

Tackling Resource Allocation for Decentralized Federated Learning: A GNN-based Approach

Chuiyang Meng*, *Graduate Student Member, IEEE*, Ming Tang†, *Member, IEEE*, Mehdi Setayesh*, *Member, IEEE*, and Vincent W.S. Wong*, *Fellow, IEEE*

Abstract—Decentralized federated learning (DFL) enables clients to train a neural network model in a device-to-device (D2D) manner without central coordination. In practical systems, DFL faces challenges due to dynamic topology changes, time-varying channel conditions, and limited computational capability of the clients. These factors can affect the learning performance and efficiency of DFL. To address the aforementioned challenges, in this paper, we propose a graph neural network (GNN)-based algorithm to minimize the total delay and energy consumption on training and improve the learning performance of DFL in D2D wireless networks. In our proposed GNN, a multi-head graph attention mechanism is used to capture different features of clients and wireless channels. We design a neighbor selection module which enables each client to select a subset of its neighbors for the participation of model aggregation. We develop a decoder that enables each client to determine its transmit power and computational resource. Experimental results show that our proposed algorithm achieves a lower total delay and energy consumption on training when compared with five baseline schemes. Furthermore, by properly selecting a subset of neighbors for each client, our proposed algorithm achieves similar testing accuracy to the full participation scheme.

Index Terms—Decentralized federated learning, graph neural networks, neighbor selection, resource allocation.

I. INTRODUCTION

Federated learning (FL) [1] [2] is a decentralized machine learning framework that allows clients to collaboratively train a global model using their local datasets under the coordination of a central server. However, conventional centralized FL schemes are vulnerable to the single point of failure since the central server has to remain available for model aggregation [2]–[5]. Moreover, the wireless links between the clients and the central server may sometimes be congested due to concurrent uplink transmissions, which may increase the communication delay for FL. To address these challenges, decentralized federated learning (DFL) [6] [7] has emerged as an alternative, leveraging device-to-device (D2D) wireless communications and decentralized stochastic gradient descent

(DSGD) techniques. DFL relies on the model exchange between clients without the assistance of a central server. It has two distinctive advantages. First, DFL is robust to the single point of failure due to its decentralized structure. Second, DFL does not require uplink transmission and hence avoids congestion at the central server. In particular, DFL has been studied under several application scenarios. In healthcare, DFL facilitates collaborative model training between hospitals and medical institutions while ensuring data privacy [8]–[11]. In vehicular networks, DFL uses vehicle-to-vehicle communication, enabling safer and more efficient transportation systems [12] [13]. In industrial Internet of Things (IIoT), DFL can be trained by harnessing distributed sensor data across industrial facilities [14] [15]. It can improve the operational efficiency and reliability. In mobile services, DFL can personalize user experiences while reducing the communication overhead [16] [17]. In addition, some companies have deployed DFL. For example, Hewlett Packard Enterprise has deployed DFL with blockchain to coordinate artificial intelligence (AI) model training among multiple distributed nodes [18].

However, DFL faces several challenges. First, the heterogeneity of clients can reduce the convergence rate of DFL. Clients with lower communication and computational capabilities may become stragglers, and they take longer time than other clients to send their model updates to their neighbors. The stragglers may reduce the frequency of model aggregation and increase the convergence time. The long convergence time can increase the communication and computation resource usage. Second, the convergence rate can also be reduced due to the decentralized structure of DFL. This is because each client can only exchange the local model with its neighbors. This may result in a low degree of consensus among clients' local models. Thus, it can degrade the learning performance of DFL. Third, due to the mobility of clients, the D2D network topology may change over time, which can hinder the system from achieving a high degree of consensus of local models across clients.

Most of the recent works on DFL focus on decentralized algorithm design [19]–[24]. Each client performs model aggregation with all of its available neighbors (i.e., full participation) in each training round, aiming to obtain local models with a high consensus. However, aggregating all the neighboring clients' models can consume a lot of network resources. Hence, we consider a neighbor selection scheme for DFL to improve the learning efficiency. Although some recent works have proposed neighbor selection schemes for DFL [25]–[30], they do not consider the dynamics of wireless

Manuscript received on Jul. 16, 2024; revised on Jan. 19, 2025 and Apr. 10, 2025; accepted on Apr. 15, 2025. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Digital Research Alliance of Canada (alliancecan.ca). (*Corresponding author: Vincent W.S. Wong*)

*Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada

†Department of Computer Science and Engineering, Southern University of Science and Technology, China

E-mail: {chuiyangmeng, setayeshm, vincentw}@ece.ubc.ca, tangm3@sustech.edu.cn

Color versions of one or more of the figures in this paper are available online at <https://ieeexplore.ieee.org>.

networks. In particular, a client may not be able to maintain reliable communications with its neighbors due to client's mobility and dynamic channel conditions. Hence, we aim to design a neighbor selection-based DFL framework in time-varying wireless environments.

In recent years, graph neural networks (GNNs) have been developed as effective tools to solve resource allocation problems in wireless networks [31]–[34]. Several recent developments in FL have introduced new applications of GNNs [35]–[37]. However, the aforementioned works utilize GNNs in the conventional centralized FL setting, which makes them incompatible with DFL.

In this paper, we study the following problem: *How to select neighbors and allocate computation and communication resources for each client efficiently in a D2D network to optimize the performance of DFL?* We aim to develop an algorithm which can achieve a low total delay and energy consumption on training and a high testing accuracy for DFL applications. The design is challenging due to the following reasons. First, it is difficult to quantify the impact of system characteristics (e.g., dynamic changes in the network topology, time-varying wireless channel conditions, and client heterogeneity) on the resource utilization and learning performance of DFL. Second, in decentralized settings, obtaining the global model can consume a significant amount of computation and wireless resources. Nevertheless, the learning performance of DFL depends on the consensus degree across clients' local models. Third, the necessity for the decentralized implementation of DFL complicates the solution. To address the aforementioned challenges, we propose a GNN-based algorithm to select neighbors and allocate wireless resources and central processing unit (CPU) to each client. This work is an extension of our previous work in [38]. The contributions of this paper are as follows:

- We propose a DFL framework which facilitates cooperative model training across clients in a D2D network. This framework enables partial client participation by incorporating neighbor selection to improve the learning efficiency. By conducting theoretical analysis, we derive the convergence bound of our proposed DFL.
- We formulate a neighbor selection and resource allocation problem for DFL. The objective is to jointly minimize the total delay and energy consumption on training and achieve a high consensus across clients' local models to guarantee the learning performance of our proposed DFL.
- We design an attention-based GNN encoder to capture the intrinsic properties of the D2D network and the local model parameters of clients. To reduce the total delay and energy consumption on training, we design a neighbor selection module to choose the neighbors for model aggregation. We also design a decoder module to determine the transmit power and CPU frequency for each client. By training our proposed GNN with different network topologies and wireless channel conditions, our proposed GNN can adapt to dynamic network settings. Our proposed GNN can be executed by each client in a distributed manner, which facilitates implementation in practical systems.

- We conduct experiments using Fashion-MNIST, CIFAR-10, and ImageNet datasets. We compare our proposed algorithm with five baseline schemes, including full participation, random sampling, FedHP scheme [27], performance-based neighbor selection (PENS) scheme [28], and a path-following FL scheme [39]. Results show that our proposed algorithm achieves a lower total delay and energy consumption than the baselines. Furthermore, it achieves a comparable average testing accuracy to the full participation DFL scheme.

The rest of this paper is organized as follows. The system model, theoretical convergence analysis, and problem formulation are presented in Section II. The proposed GNN-based algorithm is described in Section III. The experimental results are presented in Section IV. Conclusion is given in Section V.

Notations: In this paper, we use calligraphic upper case letters, boldface upper case letters, and boldface lower case letters to denote sets, matrices/tensors, and vectors, respectively. $\mathbb{R}^{M \times N}$ denotes the set of $M \times N$ real-valued matrices. $\mathbb{1}(\cdot)$ denotes the indicator function, which is equal to 1 if the argument is true and is equal to 0 otherwise. $\mathbf{1} \in \mathbb{R}^N$ denotes the all-ones column vector. $\mathbb{E}[\cdot]$ denotes the expected value of a random variable. The operators $\langle \cdot \rangle$, $\odot$, $|\cdot|$, $\|\cdot\|_1$, $\|\cdot\|$, $\|\cdot\|_F$, $\|$, $(\cdot)^T$, and $\lfloor \cdot \rfloor$ denote the inner product, elementwise multiplication, cardinality, 1-norm, 2-norm, Frobenius norm, concatenation, transpose, and floor function, respectively. $\sim$ denotes “distributed as”. $\mathcal{U}[\cdot]$ denotes the uniform distribution.

II. SYSTEM MODEL

We consider a D2D wireless network with N distributed clients in a given area. The DFL process iterates for R training rounds. In each training round, each client performs K stochastic gradient descent (SGD) steps. To capture the mobility of clients, we model the D2D network as a time-varying directed graph $\mathcal{G}^r(\mathcal{V}, \mathcal{E}^r)$, where $\mathcal{V} = \{1, 2, \dots, N\}$ and $\mathcal{E}^r$ denote the set of clients (i.e., nodes) and the set of communication links (i.e., edges) between clients in the r -th training round, respectively. We consider that clients can move freely at the end of each training round, while they remain stationary within each training round. There is a communication link between two clients when their signal-to-interference-plus-noise ratio (SINR) is greater than or equal to a threshold Ω . The value of Ω controls the node degree (i.e., the number of neighbors of a client) in the D2D network. The link from client i to client j is denoted by $(i, j) \in \mathcal{E}^r$. We define $\mathbf{A}^r \in \{0, 1\}^{N \times N}$ as the adjacency matrix in the r -th training round with $\mathbf{a}_i^r \in \mathbb{R}^{1 \times N}$, $i \in \mathcal{V}$, denoting the i -th row of matrix $\mathbf{A}^r$. We have $a_{i,j}^r = \mathbb{1}((i, j) \in \mathcal{E}^r)$. We define the set of neighbors of client i in the r -th training round as $\mathcal{N}_i^r = \{j \mid j \in \mathcal{V}, a_{i,j}^r = 1\}$.

In the following, we first propose the DFL framework. Then, we provide the convergence analysis and present the delay and energy constraints. Finally, we formulate the neighbor selection and resource allocation problem.

A. Decentralized Federated Learning

Each client $i \in \mathcal{V}$ has a dataset $\mathcal{D}_i$ with D_i training samples. Let $\boldsymbol{\theta}_i^r \in \mathbb{R}^{D_m}$ denote the model parameters of client i in the

r -th training round, where D_m denotes the dimension of θ_i^r . The loss function of the model of client i is given by

$$F_i(\theta_i^r) = \mathbb{E}_{\xi_i^r \sim \mathcal{D}_i} [F_i(\theta_i^r; \xi_i^r)], \quad i \in \mathcal{V}, \quad (1)$$

where $F_i(\theta_i^r; \xi_i^r)$ denotes the loss on data samples in a mini-batch $\xi_i^r \sim \mathcal{D}_i$ with model θ_i^r . Let θ_G^r denote the parameters of the global model at the beginning of the r -th training round. In centralized FL, the global model is obtained in the central server as $\theta_G^r = \frac{1}{\sum_{j=1}^N D_j} \sum_{i=1}^N D_i \theta_i^r$. The loss function of the global model is given by

$$F(\theta_G^r) = \frac{1}{\sum_{j=1}^N D_j} \sum_{i=1}^N D_i F_i(\theta_G^r). \quad (2)$$

Note that θ_G^r is the ground-truth, while acquiring it in D2D networks is challenging due to the decentralized topology. Hence, DFL uses the DSGD algorithm as an alternative to learn a common model across clients, aiming to minimize $F(\theta_G^r)$ in a decentralized manner [3]. In particular, each client performs several steps of local training and exchanges its model parameters with its neighbors in each training round.

The DFL process is depicted in Fig. 1. Before training, each client has the same randomly initialized local model parameters. In the r -th training round, each client $i \in \mathcal{V}$ performs the following three steps. First, client i trains its model locally with K SGD steps. Let $\nabla F_i(\theta_i^{r,k}; \xi_i^r)$ denote the gradient of the loss function of client i at the k -th SGD step in the r -th training round. We have $\nabla F_i(\theta_i^r; \xi_i^r) = \sum_{k=0}^{K-1} \nabla F_i(\theta_i^{r,k}; \xi_i^r)$. Therefore, client i 's model in the r -th training round is locally updated as follows:

$$\begin{aligned} \theta_i^{r,K} &= \theta_i^{r,0} - \eta \sum_{k=0}^{K-1} \nabla F_i(\theta_i^{r,k}; \xi_i^r) \\ &= \theta_i^{r,0} - \eta \nabla F_i(\theta_i^r; \xi_i^r), \end{aligned} \quad (3)$$

where η denotes the step size. For illustration simplicity, we consider that each client has the same number of training samples in its local dataset. The average model at the end of the r -th training round is given by $\theta_G^{r,K} = \frac{1}{N} \sum_{i=1}^N \theta_i^{r,K}$.

