methods) with 1,412 nodes. The second part Co_2 contains the rest three label categories (possessing neural networks, case based and rule learning labels) with 1,296 nodes. We also divide Citeseer and Pubmed datasets into two parts (Ci_1 and Ci_2, Pu_1 and Pu_2) in a similar way. We report the data split result in Table 2. In order to study the influence of data Non-IID on our method, we use α to denote the label distribution ratio. The data of client $\mathcal{A}$ is made up of $\alpha \cdot N_1$ samples from the first part and $(1 - \alpha) \cdot N_1$ samples from the second part. Similarly, the data of client $\mathcal{B}$ is made up of $(1 - \alpha) \cdot N_2$ samples from the first part and $\alpha \cdot N_2$ samples from the second part. In other words, the hyper-parameters α implies the non-iid level. We assume that α ranges from 0.5 to 1.0 due

Table 2: Statistic analysis of subsets.

Subset	#Nodes	#Edges	#Features	#Classes
Co_1	1,412	2,657	1,433	4
Co_2	1,296	1,961	1,433	3
Ci_1	1,507	2,024	3,703	3
Ci_2	1,805	2,005	3,703	3
Pu_1	9,791	16,585	500	2
Pu_2	9,926	19,020	500	2

Table 3: Performance comparison on three datasets in terms of accuracy.

Dataset	CM	FL	SP	SFGNN
Cora	0.8150	0.8833	0.9101	**0.9264**
Citeseer	0.7001	0.7500	0.7823	**0.8055**
Pubmed	0.7910	0.8889	0.9174	**0.9340**
Average	0.7687	0.8407	0.8699	**0.8886**

to the symmetry. We use exactly the same dataset split of training, validate, and test following the prior work [35]. Apparently, our proposal can be applied into the scenario where there are multiple clients.

5.1.3 Metrics

Following the existing work [35], we use accuracy as the evaluation metric. To compare the performance of different strategies in decentralized scenario, we choose the average of metrics in all clients as the optimization target.

5.1.4 Hyper-parameters

Following recent research [36], we use hyperbolic tangent (TanH) as the active function of hidden layers and set the max layer of the fully-connected deep neural network in the discrimination model ($L = 2$). We tune other hyper-parameters by using Bayesian optimization. The hyper-parameters include dropout rate $d \in \{0.0, 0.5\}$, L2 regularization $l_2 \in \{0.0, 5e^{-4}, 1e^{-3}, 5e^{-3}, 1e^{-2}\}$, propagation depth $K \in \{1, 2, 3, 4, 5\}$, learning rate $l_r \in \{5e^{-4}, 1e^{-3}, 5e^{-3}, 1e^{-2}\}$, and dimension of hidden units $l \in \{64, 128, 256, 512\}$. As clients train the discrimination model federated, the dimension of embeddings should be aligned, which means that all clients have the same hidden units dimension. The experiment are conducted in a stand-alone PC to simulate the communication in federated learning. We tune parameters based on the validate dataset and evaluate model performance on the test dataset.

5.2 Accuracy comparison

5.2.1 Accuracy comparison of different models with Non-IID label

To answer the proposed question **Q1**, we first set the label distribution ratio $\alpha = 1.0$, which implies the labels between client $\mathcal{A}$ and client $\mathcal{B}$ are totally different. In general, we take advantage of grid search method to find the highest accuracy with the proper hyper-parameters. We summarize the comparison results in Table 3, and report the corresponding best hyper-parameters set in Table 4.

Table 4: Hyper-parameters of the SFGNN model and FL model with the best accuracy.

Model	K	lr	l_2	d
FL of Cora	4	0.01	0.005	0.0
SFGNN of $\mathcal{A}$ on Cora	4	0.01	0.005	0.5
SFGNN of $\mathcal{B}$ on Cora	2	0.01	0.005	0.5
FL on Citeseer	4	0.005	0.005	0.0
SFGNN of $\mathcal{A}$ on Citeseer	4	0.005	0.01	0.5
SFGNN of $\mathcal{B}$ on Citeseer	4	0.01	0.01	0.0
FL on Pubmed	5	0.005	0.001	0.5
SFGNN of $\mathcal{A}$ on Pubmed	3	0.01	0.001	0.0
SFGNN of $\mathcal{B}$ on Pubmed	2	0.01	0.005	0.5

Table 5: Performance comparison on both Non-IID label and Non-IID graph.