In the second step, client i selects a subset of neighboring clients to send its local model update and receives the corresponding model updates from its selected neighbors. We define the neighbor selection decision variable as $s_{i,j}^r \in \{0, 1\}$, which satisfies $s_{i,j}^r \leq a_{i,j}^r, i, j \in \mathcal{V}$. The neighbor selection matrix for aggregation in the r -th training round is denoted as $\mathbf{S}^r \in \mathbb{R}^{N \times N}$, with $\mathbf{s}_i^r \in \mathbb{R}^{1 \times N}$ denoting the neighbor selection row vector of client i . We define the set of neighbors that client i has selected for aggregation in the r -th training round as $\hat{\mathcal{N}}_i^r = \{j \mid s_{i,j}^r = 1, j \in \mathcal{V}\}$. $\hat{\mathcal{N}}_i^r$ is a set of indices corresponding to non-zero elements in vector $\mathbf{s}_i^r$. We have $\hat{\mathcal{N}}_i^r \subseteq \mathcal{N}_i^r$. Then, client i sends the local model update $\theta_i^{r,K}$ to each of its selected neighbor $j \in \hat{\mathcal{N}}_i^r$. In DFL, each client needs to exchange local model parameters with its neighbors, which requires bi-directional model exchange. That is, client i also receives the updated model $\theta_j^{r,K}$ from its selected neighbor j . We have $s_{i,j}^r = s_{j,i}^r$. In addition, client i includes its own model for aggregation. We also have $s_{i,i}^r = 1$.

In the third step, client i updates its local model by aggregat-

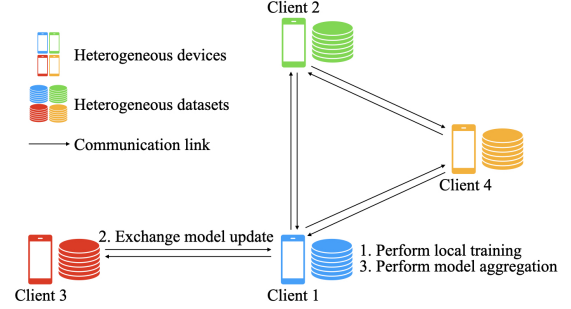


Fig. 1: An illustration of DFL in a D2D wireless system network. Each client shares its local model parameters with its neighbors after local training.

ing the local models of its selected neighbors. It uses a doubly stochastic mixing weight matrix $\mathbf{W}^r \in \mathbb{R}^{N \times N}$ for model aggregation. Similar to [40], we set the weights according to the maximum number of neighbors among all clients as

$$w_{i,j}^r = \begin{cases} 1 - \frac{|\mathcal{N}_i^r|}{\max_{l \in \mathcal{V}} \{|\mathcal{N}_l^r|\}}, & \text{if } i = j, \\ \frac{1}{\max_{l \in \mathcal{V}} \{|\mathcal{N}_l^r|\}}, & \text{otherwise.} \end{cases} \quad (4)$$

Client i then updates the local model in the r -th training round as follows:

$$\theta_i^{r+1,0}(\mathbf{s}_i^r) \triangleq \sum_{j \in \hat{\mathcal{N}}_i^r \cup \{i\}} w_{i,j}^r \theta_j^{r,K}. \quad (5)$$

$\theta_i^{r+1,0}$ will be used as the model of client i to perform local training in the $(r+1)$ -th training round.

B. Convergence Analysis

Due to the decentralized structure of DFL, local models across clients may differ, which may affect the learning performance of DFL. In this subsection, we conduct the theoretical analysis of our proposed DFL, discussing the impact of the neighbor selection on the model discrepancies and the convergence rate.

Let $\theta \in \mathbb{R}^{D_m}$ and $\xi_i \sim \mathcal{D}_i$ denote the model parameters and the local training samples of client i in any training round, respectively. We make the following assumptions, which are widely used in the literature (e.g., [21], [23], [24], [27], [41]).

Assumption 1. (L -smoothness) *The loss function of each client $i \in \mathcal{V}$ is continuously differentiable and is L -smooth. That is, for any $\theta_1, \theta_2 \in \mathbb{R}^{D_m}$, we have*

$$F_i(\theta_1) \leq F_i(\theta_2) + \nabla F_i(\theta_2)^\top (\theta_1 - \theta_2) + \frac{L}{2} \|\theta_1 - \theta_2\|^2. \quad (6)$$

Assumption 2. (Bounded gradient variance) *The variance of stochastic gradient obtained by using the loss function of each client $i \in \mathcal{V}$ is bounded by*

$$\mathbb{E} \left[\left\| \nabla F_i(\theta; \xi_i) - \nabla F_i(\theta) \right\|^2 \right] \leq \sigma_F^2. \quad (7)$$

Assumption 3. (Bounded gradients) *The stochastic gradient obtained by using the loss function of each client $i \in \mathcal{V}$ is bounded by*

$$\mathbb{E} \left[\left\| \nabla F_i(\theta; \xi_i) \right\|^2 \right] \leq \psi_F^2. \quad (8)$$

In addition, we introduce the following lemma.

Lemma 1. For arbitrary set of vectors $\{\mathbf{x}_i\}_{i=1}^N$, we have

$$\left\| \sum_{i=1}^N \frac{1}{N} \mathbf{x}_i \right\|^2 \leq \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i\|^2.$$

Proof. The square of 2-norm (i.e., $\|\cdot\|^2$) is convex. According to Jensen's inequality, we have

$$\left\| \sum_{i=1}^N \omega_i \mathbf{x}_i \right\|^2 \leq \sum_{i=1}^N \omega_i \|\mathbf{x}_i\|^2, \quad (9)$$

where $\omega_1, \omega_2, \dots, \omega_N$ are non-negative coefficients. When we set $\omega_1 = \omega_2 = \dots = \omega_N = \frac{1}{N}$, we obtain $\left\| \sum_{i=1}^N \frac{1}{N} \mathbf{x}_i \right\|^2 \leq \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i\|^2$. $\square$

Let $L_i^{\text{gap},r}(\mathbf{s}_i^r) = \|\boldsymbol{\theta}_i^{r+1,0}(\mathbf{s}_i^r) - \boldsymbol{\theta}_G^{r,K}\|^2$ denote the consensus distance between the model of client i and the global model at the end of the r -th training round. Let F^* denote the minimum value of F across the feasible domain. We now present the result of the convergence rate of our proposed DFL in the following theorem.

Theorem 1. Under Assumptions 1–3 and Lemma 1, the convergence rate of our proposed DFL is bounded by

$$\begin{aligned} & \frac{1}{R} \sum_{r=0}^{R-1} \left\| \nabla F(\boldsymbol{\theta}_G^r) \right\|^2 \\ & \leq \underbrace{\frac{2(F(\boldsymbol{\theta}_G^0) - F^*)}{R}}_{\text{Initial optimality gap}} + \underbrace{\frac{2(L+2)}{NR} \sum_{r=0}^{R-1} \sum_{i=1}^N L_i^{\text{gap},r}(\mathbf{s}_i^r)}_{\text{Consensus distance induced gap}} \\ & \quad + \underbrace{2(L+2)\eta^2\psi_F^2 + 3(\sigma_F^2 + \psi_F^2)}_{\text{Gradient induced gap}}. \end{aligned} \quad (10)$$

Proof. We first introduce the following notations to facilitate our proof:

$$\begin{aligned} \boldsymbol{\Theta}^{r,0} &= (\boldsymbol{\theta}_1^{r,0}, \boldsymbol{\theta}_2^{r,0}, \dots, \boldsymbol{\theta}_N^{r,0}) \in \mathbb{R}^{D_m \times N}, \\ \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r) &= (\nabla F_1(\boldsymbol{\theta}_1^r; \boldsymbol{\xi}_1^r), \nabla F_2(\boldsymbol{\theta}_2^r; \boldsymbol{\xi}_2^r), \dots, \\ & \quad \nabla F_N(\boldsymbol{\theta}_N^r; \boldsymbol{\xi}_N^r)) \in \mathbb{R}^{D_m \times N}, \\ \mathbf{W}_s^r &= \mathbf{W}^r \odot \mathbf{S}^r \in \mathbb{R}^{N \times N}. \end{aligned}$$

We begin by expanding the equation of the model update into several terms. Then, we analyze each term respectively to bound the convergence rate. Based on eqns. (3) and (5), in the r -th training round, the model is updated as follows:

$$\begin{aligned} \boldsymbol{\Theta}^{r+1,0} &= (\boldsymbol{\Theta}^{r,0} - \eta \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r)) \mathbf{W}_s^r \\ &= \boldsymbol{\Theta}^{r,0} \mathbf{W}^r + \boldsymbol{\Theta}^{r,0} (\mathbf{W}_s^r - \mathbf{W}^r) - \eta \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r) \mathbf{W}_s^r. \end{aligned} \quad (11)$$

The loss made on the global model after the model update at the beginning of the $(r+1)$ -th training round is given by

$$\begin{aligned} & F(\boldsymbol{\theta}_G^{r+1}) \\ &= F\left(\frac{\boldsymbol{\Theta}^{r+1,0} \mathbf{1}_N}{N}\right) \\ &= F\left(\frac{\boldsymbol{\Theta}^{r,0} \mathbf{W}^r \mathbf{1}_N}{N} + \frac{\boldsymbol{\Theta}^{r,0} (\mathbf{W}_s^r - \mathbf{W}^r) \mathbf{1}_N}{N} - \frac{\eta \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r) \mathbf{W}_s^r \mathbf{1}_N}{N}\right) \end{aligned}$$

$$\begin{aligned} & \stackrel{(a)}{\leq} F\left(\frac{\boldsymbol{\Theta}^{r,0} \mathbf{W}^r \mathbf{1}_N}{N}\right) + \left\langle \nabla F\left(\frac{\boldsymbol{\Theta}^{r,0} \mathbf{W}^r \mathbf{1}_N}{N}\right), \right. \\ & \quad \left. \frac{\boldsymbol{\Theta}^{r,0} (\mathbf{W}_s^r - \mathbf{W}^r) \mathbf{1}_N - \eta \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r) \mathbf{W}_s^r \mathbf{1}_N}{N} \right\rangle \\ & \quad + \frac{L}{2} \left\| \frac{\boldsymbol{\Theta}^{r,0} (\mathbf{W}_s^r - \mathbf{W}^r) \mathbf{1}_N - \eta \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r) \mathbf{W}_s^r \mathbf{1}_N}{N} \right\|^2, \end{aligned} \quad (12)$$

where inequality (a) results from Assumption 1. Then, we take expectation on both sides of inequality (12). Since $\mathbf{W}^r$ is a doubly stochastic matrix, we have

$$\begin{aligned} & F(\boldsymbol{\theta}_G^{r+1}) \\ & \leq F\left(\frac{\boldsymbol{\Theta}^{r,0} \mathbf{1}_N}{N}\right) + \mathbb{E} \left[\left\langle \nabla F\left(\frac{\boldsymbol{\Theta}^{r,0} \mathbf{1}_N}{N}\right), \right. \right. \\ & \quad \left. \left. \frac{\boldsymbol{\Theta}^{r,0} (\mathbf{W}_s^r - \mathbf{W}^r) \mathbf{1}_N - \eta \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r) \mathbf{W}_s^r \mathbf{1}_N}{N} \right\rangle \right] \\ & \quad + \frac{L}{2} \mathbb{E} \left[\left\| \frac{\boldsymbol{\Theta}^{r,0} (\mathbf{W}_s^r - \mathbf{W}^r) \mathbf{1}_N}{N} - \frac{\eta \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r) \mathbf{W}_s^r \mathbf{1}_N}{N} \right\|^2 \right] \\ & = F(\boldsymbol{\theta}_G^r) + \mathbb{E} \left[\left\langle \nabla F(\boldsymbol{\theta}_G^r), \frac{(\boldsymbol{\Theta}^{r,0} - \eta \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r)) \mathbf{W}_s^r \mathbf{1}_N}{N} \right. \right. \\ & \quad \left. \left. - \frac{\boldsymbol{\Theta}^{r,0} \mathbf{W}^r \mathbf{1}_N}{N} \right\rangle \right] \\ & \quad + \frac{L}{2} \mathbb{E} \left[\left\| \frac{(\boldsymbol{\Theta}^{r,0} - \eta \nabla F(\boldsymbol{\Theta}^r; \boldsymbol{\xi}^r)) \mathbf{W}_s^r \mathbf{1}_N}{N} - \frac{\boldsymbol{\Theta}^{r,0} \mathbf{W}^r \mathbf{1}_N}{N} \right\|^2 \right] \\ & = F(\boldsymbol{\theta}_G^r) + \mathbb{E} \left[\left\langle \nabla F(\boldsymbol{\theta}_G^r), \frac{(\boldsymbol{\Theta}^{r+1,0} - \boldsymbol{\Theta}^{r,0} \mathbf{W}^r) \mathbf{1}_N}{N} \right\rangle \right] \\ & \quad + \frac{L}{2} \mathbb{E} \left[\left\| \frac{(\boldsymbol{\Theta}^{r+1,0} - \boldsymbol{\Theta}^{r,0} \mathbf{W}^r) \mathbf{1}_N}{N} \right\|^2 \right] \\ & = F(\boldsymbol{\theta}_G^r) + \underbrace{\mathbb{E} \left[\left\langle \nabla F(\boldsymbol{\theta}_G^r), \boldsymbol{\theta}_G^{r+1} - \boldsymbol{\theta}_G^r \right\rangle \right]}_{T_1} \\ & \quad + \underbrace{\frac{L}{2} \mathbb{E} \left[\left\| \boldsymbol{\theta}_G^{r+1} - \boldsymbol{\theta}_G^r \right\|^2 \right]}_{T_2}, \end{aligned} \quad (13)$$

where T_1 measures the alignment of the update of the global model and the gradient in the r -th training round. T_2 measures the update of the global model in the r -th training round. Now, we analyze T_1 . For any vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, we have $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \frac{1}{2}(\|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|^2 - \|\mathbf{x}_1 - \mathbf{x}_2\|^2)$. Thus, T_1 satisfies

$$\begin{aligned} T_1 &= \mathbb{E} \left[\left\langle \nabla F(\boldsymbol{\theta}_G^r), \boldsymbol{\theta}_G^{r+1} - \boldsymbol{\theta}_G^r \right\rangle \right] \\ &= -\mathbb{E} \left[\left\langle \nabla F(\boldsymbol{\theta}_G^r), \boldsymbol{\theta}_G^r - \boldsymbol{\theta}_G^{r+1} \right\rangle \right] \\ &= -\frac{1}{2} \mathbb{E} \left[\left\| \nabla F(\boldsymbol{\theta}_G^r) \right\|^2 + \left\| \boldsymbol{\theta}_G^r - \boldsymbol{\theta}_G^{r+1} \right\|^2 \right. \\ & \quad \left. - \left\| \nabla F(\boldsymbol{\theta}_G^r) - (\boldsymbol{\theta}_G^r - \boldsymbol{\theta}_G^{r+1}) \right\|^2 \right] \\ &= \frac{1}{2} \underbrace{\mathbb{E} \left[\left\| \nabla F(\boldsymbol{\theta}_G^r) - (\boldsymbol{\theta}_G^r - \boldsymbol{\theta}_G^{r+1}) \right\|^2 \right]}_{T_3} - \frac{1}{2} \left\| \nabla F(\boldsymbol{\theta}_G^r) \right\|^2 \\ & \quad - \frac{1}{2} T_2, \end{aligned} \quad (14)$$

where T_3 measures the deviation of the gradient and the update of the global model in the r -th training round. Note that T_1 depends on T_2 and T_3 . Hence, we first analyze T_3 and then analyze T_2 . In particular, T_3 satisfies

$$\begin{aligned}
T_3 &= \mathbb{E} \left[\left\| \nabla F(\theta_G^r) - (\theta_G^r - \theta_G^{r+1}) \right\|^2 \right] \\
&\stackrel{(a)}{=} \mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N \nabla F_i(\theta_G^r) + (\theta_G^{r+1} - \theta_G^r) \right\|^2 \right] \\
&= \mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N (\nabla F_i(\theta_G^r) - \nabla F_i(\theta_G^r; \xi_i^r) + \nabla F_i(\theta_G^r; \xi_i^r)) \right. \right. \\
&\quad \left. \left. + (\theta_G^{r+1} - \theta_G^r) \right\|^2 \right] \\
&\stackrel{(b)}{\leq} 3\mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N (\nabla F_i(\theta_G^r) - \nabla F_i(\theta_G^r; \xi_i^r)) \right\|^2 \right] \\
&\quad + 3\mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N \nabla F_i(\theta_G^r; \xi_i^r) \right\|^2 \right] + 3\mathbb{E} \left[\left\| \theta_G^{r+1} - \theta_G^r \right\|^2 \right] \\
&\stackrel{(c)}{\leq} \frac{3}{N} \sum_{i=1}^N \mathbb{E} \left[\left\| \nabla F_i(\theta_G^r) - \nabla F_i(\theta_G^r; \xi_i^r) \right\|^2 \right] \\
&\quad + \frac{3}{N} \sum_{i=1}^N \mathbb{E} \left[\left\| \nabla F_i(\theta_G^r; \xi_i^r) \right\|^2 \right] + 3T_2 \\
&\stackrel{(d)}{\leq} 3(\sigma_F^2 + \psi_F^2) + 3T_2, \tag{15}
\end{aligned}$$

where equality (a) is obtained by using $\nabla F(\theta_G^r) = \frac{1}{N} \sum_{i=1}^N \nabla F_i(\theta_G^r)$. Inequalities (b) and (c) are obtained by using Lemma 1. Inequality (d) is obtained by using Assumptions 2 and 3. By substituting T_3 into T_1 , we have

$$T_1 \leq -\frac{1}{2} \left\| \nabla F(\theta_G^r) \right\|^2 + T_2 + \frac{3}{2}(\sigma_F^2 + \psi_F^2). \tag{16}$$

Then, we analyze T_2 . In particular, it satisfies

$$\begin{aligned}
T_2 &= \mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N \theta_i^{r+1,0} - \frac{1}{N} \sum_{i=1}^N \theta_i^{r,0} \right\|^2 \right] \\
&= \mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N (\theta_i^{r+1,0} - \theta_i^{r,K} + \theta_i^{r,K} - \theta_i^{r,0}) \right\|^2 \right] \\
&\stackrel{(a)}{\leq} 2\mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N (\theta_i^{r+1,0} - \theta_i^{r,K}) \right\|^2 \right] \\
&\quad + 2\mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N (\theta_i^{r,K} - \theta_i^{r,0}) \right\|^2 \right] \\
&= 2 \left\| \theta_G^{r+1} - \theta_G^{r,K} \right\|^2 + 2\mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N \eta \nabla F_i(\theta_i^r; \xi_i^r) \right\|^2 \right] \\
&\stackrel{(b)}{\leq} 2 \left\| \theta_G^{r+1} - \theta_G^{r,K} \right\|^2 + 2\eta^2 \psi_F^2, \tag{17}
\end{aligned}$$

where inequality (a) is obtained by using Lemma 1. Inequality (b) results from Assumption 3. By substituting T_1 and T_2 into inequality (13), we have

$$F(\theta_G^{r+1}) \leq F(\theta_G^r) - \frac{1}{2} \left\| \nabla F(\theta_G^r) \right\|^2 + (L+2)\eta^2 \psi_F^2$$

$$+ (L+2) \left\| \theta_G^{r+1} - \theta_G^{r,K} \right\|^2 + \frac{3}{2}(\sigma_F^2 + \psi_F^2). \tag{18}$$

Finally, we rearrange inequality (18). By summing up all R training rounds and multiplying $\frac{2}{R}$ on both sides, we have

$$\begin{aligned}
&\frac{1}{R} \sum_{r=0}^{R-1} \left\| \nabla F(\theta_G^r) \right\|^2 \\
&\stackrel{(a)}{\leq} \frac{2(F(\theta_G^0) - F^*)}{R} + \frac{2(L+2)}{R} \sum_{r=0}^{R-1} \left\| \theta_G^{r+1} - \theta_G^{r,K} \right\|^2 \\
&\quad + 2(L+2)\eta^2 \psi_F^2 + 3(\sigma_F^2 + \psi_F^2) \\
&= \frac{2(F(\theta_G^0) - F^*)}{R} + \frac{2(L+2)}{R} \sum_{r=0}^{R-1} \left\| \frac{1}{N} \sum_{i=1}^N (\theta_i^{r+1,0} \right. \\
&\quad \left. - \theta_i^{r,K}) \right\|^2 + 2(L+2)\eta^2 \psi_F^2 + 3(\sigma_F^2 + \psi_F^2) \\
&\stackrel{(b)}{\leq} \frac{2(F(\theta_G^0) - F^*)}{R} + \frac{2(L+2)}{NR} \sum_{r=0}^{R-1} \sum_{i=1}^N L_i^{\text{gap},r} \\
&\quad + 2(L+2)\eta^2 \psi_F^2 + 3(\sigma_F^2 + \psi_F^2), \tag{19}
\end{aligned}$$

where inequality (a) is obtained by using $F(\theta_G^R) \geq F^*$. Inequality (b) results from Lemma 1. This completes the proof. $\square$

Theorem 1 characterizes the impact of the neighbor selection decision on the convergence rate of our proposed DFL. In particular, the initial optimality gap represents the difference between the initial loss of the global model and the optimal loss, while the gradient-induced gap captures the noise introduced by stochastic gradients and the step size. These two terms are independent of the decision variables. On the other hand, the consensus distance induced gap depends on the neighbor selection decisions $s_{i,j}^r$.

Inspired by the aforementioned observations, we are motivated to reduce the consensus distance in each training round in order to improve the learning performance of our proposed DFL. Notably, when each client selects all of other clients in each training round (i.e., $s_{i,j}^r = 1, i, j \in \mathcal{N}, 0 \leq r \leq R-1$), the consensus distance induced gap reduces to 0. The corresponding convergence rate is maximized. However, the D2D network topologies in practical systems are not often fully connected due to system characteristics (e.g., wireless channel conditions). As a result, each client cannot select all of other clients for model aggregation. In addition, we will show in Section II-C that the corresponding system cost (i.e., delay and energy consumption on training) increases significantly with an increase of the selected neighboring clients for each client. Hence, it is crucial that the neighboring clients for each client are selected properly to maintain a balance between the learning performance and the system cost of DFL.

C. Delay and Energy Consumption

Each client needs to select its neighboring clients, CPU frequency, and transmit power in each training round. The chosen values determine the delay and energy consumption on training.

1) *Delay*: The total delay includes the computation and communication delay of all clients. The computation delay of a client corresponds to the time required for local training. Let decision variable $f_i^r \in [f_i^{\min}, f_i^{\max}]$ denote the CPU frequency chosen by client i in the r -th training round, where $f_i^{\min}$ and $f_i^{\max}$ are the minimum and maximum CPU frequency of client i , respectively. Let c_i denote the number of CPU cycles that client i required to process one bit of data. Let ϵ_i denote the size of the dataset $\mathcal{D}_i$ (in bits). The computation delay of client i to perform K SGD steps in the r -th training round is given by $T_i^{\text{comp},r} = \frac{c_i \epsilon_i K}{f_i^r}$.

When client i has completed its local training in one training round, it sends its updated local model to its selected neighbors, and the communication delay is incurred. Let decision variable $p_i^r \in [0, p_i^{\max}]$ denote the transmit power of client i in the r -th training round, where $p_i^{\max}$ denotes the maximum transmit power of client i . Let $g_{i,j}^r$ denote the channel gain from client i to client j in the r -th training round. At an arbitrary time instant t in the r -th training round, we define the set of clients that are transmitting their updated models as $\mathcal{M}(t) \subset \mathcal{V}$. The SINR of the link from client i to client j at time instant t in the r -th training round is given by

$$\psi_{i,j}^r(t) = \frac{|g_{i,j}^r|^2 p_i^r}{\sum_{k \in \mathcal{M}(t) \setminus \{i\}} |g_{k,j}^r|^2 p_k^r + BN_0}, \quad (20)$$

where N_0 is the noise spectral density. The achievable data rate at time instant t in the r -th training round is $C_{i,j}^r(t) = \text{Blog}_2(1 + \psi_{i,j}^r(t))$, where B is the bandwidth.