Dataset	FL	SFGNN	Improvement
Cora	0.7525	0.7986	6.13%
Citeseer	0.7020	0.7583	8.02%
Average	0.7273	0.7785	7.04%

From the Table 3, we can conclude that SFGNN outperforms the other three models in all the datasets. Besides, comparing with the traditional FL model, the improvement of accuracy is about 5.70% percent in average, which means the SFGNN model is more effective for data Non-IID scenarios. Because our proposed SFGNN generates embeddings separately with individual hyper-parameters and aggregates discrimination layers of clients, the SFGNN model can balance the inference and contributions from samples with different labels. From Table 3, we can also find an interesting result. That is, the Centralized GNN Model (CM) achieves the worst performance. This is because clients have absolutely different label classes when the $\alpha = 1.0$, and the models with relatively pure label classes will naturally achieve better performance. When different label classes are combined together in CM, it introduces distractions to the model learning target, which makes CM behave the worst.

From Table 4, we can also observe that clients generally have different hyper-parameters to achieve the best accuracy, and these parameters are also different from the hyper-parameters of FL model. The individual hyper-parameters describe the diversity of Non-IID datasets.

5.2.2 Accuracy comparison of different models with both Non-IID label and Non-IID graph

he GNN model benefits a lot from adjacent information, which is different from the traditional neural network model, the distribution of graph data also has an important influence on model accuracy. As median of edges in Cora dataset is 3.8, we firstly split the Cora dataset into two sub-datasets, i.e., Co_3 and Co_4, according to the average edges. The Co_3 has the samples with equal or lesser than 3 edges, while Co_4 containts the rest samples. Furthermore, we combine the Non-IID graph data with the Non-IID label data, which means the datasets have different graph and label distributions. Similar as the setting in Table 2, we build a subset of Co_3 as Co_5, which only has label classes of 'theory', 'reinforcement learning', 'genetic algorithms', and 'probabilistic methods', and a subset of Co_4 as Co_6, which only has labels of 'possessing neural networks', 'case based' and 'rule learning'. Similarly, we get the subsets of Citeseer dataset (Ci_3, Ci_4, Ci_5, Ci_6). After the data being preprocessed, client $\mathcal{A}$ owns the Co_5 subset and client $\mathcal{B}$ owns the Co_6 subset of Cora dataset, and client $\mathcal{A}$ owns the Ci_5 subset and client $\mathcal{B}$ owns the Ci_6 subset of Citeseer dataset. We train the SFGNN model and FL model respectively and compare their accuracy in Table 5. We can conclude that the SFGNN model performs better than FL model when both label and graph are Non-IID, and the improvement percent average increases from 5.70% to 7.04%. The experiment results indicate that SFGNN model is more appropriate for the scenarios where both graph and label have different distributions.

5.2.3 Accuracy comparison with different label distribution

To answer the proposed question **Q2**, we vary α from 0.6 to 1.0 on Cora dataset and report the accuracy of different models in Fig. 3. We compare our model with the SP model in which client $\mathcal{A}$ and client $\mathcal{B}$ can only use their own data without any communications. The bigger ratio means the less similar distributions of clients' datasets. From the results, we can conclude that (1) SFGNN performs better than both SP model and FL model when data distribution is more asymmetrical ($\alpha > 0.75$), which is a quite common situation in real-world applications, (2) FL model is more suitable for training a single global model when all the clients tend to have IID data ($0.5 <= \alpha < 0.75$), and (3) SP model even works better than FL when clients have severely heterogeneous label distributions ($0.88 < \alpha <= 1.0$).