We consider a synchronized DFL scheme, where all clients begin their local training in each training round simultaneously.¹ When client i has completed its local training, it transmits local model parameters to its selected neighbors. At the same time, client i receives model parameters from them. Note that the interference may change over time. This is because the selected neighbors may use different transmit power, and their channel gains can be different. The selected neighbors may complete sending model parameters to client i at different time instants. Let $\tau_{j,i}^r$ denote the time that client i begins to receive the model from client j in the r -th training round. Since clients' model structures are identical, let χ_m denote the size of the model (in bits). The corresponding communication delay is denoted as $T_{j,i}^{\text{tr},r}$. From time $\tau_{j,i}^r$ to $\tau_{j,i}^r + T_{j,i}^{\text{tr},r}$, the size of the cumulative transmitted data for sending the model parameters from the selected neighboring client j to client i is equal to the size of the model. It satisfies $\int_{\tau_{j,i}^r}^{\tau_{j,i}^r + T_{j,i}^{\text{tr},r}} C_{j,i}^r(t) dt = \chi_m$.

As illustrated in Fig. 2, where the adjacent relationship between clients follows from Fig. 1, when client i has completed its local training before its selected neighbors, client i waits until they are finished. When the selected neighbors have completed their local training, client i receives model

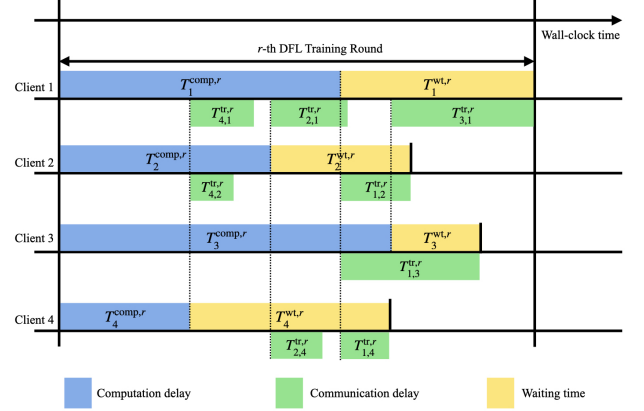


Fig. 2: An illustration of communication delay and computation delay in one DFL training round.

parameters from them. Furthermore, when the selected neighbors of client i complete their local training before client i , client i receives their local model parameters and stores them until it has completed its own local training. Then, client i aggregates its local model with the models received from its selected neighbors.

We define the waiting time of client i in the r -th training round as the duration starting from the time when client i has completed its local training to the time when it has received the model parameters from all the selected neighbors. Let $\mathbf{p}^r \triangleq (p_1^r, p_2^r, \dots, p_N^r)$ and $\mathbf{f}^r \triangleq (f_1^r, f_2^r, \dots, f_N^r)$ denote the vectors of transmit power and CPU frequency in the r -th training round, respectively. The waiting time of client i in the r -th training round can be represented as a function of neighbor selection decisions of client i , the transmit power of all clients, and the CPU frequency of all clients. It is given by

$$T_i^{\text{wt},r}(\mathbf{p}^r, \mathbf{f}^r, \mathbf{s}_i^r) = \max \left\{ 0, \max_{j \in \mathcal{N}_i^r} \{T_j^{\text{comp},r} + T_{j,i}^{\text{tr},r}\} - T_i^{\text{comp},r} \right\}. \quad (21)$$

The total delay of client i in the r -th training round is denoted as T_i^r , which is defined as the duration from the time when client i begins its local training to the time when it has received the model parameters from its selected neighbors. T_i^r is equal to the sum of the waiting time of client i in the r -th training round $T_i^{\text{wt},r}$ and the computation delay of client i . That is,

$$T_i^r(\mathbf{p}^r, \mathbf{f}^r, \mathbf{s}_i^r) = T_i^{\text{wt},r} + T_i^{\text{comp},r} = \max \left\{ T_i^{\text{comp},r}, \max_{j \in \mathcal{N}_i^r} \{T_j^{\text{comp},r} + T_{j,i}^{\text{tr},r}\} \right\}. \quad (22)$$

The total delay of all clients in the r -th training round is defined as the maximum value of the total delay of all clients. That is, $T^r(\mathbf{p}^r, \mathbf{f}^r, \mathbf{S}^r) = \max_{i \in \mathcal{V}} \{T_i^r\}$.

2) *Energy consumption*: The energy consumption includes the computation and communication energy consumption. For the computation energy consumption, we follow the settings in [43]. The computation energy consumption of K SGD steps

¹In our DFL framework, a coordinator is required to facilitate synchronization during the training process of DFL. The coordinator can be deployed at each client or on a virtual server, as described in [42]. The coordinator sends synchronization messages to all clients to ensure that all clients align their training processes in each training round. The size of the synchronization message is relatively small compared to the model parameters. Hence, the communication costs of synchronization are considered to be negligible.

in client i is given by $E_i^{\text{comp},r} = K\phi c_i D_i(f_i^r)^2$, where ϕ is the capacitance coefficient of each client. The communication energy consumption of client i in the r -th training round is denoted as $E_i^{\text{tr},r}$, which is given by $E_i^{\text{tr},r} = p_i^r \max_{j \in \mathcal{N}_i^r} \{T_{i,j}^{\text{tr},r}\}$. The total energy consumption of client i in the r -th training round is the sum of computation and communication energy consumption. It is given by $E_i^r = E_i^{\text{comp},r} + E_i^{\text{tr},r}$. Similarly, we can represent the energy consumption of all clients in the r -th training round as a function of $\mathbf{p}^r$, $\mathbf{f}^r$, and $\mathbf{S}^r$, denoted as $E^r(\mathbf{p}^r, \mathbf{f}^r, \mathbf{S}^r) = \sum_{i=1}^N E_i^r$.

D. Problem Formulation

We aim to jointly optimize the neighbor selection and resource allocation decisions to minimize the total delay and energy consumption on training and the consensus distance of local models in DFL. First, we characterize the system cost. Let U^r denote the system cost function in the r -th training round, which is a weighted sum of the total delay and energy consumption. We have $U^r(\mathbf{p}^r, \mathbf{f}^r, \mathbf{S}^r) = T^r(\mathbf{p}^r, \mathbf{f}^r, \mathbf{S}^r) + \lambda E^r(\mathbf{p}^r, \mathbf{f}^r, \mathbf{S}^r)$, where λ is a non-negative coefficient. We denote $F^{\text{obj},r}(\mathbf{p}^r, \mathbf{f}^r, \mathbf{S}^r) = U^r(\mathbf{p}^r, \mathbf{f}^r, \mathbf{S}^r) + \beta \sum_{i=1}^N L_i^{\text{gap},r}(\mathbf{s}_i^r)$ as the objective function, where β is a non-negative coefficient. Since the neighbor selection and resource allocation decisions are independent across training rounds, we decouple the optimization problem of all training rounds into R optimization problems, one in each training round. In the r -th training round, the problem can be formulated as

$$\underset{\mathbf{p}^r, \mathbf{f}^r, \mathbf{S}^r}{\text{minimize}} \quad F^{\text{obj},r}(\mathbf{p}^r, \mathbf{f}^r, \mathbf{S}^r) \quad (23a)$$

$$\text{subject to} \quad s_{i,j}^r \leq a_{i,j}^r, \quad i, j \in \mathcal{V}, \quad (23b)$$

$$0 \leq p_i^r \leq p_i^{\text{max}}, \quad i \in \mathcal{V}, \quad (23c)$$

$$f_i^{\text{min}} \leq f_i^r \leq f_i^{\text{max}}, \quad i \in \mathcal{V}. \quad (23d)$$

Problem (23) aims to determine the neighbor selection matrix $\mathbf{S}^r$, transmit power vector $\mathbf{p}^r$, and CPU frequency vector $\mathbf{f}^r$. Constraint (23b) ensures that client i can send its model to its neighbor j only if there is a communication link between them. Constraint (23c) corresponds to the transmit power limit of each client. Constraint (23d) ensures that the CPU frequency of each client is bounded. Solving this mixed-integer nonlinear programming problem is challenging due to the following reasons. First, the time-varying channel conditions and dynamic topology changes introduce additional complexity, which requires a robust algorithm that can adapt to these dynamic conditions. Second, the requirement of decentralized implementation across clients complicates the solution. This is because in decentralized settings, the resource allocation decisions of each client depend on the communication and computation capabilities of its neighboring clients and the corresponding channel gains. Third, handling the coupling of the neighbor selection and resource allocation decisions is difficult because the neighbor selection decisions made by each client directly affect the total delay and energy consumption on training as well as the learning performance of DFL. Hence, we develop an attention-based GNN in the following section to select neighbors for each client and solve problem (23).

III. ALGORITHM DESIGN

We propose a multi-head attention-based GNN for each client $i \in \mathcal{V}$. The graph attention mechanism has the following advantages. First, the attention mechanism determines the attention scores of neighbors which reflect the relevance between clients. Second, it prioritizes the features of neighboring clients with higher attention scores and can learn the intrinsic properties of the graph. Finally, the attention mechanism can be implemented in a decentralized manner, which satisfies the requirements of decentralized implementation.

As shown in Fig. 3, our proposed GNN has four modules: a client encoder, a local model encoder, a neighbor selection module, and a decoder. The client encoder encodes the client features (i.e., maximum transmit power, maximum and minimum CPU frequencies) and edge features (i.e., channel gains) to capture the intrinsic properties of the D2D networks. The local model encoder generates embeddings of the model parameters. The neighbor selection module determines the set of neighbors for model aggregation. The decoder determines the transmit power and CPU frequency of the client.

For illustration simplicity, we drop the superscript r in the following subsections. Let $\mathcal{N}_i = \mathcal{N}_i \cup \{i\}$ denote the modified neighbor set of client i . Let $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ denote the identity matrix. We define matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ for attention score calculation. Let $\tilde{\mathbf{a}}_i$ denote the i -th row vector of matrix $\tilde{\mathbf{A}}$. Let $\mathbf{x}_i^{\text{ndin}} = (p_i^{\text{max}}, f_i^{\text{max}}, f_i^{\text{min}})$ denote client i 's node feature vector. The edge feature $x_{i,j}^{\text{ein}}$ is the magnitude of the channel gain of link $(i, j) \in \mathcal{E}$, i.e., $|g_{i,j}|$. Since the node features have different orders of magnitude, we perform normalization such that the mean and standard deviation of the features are zero and one, respectively. The m -th normalized node feature in $\mathbf{x}_i^{\text{ndin}}$ is given by $x_{i,m}^{\text{nd}} = (x_{i,m}^{\text{ndin}} - \mu_m^{\text{nd}})/\sigma_m^{\text{nd}}$, $1 \leq m \leq 3$, where μ_m^{nd} and σ_m^{nd} are the mean and standard deviation of the m -th node feature. The normalized edge feature of $x_{i,j}^{\text{ein}}$ is obtained by $x_{j,i}^{\text{e}} = (x_{j,i}^{\text{ein}} - \mu^{\text{e}})/\sigma^{\text{e}}$, where μ^{e} and σ^{e} are the mean and standard deviation of edge features, respectively.

A. Client Encoder

We consider encoding both the node and edge features. This is because the node features characterize the communication and computational capabilities of clients. The edge features characterize the channel gains which affect the communication delay. The client encoder utilizes these two features to capture the intrinsic properties of the network topology. It generates the edge-enhanced node embeddings which combine the node features and edge features.

Node Attention Module: In this module, we introduce the node attention score to assess the importance of node j 's feature $\mathbf{x}_j^{\text{nd}}$ to node i 's feature $\mathbf{x}_i^{\text{nd}}$ and generate the node embeddings. Only the neighboring nodes are considered when calculating the node attention scores for each node. We employ a multi-head attention mechanism with a predefined K_h independent sets of trainable weights of GNN. The k_h -th set

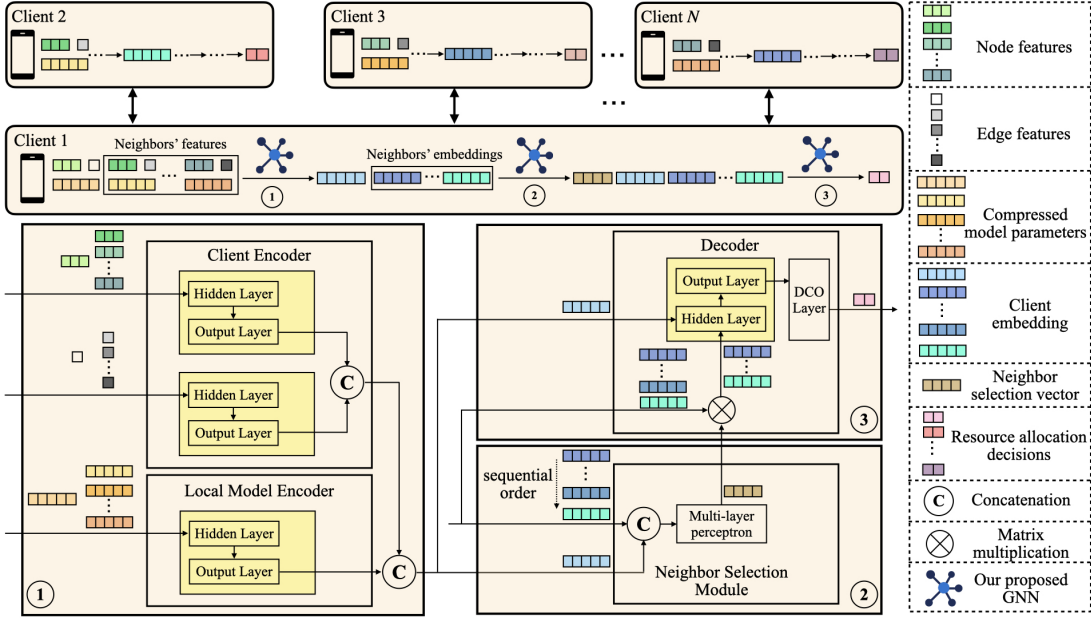


Fig. 3: A schematic illustration of our proposed GNN for DFL. This is an example of our proposed GNN deployed at client 1.

of node attention scores generated by the hidden layer are

$$\alpha_{i,j}^{k_h} = \frac{\exp\left(\sigma\left((\mathbf{a}^{\text{ndm},k_h})^\top [\mathbf{W}^{\text{ndm},k_h} \mathbf{x}_i^{\text{nd}} \parallel \mathbf{W}^{\text{ndm},k_h} \mathbf{x}_j^{\text{nd}}]\right)\right)}{\sum_{k \in \tilde{\mathcal{N}}_i} \exp\left(\sigma\left((\mathbf{a}^{\text{ndm},k_h})^\top [\mathbf{W}^{\text{ndm},k_h} \mathbf{x}_i^{\text{nd}} \parallel \mathbf{W}^{\text{ndm},k_h} \mathbf{x}_k^{\text{nd}}]\right)\right)}, \quad (24)$$

where $\sigma(\cdot)$ is a nonlinear activation function. It is a leaky rectified linear unit in our proposed GNN. $\mathbf{a}^{\text{nd},k_h} \in \mathbb{R}^{2D_{\text{ndm}}}$ is the k_h -th set of weights of the fully connected layer in the hidden layer. $\mathbf{W}^{\text{ndm},k_h} \in \mathbb{R}^{D_{\text{ndm}} \times 3}$ is the k_h -th set of weights for the node feature in the hidden layer. D_{ndm} is the dimension of the output embedding $\mathbf{h}_i^{\text{ndm}}, i \in \mathcal{V}$ in the hidden layer. In particular, each client calculates K_h independent node attention scores in parallel by utilizing the multi-head graph attention mechanism. Then, in the hidden layer, the attention scores are used to generate the output by performing concatenation as follows:

$$\mathbf{h}_i^{\text{ndm}} = \big\|_{k_h=1}^{K_h} \sigma\left(\sum_{j \in \tilde{\mathcal{N}}_i} \alpha_{i,j}^{k_h} \mathbf{W}^{\text{ndm},k_h} \mathbf{x}_j^{\text{nd}}\right). \quad (25)$$

At the output layer of the attention module, we use $\mathbf{h}_i^{\text{ndm}}$ as the input and generate the embedding of node i , which is denoted as $\mathbf{h}_i^{\text{nd}}$. It is given by

$$\mathbf{h}_i^{\text{nd}} = \sigma\left(\frac{1}{K_h} \sum_{k_h=1}^{K_h} \sum_{j \in \tilde{\mathcal{N}}_i} \alpha_{i,j}^{o,k_h} \mathbf{W}^{\text{ndo},k_h} \mathbf{h}_j^{\text{ndm}}\right), \quad (26)$$

where $\alpha_{i,j}^{o,k_h}$ denotes the k_h -th set of node attention scores generated by the output layer. $\mathbf{W}^{\text{ndo},k_h}$ denotes the k_h -th set of weights for the node feature in the output layer. Let $\mathbf{A}^{\text{ndm}} \in \mathbb{R}^{2D_{\text{ndm}} \times K_h}$ and $\mathbf{A}^{\text{ndo}} \in \mathbb{R}^{2D_{\text{ndm}} K_h \times K_h}$ denote the K_h sets of trainable weights of fully-connected layers in the hidden layer and output layer, respectively. Let $\mathbf{W}^{\text{ndm}} \in \mathbb{R}^{D_{\text{ndm}} \times 3 \times K_h}$ and $\mathbf{W}^{\text{ndo}} \in \mathbb{R}^{D_{\text{ndo}} \times D_{\text{ndm}} K_h \times K_h}$ denote the K_h sets of weights for node features in the hidden layer and output layer of

the node attention module, respectively. Let $\mathbf{X}^{\text{nd}} \in \mathbb{R}^{3 \times N}$ denote all node features. For simplicity, we represent the node embedding generation process in (24) – (26) as a function of $\mathbf{A}^{\text{ndm}}, \mathbf{A}^{\text{ndo}}, \mathbf{W}^{\text{ndm}}, \mathbf{W}^{\text{ndo}}, \mathbf{X}^{\text{nd}}$, and $\tilde{\mathbf{a}}_i$ as

$$\mathbf{h}_i^{\text{nd}} = \text{Att}(\mathbf{A}^{\text{ndm}}, \mathbf{A}^{\text{ndo}}, \mathbf{W}^{\text{ndm}}, \mathbf{W}^{\text{ndo}}, \mathbf{X}^{\text{nd}}, \tilde{\mathbf{a}}_i). \quad (27)$$

Edge Attention Module: Let $\mathbf{h}_i^e \in \mathbb{R}^{D_{\text{eo}}}$ denote the edge embedding of node i , where D_{eo} is the dimension of $\mathbf{h}_i^e$. Let D_{em} denote the dimension of the output embedding in the hidden layer of the edge attention module. By using the attention mechanism, $\mathbf{h}_i^e$ generated by the hidden layer can be expressed as $\mathbf{h}_i^e = \text{Att}(\mathbf{A}^{\text{em}}, \mathbf{A}^{\text{eo}}, \mathbf{W}^{\text{em}}, \mathbf{W}^{\text{eo}}, \mathbf{x}^e, \tilde{\mathbf{a}}_i)$, where $\mathbf{A}^{\text{em}} \in \mathbb{R}^{2D_{\text{em}} \times K_h}$ and $\mathbf{A}^{\text{eo}} \in \mathbb{R}^{2D_{\text{em}} K_h \times K_h}$ denote the K_h sets of trainable weights of fully-connected layers in the hidden layer and output layer of the edge attention module, respectively. $\mathbf{W}^{\text{em}} \in \mathbb{R}^{D_{\text{em}} \times 1 \times K_h}$ and $\mathbf{W}^{\text{eo}} \in \mathbb{R}^{D_{\text{eo}} \times D_{\text{em}} K_h \times K_h}$ denote the K_h sets of weights for edge features in the hidden layer and output layer of the edge attention module, respectively. $\mathbf{x}^e \in \mathbb{R}^{|\mathcal{E}|}$ denotes the edge features of all clients.

The edge-enhanced node embedding of client i is obtained by concatenating the node and edge embeddings, which is given by $\mathbf{h}_i = (\mathbf{h}_i^{\text{nd}} \parallel \mathbf{h}_i^e) \in \mathbb{R}^{D_{\text{ndo}} + D_{\text{eo}}}$.

B. Local Model Encoder

The aggregated model of client i in DFL can be represented as a function of the model parameters of client i and its neighbors, i.e., θ_i and $\theta_j, j \in \mathcal{N}_i$. To obtain the information on the local model of each client, we design a local model encoder. This encoder can learn the model discrepancies between clients, which contributes to achieving a high degree of consensus across clients' local models. However, as local models often have a large number of parameters, transmitting all of them between clients for training can incur substantial communication overhead, potentially impacting communication efficiency. Thus, we reduce the number of transmitted

local model parameters between clients in our proposed GNN by employing a random sampling technique which is simple to implement. By using random sampling, each client randomly selects a fraction of local model parameters based on a predefined sampling ratio, which is denoted as κ . Note that κ can be regarded as a measure of sparsity. When the value of κ is small, the transmitted local model parameters for GNN training will be sparse. By using random sampling, the communication cost for transmitting model parameters between clients in GNN can be reduced.

To characterize the random sampling process, we generate a mask $\mathbf{m} \in \mathbb{R}^{D_m}$. In particular, $\mathbf{m}$ satisfies $\mathbf{1}^\top \mathbf{m} = \lfloor \kappa D_m \rfloor$. For each element $m_l, 1 \leq l \leq D_m$, we have $m_l = 1$ if the l -th local model parameter is sampled. Otherwise, $m_l = 0$. Therefore, the compressed model parameters of client i can be expressed as $\mathbf{x}_i^{\text{pa}} = \boldsymbol{\theta}_i \odot \mathbf{m}$, which will be used for generating parameter embeddings. Let $\mathbf{X}^{\text{pa}} \in \mathbb{R}^{\lfloor \kappa D_m \rfloor \times N}$ denote the compressed model parameters of all clients. Let $\mathbf{h}_i^{\text{pa}} \in \mathbb{R}^{D_{\text{pao}}}$ denote the parameter embedding of client i , where D_{pao} is the dimension of $\mathbf{h}_i^{\text{pa}}$. Let D_{pam} denote the dimension of the output embedding in the hidden layer of the parameter attention module. $\mathbf{h}_i^{\text{pa}}$ can be obtained by $\mathbf{h}_i^{\text{pa}} = \text{Att}(\mathbf{A}^{\text{pam}}, \mathbf{A}^{\text{pao}}, \mathbf{W}^{\text{pam}}, \mathbf{W}^{\text{pao}}, \mathbf{X}^{\text{pa}}, \tilde{\mathbf{a}}_i)$, where $\mathbf{A}^{\text{pam}} \in \mathbb{R}^{2D_{\text{pam}} \times K_h}$ and $\mathbf{A}^{\text{pao}} \in \mathbb{R}^{2D_{\text{pam}}K_h \times K_h}$ denote the K_h sets of trainable weights of fully-connected layers in the hidden layer and output layer of the parameter attention module, respectively. $\mathbf{W}^{\text{pam}} \in \mathbb{R}^{D_{\text{pam}} \times \lfloor \kappa D_m \rfloor \times K_h}$ and $\mathbf{W}^{\text{pao}} \in \mathbb{R}^{D_{\text{pao}} \times D_{\text{pam}}K_h \times K_h}$ denote the K_h sets of weights for the compressed parameters in the hidden layer and output layer of the parameter attention module, respectively.

After generating edge-enhanced node embedding and parameter embedding by the output layers, we concatenate them into one embedding as the input to the neighbor selection module. The concatenated embedding is denoted as $\mathbf{h}_i^{\text{mu}} = (\mathbf{h}_i \parallel \mathbf{h}_i^{\text{pa}}) \in \mathbb{R}^{D_{\text{ndo}} + D_{\text{eo}} + D_{\text{pao}}}$.

C. Neighbor Selection Module

Since the full participation scheme may incur high delay and energy consumption in DFL, we propose a neighbor selection module, which selects a subset of neighbors for each client. The inputs to the module are the concatenated embedding of a client and the embeddings of its neighbors. They are sent to the module sequentially. The concatenated embedding of client i and its neighbor j is given by $\mathbf{h}_{i,j}^s = (\mathbf{h}_i^{\text{mu}} \parallel \mathbf{h}_j^{\text{mu}}) \in \mathbb{R}^{2(D_{\text{ndo}} + D_{\text{eo}} + D_{\text{pao}})}$.

Then, we apply a multi-layer perceptron (MLP) module to determine whether a certain neighbor will be selected. The output selection is given by $b_{i,j} = \sigma_s((\mathbf{w}_i^s)^\top \mathbf{h}_{i,j}^s)$, where $\mathbf{w}_i^s \in \mathbb{R}^{2(D_{\text{ndo}} + D_{\text{eo}} + D_{\text{pao}})}$ denotes the weights of the neighbor selection module in GNN. $\sigma_s(\cdot)$ is the sigmoid function that generates the normalized probability vector. Given $b_{i,j}$, client i determines the selection decision $s_{i,j}$ based on a predefined threshold $\gamma \in (0, 1)$. Since the bi-directional model exchange is required for aggregation in DSGD, we have

$$s_{i,j} = \mathbb{1}(b_{i,j} \geq \gamma \text{ and } b_{j,i} \geq \gamma). \quad (28)$$

After all neighbors' embeddings have been sent to the neighbor selection module, client i generates the selection vector $\mathbf{s}_i$. The neighbor selection matrix $\mathbf{S}$ can then be constructed.