5.2.4 Accuracy comparison of different clients' number with Non-IID label.

The CM model is trained by the whole dataset, which can be regarded as the ASFGNN model with only one client. To answer the proposed question **Q3**, we vary the number of clients from 2 to 5 on Cora dataset. The labels of clients' data are different from each other. We report the average accuracy of ASFGNN in Fig. 4. From it, we can find that the average accuracy of ASFGNN first increases with the number of clients, and then tends to be stable. This is because, when the number of clients first increases, each client has fewer kinds of labels, which makes the Non-IID problem more serious. Therefore, our proposed ASFGNN achieves better performance with the increase of client number.
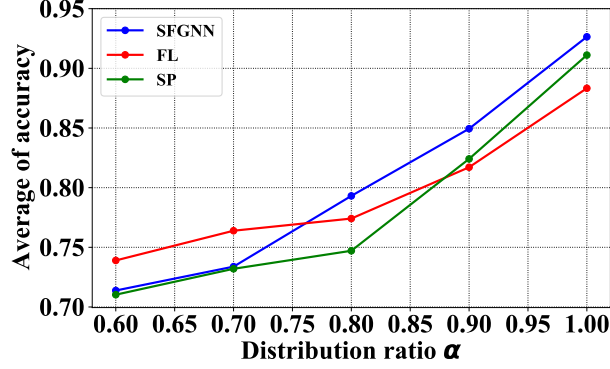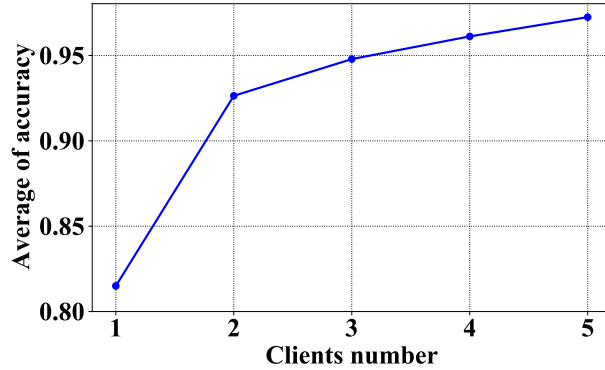
Figure 3: Average accuracy comparison of three models with different α.

Figure 4: Average accuracy comparison of different clients' number with Non-IID label.

5.2.5 Accuracy comparison of JS divergence

To answer the proposed question **Q4**, we execute the **FGNN** model in Algorithm 2 in an another way. That is, clients update the local discrimination models using global discrimination model directly. Then we compare the accuracy with different α on Cora dataset, the results are shown in Table 6. From it, we find that SFGNN with JS consistently outperforms SFGNN without JS, which shows the effectiveness of the proposed JS method. Besides, we also find that the promotion of JS on SFGNN increases with the raise of α. The contribution of JS method becomes negligible when clients tend to have IID data, e.g., $\alpha = 0.6$.

5.3 Efficiency comparison

To answer the proposed question **Q4**, we compare the parameters tuning time of grid search and Bayesian optimization on the three datasets. We perform the Bayesian optimization of hyper-parameters with the help of the open source framework *SMAC3* [37]. The domains of l_r, l_2, d in *SMAC3* are continuous, and the domains of K, l are discrete with the interval of 1. Both grid search method and Bayesian optimization method are implemented under the same computation and communication environment. We report the parameter tuning time in Table 7. Note that both methods achieve

Table 6: Performance comparison of JS in SFGNN model.

Ratio α	Without JS	With JS	Improvement
1.0	0.9081	0.9264	3.66%
0.9	0.8226	0.8494	3.26%
0.8	0.7692	0.7931	3.10%
0.7	0.7120	0.7338	3.06%
0.6	0.7018	0.7138	1.72%

12

Table 7: Training time comparison between BO and grid search on three datasets.

Datasets	Grid Search	BO	Speedup
Cora	6.67h	0.39h	17.10
Citeseer	40.85h	0.42h	97.26
Pubmed	427.67h	4.17h	102.56

comparable accuracy on these three datasets. From Table 7, we can observe that, (1) the Bayesian optimization method greatly reduces the hyper-parameters tuning time on all the three datasets, comparing with the traditional grid search method, and (2) the speedup of Bayesian optimization against grid search becomes higher when dataset gets larger. For example, the speedup on Pubmed dataset is 102.56 while it is 10.10 on Cora dataset. This is because Bayesian optimization reduce the parameter tuning time of grid search by decreasing the parameter search space. In real-world applications, the network bandwidth is always limited between clients and server, and the model training procedure under data isolated setting usually takes much longer time then traditional centralized model training. Therefore, decreasing the parameter search space becomes the key of reducing the tuning time. The results demonstrate that our proposal is good at doing this.