D. Decoder

By using the aforementioned edge-enhanced node embeddings and the neighbor selection vector, the decoder determines the transmit power and CPU frequency for client i . We define $\hat{\mathbf{S}} = \mathbf{S} + \mathbf{I}_N$. Let $\hat{\mathbf{u}}_i = (\hat{p}_i, \hat{f}_i)$ denote the resource allocation decision of client i . It is obtained by $\hat{\mathbf{u}}_i = \text{Att}(\hat{\mathbf{A}}^{\text{ndm}}, \hat{\mathbf{A}}^{\text{ndo}}, \hat{\mathbf{W}}^{\text{ndm}}, \hat{\mathbf{W}}^{\text{ndo}}, \mathbf{H}, \hat{\mathbf{s}}_i)$, where $\hat{\mathbf{A}}^{\text{ndm}} \in \mathbb{R}^{2D_{\text{ndm}} \times K_h}$ and $\hat{\mathbf{A}}^{\text{ndo}} \in \mathbb{R}^{4 \times K_h}$ denote the K_h sets of weights of fully connected layers in the hidden layer and output layer of the attention module, respectively. $\hat{\mathbf{W}}^{\text{ndm}} \in \mathbb{R}^{D_{\text{ndm}} \times 2(D_{\text{ndo}} + D_{\text{eo}}) \times K_h}$ and $\hat{\mathbf{W}}^{\text{ndo}} \in \mathbb{R}^{D_{\text{ndo}} \times D_{\text{ndm}}K_h \times K_h}$ denote the K_h sets of weights for the edge-enhanced node embeddings in the hidden layer and output layer of the attention module, respectively. $\mathbf{H} \in \mathbb{R}^{(D_{\text{ndo}} + D_{\text{eo}}) \times N}$ denotes the edge-enhanced node embeddings of all clients.

Since the output $\hat{\mathbf{u}}_i$ is generated based on the embeddings, it may not satisfy constraints (23c) and (23d). Therefore, we include a differentiable convex optimization (DCO) layer to the last layer of the decoder in our proposed GNN to determine the feasible resource allocation decision $\mathbf{u}_i = (p_i, f_i)$. This can be achieved by projecting $\hat{\mathbf{u}}_i$ onto the feasible domain while preserving the gradient flow during GNN training. With the DCO layer, we can train our proposed GNN directly and obtain feasible decisions efficiently.

E. Loss Function of GNN

We define the loss function of our proposed GNN as a weighted sum of the system cost and the consensus distance. We approximate the consensus distance of model parameters by using the aggregated compressed model parameters of each client. Given $\mathbf{s}_i$, the aggregated compressed model parameters of client i can be obtained by a weighted sum of the compressed model parameters of selected neighboring clients $j \in \hat{\mathcal{N}}_i$ and itself, which can be represented as a function of $\mathbf{s}_i$ as $\bar{\mathbf{x}}_i^{\text{pa}}(\mathbf{s}_i) = \sum_{j \in \hat{\mathcal{N}}_i \cup \{i\}} w_{i,j} \mathbf{x}_j^{\text{pa}}$. Moreover, we define the average compressed model parameters as $\bar{\mathbf{x}}^{\text{pa}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i^{\text{pa}}$. During GNN training, we calculate $\bar{\mathbf{x}}^{\text{pa}}$ by collecting $\mathbf{x}_i^{\text{pa}}, i \in \mathcal{V}$ at the coordinator.² Then, we can approximate L_i^{gap} based on the compressed model parameters. By using the random sampling technique with sampling ratio κ , L_i^{gap} can be approximated as a function of $\mathbf{s}_i$ as $L_i^{\text{gap}}(\mathbf{s}_i) \approx \left\| \tilde{\mathbf{x}}_i^{\text{pa}}(\mathbf{s}_i) - \bar{\mathbf{x}}^{\text{pa}} \right\|^2 / \kappa$. The detailed result of this approximation will be presented in Section IV-A.

After we have obtained $\mathbf{p}$ and $\mathbf{f}$, the delay and energy consumption can be calculated. The system cost can then be determined. The loss function of client i in GNN training is defined as

$$L_i^{\text{GNN}}(\mathbf{p}, \mathbf{f}, \mathbf{s}_i) = U_i(\mathbf{p}, \mathbf{f}, \mathbf{s}_i) + \beta L_i^{\text{gap}}(\mathbf{s}_i). \quad (29)$$

²The size of the compressed model parameters is relatively small. Hence, the communication costs of transmitting the compressed model parameters between clients are considered to be negligible.

Algorithm 1 Algorithm for GNN Training

- 1: **Input:** Total number of epochs N_{ep} , predefined threshold γ , randomly initialized network scenarios with different topologies determined by Ω , normalized node features $\mathbf{X}^{\text{nd}}$, normalized edge features $\mathbf{x}^{\text{e}}$, compressed model parameters $\mathbf{X}^{\text{pa}}$, and initialized GNN model Φ .
- 2: **For** $n = 1$ to N_{ep} **do**
- 3: Sample a mini-batch of N_{b} network scenarios.
- 4: **For** $i \in \mathcal{V}$ in parallel **do**
- 5: Sample local model parameters and obtain $\mathbf{x}_i^{\text{pa}}$.
- 6: Send $\mathbf{x}_i^{\text{pa}}$ to neighboring clients $j \in \mathcal{N}_i$ and receive the corresponding $\mathbf{x}_j^{\text{pa}}$.
- 7: Use the client encoder, local model encoder, and neighbor selection modules of Φ to determine the set of selected neighboring clients $\hat{\mathcal{N}}_i$ based on eq. (28).
- 8: Determine $\mathbf{u}_i$ by using the decoder module of Φ .
- 9: Obtain $\tilde{\mathbf{x}}_i^{\text{pa}}$ and calculate U_i .
- 10: **End for**
- 11: Calculate the average compressed model parameters. Estimate the consensus distance of model parameters.
- 12: Update the GNN model Φ by performing backpropagation based on the loss obtained by eq. (29).
- 13: **End for**
- 14: **Output:** Trained GNN model Φ^* .

F. GNN Training and Implementation Procedure

In practical systems, our proposed GNN should adapt to different network settings (i.e., topology changes and time-varying channel conditions). Hence, we train our proposed GNN with a set of D2D network scenarios to enable it to be adaptable to dynamic settings. In each scenario, we randomly place the clients in a given geographical area and generate the channel conditions. Mini-batches of D2D network scenarios are used to train our proposed GNN. Each client collects the neighbors' node features, model parameters, and edge features. It then determines the resource allocation decision. The loss function of each client in GNN can be calculated. Each client performs backpropagation to update the weights of the GNN based on its own loss function. Let $\mathbf{W}^{\text{s}} = \{\mathbf{w}_i^{\text{s}}, i \in \mathcal{V}\}$ denote the weights of the neighbor selection module. Let $\Phi = \{\mathbf{A}^{\text{ndm}}, \mathbf{A}^{\text{ndo}}, \mathbf{A}^{\text{em}}, \mathbf{A}^{\text{eo}}, \mathbf{A}^{\text{pam}}, \mathbf{A}^{\text{pao}}, \hat{\mathbf{A}}^{\text{ndm}}, \hat{\mathbf{A}}^{\text{ndo}}, \mathbf{W}^{\text{ndm}}, \mathbf{W}^{\text{ndo}}, \mathbf{W}^{\text{em}}, \mathbf{W}^{\text{eo}}, \mathbf{W}^{\text{s}}, \mathbf{W}^{\text{pam}}, \mathbf{W}^{\text{eo}}, \hat{\mathbf{W}}^{\text{ndm}}, \hat{\mathbf{W}}^{\text{ndo}}\}$ denote the parameters of GNN. The training algorithm of our proposed GNN is given in Algorithm 1. After training, we use the trained GNN for implementation. Let Φ^* denote the parameters of the trained GNN. During implementation, we first initialize the transmit power and CPU frequency of each client to the maximum value. Then, we use our trained GNN to determine the transmit power and CPU frequency for each client in each training round.

IV. PERFORMANCE EVALUATION

We consider a scenario where the clients are randomly located in a $2 \times 2 \text{ km}^2$ coverage area. To model the mobility of clients, each client can move in an arbitrary direction. The traveling distance of each client $i \in \mathcal{V}$ between two training rounds follows $\Delta d_i \sim \mathcal{U}[0, 100 \text{ m}]$. To depict the heterogeneity of clients, the maximum transmit power of each client i follows $p_i^{\text{max}} \sim \mathcal{U}[6.667 \text{ mW}, 13.333 \text{ mW}]$. The minimum and maximum CPU frequency of each client

Table I: List of key simulation parameters.

Para.	Value	Para.	Value	Para.	Value
B	1 MHz [50]	Ω	-25 dB	N_{ep}	200
K_h	4	K	20	ϕ	10^{-28} [51]
N_0	-174 dBm/Hz [50]	c_i	20	κ	0.03
D_{ndo}	3	D_{eo}	2	D_{pao}	20
λ	1	ϵ_i	20.88 Mbits	χ_{m}	160 kB
β	100	N_{s}	2000	N_{b}	100

i follow $f_i^{\text{min}} \sim \mathcal{U}[0.067 \text{ GHz}, 0.133 \text{ GHz}]$ and $f_i^{\text{max}} \sim \mathcal{U}[1.667 \text{ GHz}, 2.333 \text{ GHz}]$, respectively. We initialize the transmit power and CPU frequency of each client to the maximum value. We consider the Rayleigh fading channel model. Unless stated otherwise, the number of clients N , the number of training rounds R , the learning rate η , and the hyperparameter γ in eq. (28) are set to 30, 300, 0.001, and 0.3, respectively. A list of key simulation parameters is given in Table I.

We use Fashion-MNIST [44], CIFAR-10 [45], and ImageNet [46] as the multi-class image classification datasets. We adopt the convolutional neural networks (CNNs), which have been used in [2], as clients' local models. We set $\eta = 0.001$. To study the effect of non-independent and identically distributed (non-IID) data partitioning, similar to other works in federated learning [47]–[49], we use the Dirichlet distribution $\text{Dir}(\rho)$ with parameter ρ to distribute the data with different labels among the clients. Let N_c denote the number of classes of a dataset. The Dirichlet distribution $\text{Dir}(\rho)$ generates the probability vector $\mathbf{q}_i = (q_i^1, q_i^2, \dots, q_i^{N_c})$, where q_i^n represents the fraction of data samples from class n assigned to client i , with $\sum_{n=1}^{N_c} q_i^n = 1$. The probability density function of the Dirichlet distribution is given by $f(\mathbf{q}_i; \rho) = \frac{\Gamma(N_c \rho)}{(\Gamma(\rho))^{N_c}} \prod_{n=1}^{N_c} (q_i^n)^{\rho-1}$, where $\Gamma(\cdot)$ is the gamma function. When ρ decreases, the data heterogeneity across clients increases and vice versa. Unless stated otherwise, ρ is set to 1. To ensure fairness, we allocate the same number of data samples to the local dataset of each client in all cases.

We compare the performance of our proposed algorithm with the following baseline schemes:

- Random sampling DFL (RS-DFL) scheme: Neighbors are randomly selected by each client based on a certain ratio ζ , which denotes the average neighbor selection ratio (i.e., the fraction of the average selected number of neighboring clients to all neighbors of all clients over all training rounds.) It is given by $\zeta = \frac{1}{RN} \sum_{r=0}^{R-1} \sum_{i=1}^N \frac{\|\mathbf{s}_i^r\|_1}{\|\mathbf{a}_i^r\|_1}$, where $\zeta \in [0, 1]$.
- Full participation DFL (FP-DFL) scheme: Each client performs local aggregation based on the model obtained from all of its neighbors. The transmit power and CPU frequencies are allocated to the clients using our proposed client encoder and decoder.
- FedHP scheme [27]: The network topology is optimized to minimize the delay on training. The transmit power and CPU frequencies are allocated to the clients using our proposed client encoder and decoder.
- Performance-based neighbor selection (PENS) scheme [28]: Each client selects the neighboring clients based on the similarity of local data distribution. The transmit

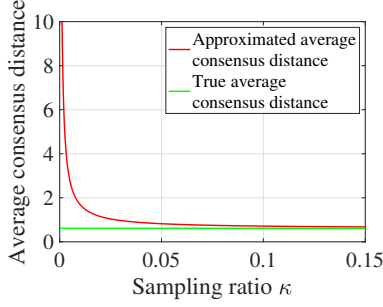


Fig. 4: Effect of the sampling ratio κ on the approximation of the average consensus distance.

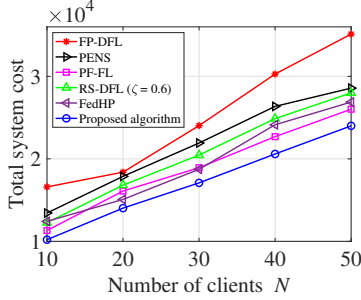


Fig. 5: Comparison of the total system cost versus the number of clients N on CIFAR-10 dataset.

power and CPU frequencies are allocated to the clients using our proposed client encoder and decoder.

- Path-following FL (PF-FL) scheme [39]: The joint delay and energy minimization problem (i.e., problem (23)) is solved by using the centralized path-following algorithm proposed in [39]. This scheme uses full participation of neighbors for model updates.

A. Effect of Sampling Ratio κ on the Approximation of the Average Consensus Distance

In Fig. 4, we compare the approximation of the average consensus distance to the ground-truth average consensus distance under different sampling ratios κ . It can be observed that with an increasing sampling ratio of the model parameters (e.g., from 0 to 0.03), the approximation of the average consensus distance across clients progressively approaches the true average consensus distance. However, further increasing κ (e.g., from 0.03 to 0.15) does not significantly contribute to minimizing the approximated average consensus distance. These results suggest that a small subset of compressed model parameters can effectively capture the differences in local model parameters across clients. Hence, by setting κ within the range from 0.03 to 0.15, we can well approximate the ground-truth average consensus distance while reducing communication costs of GNN training and implementation.

B. Computation Overhead Analysis of GNN Training

We compare the corresponding computation overhead of the training of our proposed GNN to the training of DFL. We use the number of floating-point operations (FLOPs) to measure the computation overhead. Results in Table II

Table II: Computation overhead comparison of our proposed DFL and GNN.

N	Our Proposed DFL (FLOPs)	Our Proposed GNN (FLOPs)
10	8.4×10^{10}	2.7×10^7
20	1.69×10^{11}	1.14×10^8
30	2.47×10^{11}	2.61×10^8
40	3.41×10^{11}	4.68×10^8
50	4.22×10^{11}	7.35×10^8

Table III: Comparison of total delay and energy consumption on the CIFAR-10 dataset with the number of clients $N = 50$.

	Delay on Training (s)	Energy Consumption on Training (J)
FP-DFL	6825	28298
RS-DFL	5834	22739
FedHP [27]	5573	21314
PENS [28]	6227	21764
PF-FL [39]	6017	19953
Proposed Algorithm	5195	18801

show that the number of FLOPs for training our proposed GNN is significantly smaller than the DFL training under different numbers of clients. Results show that the computation overhead of GNN training is negligible when compared to the subsequent DFL training process, which highlights the efficiency of our proposed GNN and its suitability for practical implementation.

C. Comparison of the Total System Cost, Total Delay, and Energy Consumption on Training

In Fig. 5, we compare the total system cost of all clients between different baseline schemes. For a fair comparison, we choose $\zeta = 0.6$ for RS-DFL scheme since it can achieve similar average testing accuracy to our proposed algorithm at this sampling ratio as shown in Fig. 6(b). It can be observed from Fig. 5 that our proposed algorithm achieves a lower total system cost than the other baselines under different number of clients. When N is equal to 50, our proposed algorithm achieves a total system cost, which is 25.93%, 16.00%, 14.26%, 10.74%, and 7.75% lower than FP-DFL, RS-DFL, PENS, FedHP, and PF-FL schemes, respectively. This indicates the effectiveness and scalability of our proposed algorithm in reducing the total system cost on training.

In Table III, we compare the total delay and energy consumption on training when the number of clients N is equal to 50. Results show that our proposed algorithm achieves a lower total delay on training when compared with FP-DFL, RS-DFL, PENS, FedHP, and PF-FL schemes. In addition, our proposed algorithm achieves the lowest total energy consumption on training among all schemes.

D. Comparison of the Average Testing Accuracy

In Fig. 6, we compare the average testing accuracy of all clients between different schemes. Results show that on Fashion-MNIST dataset, our proposed algorithm outperforms RS-DFL ($\zeta = 0.5$), PENS, and FedHP schemes in terms of the average testing accuracy by 3.77%, 0.94%, and 0.58%, respectively. On CIFAR-10 dataset, our proposed algorithm outperforms RS-DFL ($\zeta = 0.6$), PENS, and FedHP schemes in terms of the average testing accuracy by 2.0%, 1.7%, and 0.7%, respectively. It is worth noting that the value of ζ differs between datasets since RS-DFL randomly selects more clients

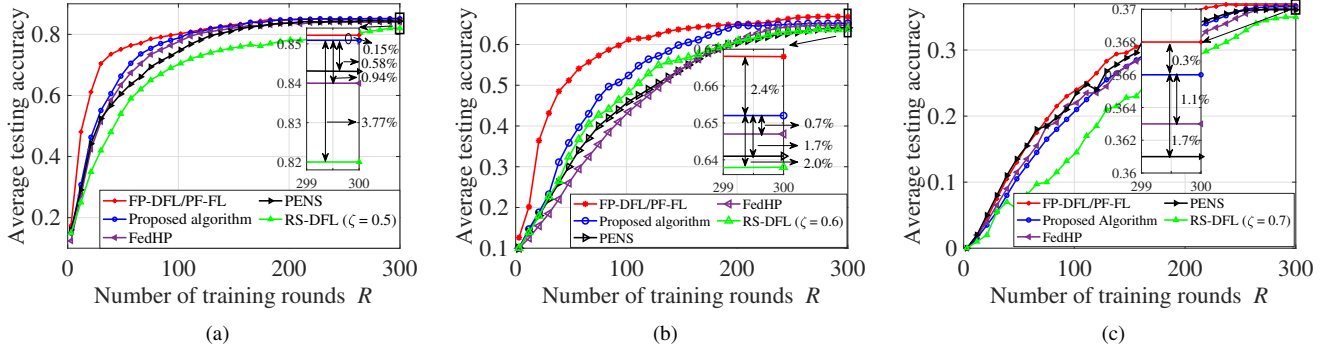


Fig. 6: Comparison of the average testing accuracy versus the number of training rounds R on (a) Fashion-MNIST, (b) CIFAR-10, and (c) ImageNet datasets.

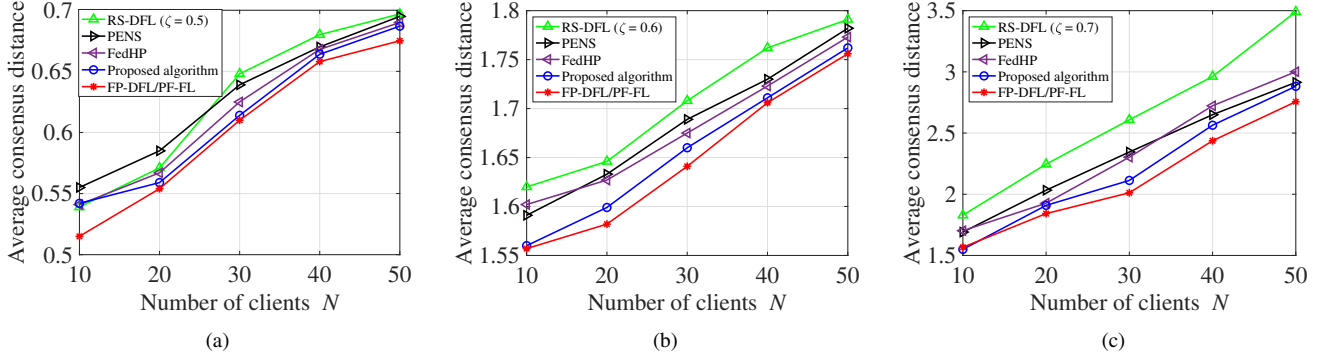


Fig. 7: Comparison of the average consensus distance versus the number of clients N on (a) Fashion-MNIST, (b) CIFAR-10, and (c) ImageNet datasets.

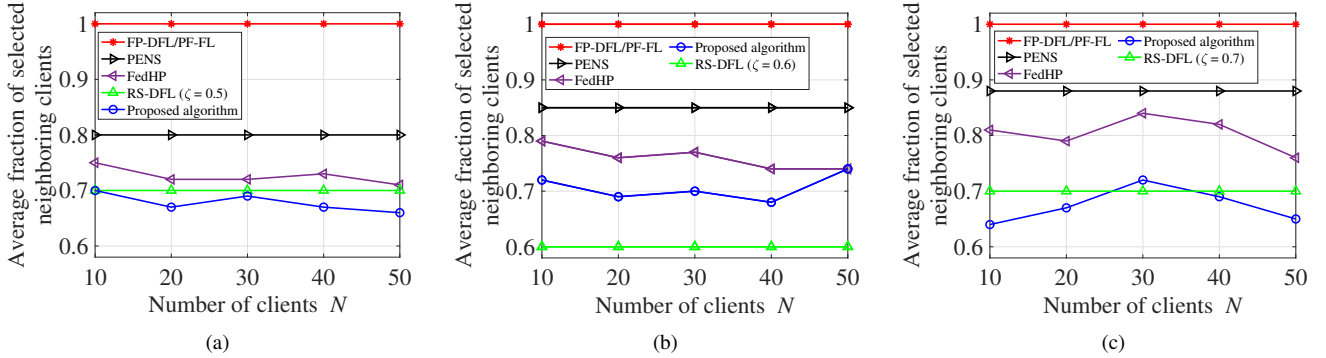


Fig. 8: Comparison of the average fraction of selected neighboring clients versus the number of clients N on (a) Fashion-MNIST, (b) CIFAR-10, and (c) ImageNet datasets.

(i.e., higher ζ) in more complex datasets to achieve a similar average testing accuracy to our proposed DFL. When using the full participation scheme, both PF-FL and FP-DFL achieve the highest accuracy. Nevertheless, our proposed algorithm still achieves an average testing accuracy comparable to these two full participation schemes, with a difference of 0.15%, 2.4%, and 0.3% on Fashion-MNIST, CIFAR-10, and ImageNet datasets, respectively.

Figs. 7 and 8 show the results on the average consensus distance and the average fraction of selected neighboring clients. Results show that our proposed algorithm outperforms PENS, FedHP, and RS-DFL schemes in terms of the average consensus distance in most cases. This highlights

the effectiveness of our proposed algorithm in enhancing the learning performance of DFL by reducing the consensus distance across clients. In addition, our proposed algorithm achieves a comparable average consensus distance to the full participation scheme with a significantly lower number of selected neighboring clients. Both PF-FL and FP-DFL schemes have the same average consensus distance and the average number of selected neighboring clients since they use the full participation approach.

E. Effect of Data Heterogeneity

We investigate the robustness of our proposed algorithm under different degrees of data heterogeneity across clients by

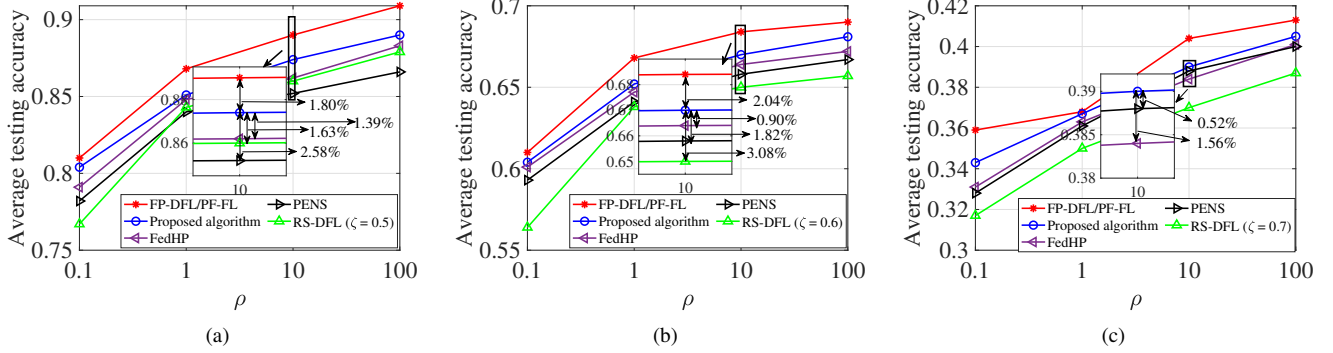


Fig. 9: Comparison of the average testing accuracy versus the degree of data heterogeneity ρ across clients' local datasets on (a) Fashion-MNIST, (b) CIFAR-10, and (c) ImageNet datasets.