6   Conclusion and future network

In this paper, we proposed a Automated Separated-Federated GNN learning paradigm in the Non-IID isolated scenario. We first proposed a separated-federated GNN learning model, which decoupled the training of GNN into two parts: the message passing part was done by clients separately, and the loss computing part was learnt by clients federally. To handle the time-consuming problem, we leveraged the Bayesian optimization technique to automatically tune the hyper-parameters of all the clients. Experiments on real world datasets demonstrated that our model significantly outperformed the federated GNN learning on the isolated Non-IID data.

In the future, we would like to verify our proposal with more existing GNN models. We are also interested in deploying our proposal into real-world applications.

References

[1] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[2] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[3] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[4] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*, pages 974–983. ACM, 2018.

[5] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 2077–2085, New York, NY, USA, 2018. Association for Computing Machinery.

[6] Pablo Ribalta Lorenzo, Jakub Nalepa, Luciano Sanchez Ramos, and José Ranilla Pastor. Hyper-parameter selection in deep neural networks using parallel particle swarm optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, page 1864–1871, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2019.

[8] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data, 2018.

[9] Yi-Wei Chen, Qingquan Song, and Xia Hu. Techniques for automated machine learning, 2019.

[10] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

[11] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *ArXiv*, abs/1602.05629, 2016.

[12] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2018.

[13] Om Thakkar, Galen Andrew, and H. Brendan McMahan. Differentially private learning with adaptive clipping, 2019.

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[15] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[16] Yansong Gao, Minki Kim, Sharif Abuadbba, Yeonjae Kim, Chandra Thapa, Kyuyeon Kim, Seyit A Camtepe, Hyoungshick Kim, and Surya Nepal. End-to-end evaluation of federated learning and split learning for internet of things. *arXiv preprint arXiv:2003.13376*, 2020.

[17] Sharif Abuadbba, Kyuyeon Kim, Minki Kim, Chandra Thapa, Seyit A Camtepe, Yansong Gao, Hyoungshick Kim, and Surya Nepal. Can we use split learning on 1d cnn models for privacy preserving training? *arXiv preprint arXiv:2003.12365*, 2020.

[18] Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task bayesian optimization. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2004–2012, Red Hook, NY, USA, 2013. Curran Associates Inc.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

[20] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017.

[21] Guangxu Mei, Ziyu Guo, Shijun Liu, and Li Pan. Sgnn: A graph neural network based federated learning approach by hiding structure. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2560–2568. IEEE, 2019.

[22] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1024–1034, 2017.

[23] Patricia S. Abril and Robert Plant. A comprehensive survey on graph neural networks. *Communications of the ACM*, 50(1):36–44, January 2007.

[24] 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.

[25] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[26] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Scalable and secure logistic regression via homomorphic encryption. In *CODASPY*, pages 142–144. ACM, 2016.

[27] 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.

[28] Chaochao Chen, Jun Zhou, Li Wang, Xibin Wu, Wenjing Fang, Jin Tan, Lei Wang, Xiaoxi Ji, Alex Liu, Hao Wang, et al. When homomorphic encryption marries secret sharing: Secure large-scale sparse logistic regression and applications in risk control. *arXiv preprint arXiv:2008.08753*, 2020.

[29] J. LIN and S. K. M. WONG. A new directed divergence measure and its characterization. *International Journal of General Systems*, 17(1):73–81, 1990.

[30] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. *ArXiv*, abs/1707.03815, 2017.

[31] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimizationb. *Journal of Electronic Science and Technology*, 17(1):26 – 40, 2019.

[32] Zhongshu Gu, Heqing Huang, Jialong Zhang, Dong Su, Ankita Lamba, Dimitrios Pendarakis, and Ian Molloy. Securing input data of deep learning inference systems via partitioned enclave execution. *CoRR*, abs/1807.00969, 2019.

[33] Longfei Zheng, Chaochao Chen, Yingting Liu, Bingzhe Wu, Xibin Wu, Li Wang, Lei Wang, Jun Zhou, and Shuang Yang. Industrial scale privacy preserving deep neural network. *arXiv preprint arXiv:2003.05198*, 2020.

[34] 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. Software available from tensorflow.org.

[35] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[36] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4424–4431, 2019.

[37] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, André Biedenkapp, and Frank Hutter. Smac v3: Algorithm configuration in python. `https://github.com/automl/SMAC3`, 2017.