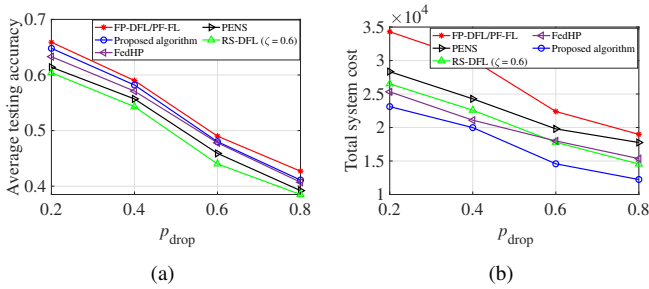


Fig. 10: Effect of p_{drop} on (a) the average testing accuracy and (b) the total system cost on CIFAR-10 dataset.

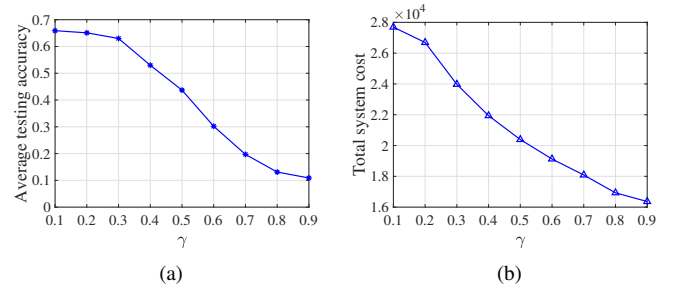


Fig. 11: Effect of γ on (a) the average testing accuracy and (b) the total system cost of the proposed algorithm on CIFAR-10 dataset.

showing the average testing accuracy under different values of hyperparameter ρ . A smaller value of ρ leads to higher data heterogeneity across clients. We set ρ to be 0.1, 1, 10, and 100 in order to control the degree of local data heterogeneity. Results in Fig. 9 show that our proposed algorithm consistently achieves the average testing accuracy comparable to that of the full participation schemes across all values of ρ . Our proposed algorithm outperforms FedHP, PENS, and RS-DFL schemes on all three datasets. In particular, on Fashion-MNIST dataset, when $\rho = 10$, our proposed algorithm outperforms PENS, RS-DFL ($\zeta = 0.5$), and FedHP schemes by 2.58%, 1.63%, and 1.39%, respectively. On ImageNet dataset, when $\rho = 10$, our proposed algorithm outperforms RS-DFL ($\zeta = 0.7$), FedHP, and PENS schemes by 7.44%, 1.56% and 0.52%, respectively. It is worth noting that the value of ζ differs between datasets since RS-DFL randomly selects more clients (i.e., higher ζ) in more complex datasets to achieve a similar average testing accuracy to our proposed DFL. This improvement across varying degrees of data heterogeneity underscores the robustness of our proposed algorithm to non-IID data.

F. Effect of Clients' Unavailability

In real-world scenarios, some clients may become unavailable during some training rounds and affect the learning performance and the total system cost of DFL. In Fig. 10, we evaluate the impact of the clients' unavailability on both the average testing accuracy and the total system cost of DFL. Similar to [52] [53], to model clients' unavailability, we define $p_{\text{drop}} \in [0, 1]$ as the probability that a client drops out in a

training round. Note that p_{drop} for each client is independent between training rounds. As p_{drop} increases, the network topology becomes more sparse. In addition, the average testing accuracy is calculated across all clients without considering dropout during testing. It ensures that the average testing accuracy reflects the overall effectiveness of the decentralized learning process. Results show that by increasing p_{drop} from 0.2 to 0.8, our proposed algorithm achieves a higher average testing accuracy when compared with FedHP, PENS, and RS-DFL schemes. In addition, our proposed algorithm achieves the lowest total system cost on training when compared with the baseline schemes. The results demonstrate the robustness of our proposed algorithm over the sparsity of network topology and clients' unavailability.

G. Impact of the Neighbor Selection Parameter γ

In Fig. 11, we present the results of the average testing accuracy and total system cost under different values of the neighbor selection parameter γ . The value of γ controls the average fraction of selected neighbors. When $\gamma = 0.1, 0.3$, and 0.9 , the average fraction of selected neighbors is 0.92, 0.74, and 0.11, respectively. This is because when γ increases, the chance of selecting a neighboring client for each client decreases. In particular, when γ increases from 0.1 to 0.3, the average fraction of selected neighbors decreases. Our proposed algorithm achieves a relatively high average testing accuracy. The total system cost on training decreases. This is because our proposed algorithm selects a smaller but appropriate number of neighbors for each client. On the other hand, selecting a

large number of neighbors for each client does not significantly improve the learning performance of DFL. Instead, it increases the communication and computation costs.

V. CONCLUSION

In this paper, we developed a DFL framework for cooperative model training in D2D networks. We analyzed the convergence performance of our proposed DFL. We proposed a GNN-based algorithm to minimize the total delay and energy consumption on training and optimize the learning performance of DFL in D2D networks. In particular, we developed an attention-based GNN model to determine the neighbor selection and resource allocation decisions. Furthermore, we trained our proposed GNN under different network topologies such that our proposed GNN could adapt to dynamic network settings. Results showed that our proposed algorithm could achieve a lower total delay and energy consumption on training when compared with five baseline schemes. In addition, our proposed algorithm could achieve similar learning performance as the full participation approach while outperforming RS-DFL, FedHP, and PENS schemes. We also showed the robustness of our proposed algorithm to different degrees of data heterogeneity across clients' datasets. For future work, we will address the issue of malicious participants in DFL.

REFERENCES

- [1] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *ArXiv preprint ArXiv:1610.02527*, Oct. 2016.
- [2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of Int'l Conf. on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, Apr. 2017.
- [3] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, and et al., "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, Jun. 2021.
- [4] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. of Machine Learning and Syst. (MLSys) Conf.*, Austin, TX, Mar. 2020.
- [5] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. of Conf. on Neural Info. Process. Syst. (NeurIPS)*, Dec. 2020.
- [6] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "BrainTorrent: A peer-to-peer environment for decentralized federated learning," *ArXiv preprint ArXiv:2106.02743*, May 2019.
- [7] Y. Shi, Y. Zhou, and Y. Shi, "Over-the-air decentralized federated learning," in *Proc. of IEEE Int'l Symp. on Info. Theory (ISIT)*, Jul. 2021.
- [8] S. Lu, Y. Zhang, and Y. Wang, "Decentralized federated learning for electronic health records," in *Proc. of Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2020.
- [9] S. Warnat-Herresthal, H. Schultze, K. L. Shastri, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, et al., "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, Jun. 2021.
- [10] Z. Lian, Q. Yang, W. Wang, Q. Zeng, M. Alazab, H. Zhao, and C. Su, "DEEP-FEL: Decentralized, efficient and privacy-enhanced federated edge learning for healthcare cyber physical systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3558–3569, Sep.-Oct. 2022.
- [11] Y. Tian, S. Wang, J. Xiong, R. Bi, Z. Zhou, and M. Z. A. Bhuiyan, "Robust and privacy-preserving decentralized deep federated learning training: Focusing on digital healthcare applications," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 21, no. 4, pp. 890–901, Jul. 2024.
- [12] J.-H. Chen, M.-R. Chen, G.-Q. Zeng, and J.-S. Weng, "BDFL: A Byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 8639–8652, Sep. 2021.
- [13] X. Hu, R. Li, L. Wang, Y. Ning, and K. Ota, "A data sharing scheme based on federated learning in IoV," *IEEE Trans. Veh. Technol.*, vol. 72, no. 9, pp. 11644–11656, Sep. 2023.
- [14] S. Savazzi, M. Nicoli, M. Bennis, S. Kianoush, and L. Barbieri, "Opportunities of federated learning in connected, cooperative, and automated industrial systems," *IEEE Commun. Mag.*, vol. 59, no. 2, pp. 16–21, Feb. 2021.
- [15] T. Ma, H. Wang, and C. Li, "Quantized distributed federated learning for industrial Internet of things," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3027–3036, Feb. 2023.
- [16] T. Wang, Y. Liu, X. Zheng, H.-N. Dai, W. Jia, and M. Xie, "Edge-based communication optimization for distributed federated learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2015–2024, Jul. 2022.
- [17] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning and permissioned blockchain for digital twin edge networks," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2276–2288, Feb. 2021.
- [18] Hewlett Packard Enterprise, "Swarm learning," <https://www.hpe.com/ca/en/what-is/swarm-learning.html>. Accessed: 2025-04-10.
- [19] S. S. Ram, A. Nedjic, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of Optimization Theory and Applications*, vol. 147, no. 3, pp. 516–545, Dec. 2010.
- [20] X. Li, W. Yang, S. Wang, and Z. Zhang, "Communication-efficient local decentralized SGD methods," *ArXiv preprint ArXiv:1910.09126*, Oct. 2019.
- [21] L. Kong, T. Lin, A. Koloskova, M. Jaggi, and S. U. Stich, "Consensus control for decentralized deep learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Jul. 2021.
- [22] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Proc. of Neural Info. Process. Syst. (NIPS)*, Montreal, Canada, Dec. 2015.
- [23] X. Lian, C. Zhang, H. Zhang, C. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. of Conf. on Neural Info. Process. Syst. (NIPS)*, Long Beach, CA, Dec. 2017.
- [24] M. Bornstein, T. Rabbani, E. Wang, A. S. Bedi, and F. Huang, "SWIFT: Rapid decentralized federated learning via wait-free model communication," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, Kigali, Rwanda, May 2023.
- [25] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "MATCHA: Speeding up decentralized SGD via matching decomposition sampling," in *Proc. of Indian Control Conf. (ICC)*, Hyderabad, India, Dec. 2019.
- [26] P. Zhou, Q. Lin, D. Loghin, B. C. Ooi, Y. Wu, and H. Yu, "Communication-efficient decentralized machine learning over heterogeneous networks," in *Proc. IEEE Int. Conf. Data Eng. (ICDE)*, Chania, Greece, Apr. 2021.
- [27] Y. Liao, Y. Xu, H. Xu, L. Wang, and C. Qian, "Adaptive configuration for heterogeneous participants in decentralized federated learning," in *Proc. of IEEE INFOCOM*, New York, NY, May 2023.
- [28] N. Onozko, G. Karlsson, O. Mogren, and E. L. Zec, "Decentralized federated learning of deep neural networks on non-IID data," in *Proc. of Int'l Workshop on Federated Learning for User Privacy and Data Confidentiality*, Jul. 2021.
- [29] M. Chen, Y. Xu, H. Xu, and L. Huang, "Enhancing decentralized federated learning for non-IID data on heterogeneous devices," in *Proc. IEEE Int. Conf. Data Eng. (ICDE)*, Anaheim, CA, Apr. 2023.
- [30] Z. Zhou, F. Sun, X. Chen, D. Zhang, T. Han, and P. Lan, "A decentralized federated learning based on node selection and knowledge distillation," *Mathematics*, vol. 11, no. 14, pp. 3162–3176, Jul. 2023.
- [31] M. Lee, G. Yu, and G. Y. Li, "Graph embedding-based wireless link scheduling with few training samples," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 4, pp. 2282–2294, Apr. 2021.
- [32] M. Eisen and A. Ribeiro, "Optimal wireless resource allocation with random edge graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 2977–2991, Apr. 2020.
- [33] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 101–115, Jan. 2021.
- [34] M. Lee, G. Yu, and H. Dai, "Decentralized inference with graph neural networks in wireless communication systems," *IEEE Trans. Mob. Comput.*, vol. 22, no. 5, pp. 2582–2598, May 2023.
- [35] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "SpreadGNN: Serverless multi-task federated learning for graph neural networks," in *Proc. AAAI Conf. Artificial Intelligence*, Feb. 2022.

- [36] S. Wang, M. Lee, S. Hosseinalipour, R. Morabito, M. Chiang, and C. G. Brinton, "Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation," in *Proc. of IEEE INFOCOM*, May 2021.
- [37] Z. Wang, Y. Zhou, Y. Zou, Q. An, Y. Shi, and M. Bennis, "A graph neural network learning approach to optimize RIS-assisted federated learning," *IEEE Trans. on Wirel. Commun.*, vol. 22, no. 9, pp. 6092–6106, Sep. 2023.
- [38] C. Meng, M. Tang, M. Setayesh, and V. W.S. Wong, "GNN-based neighbor selection and resource allocation for decentralized federated learning," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Kuala Lumpur, Malaysia, Dec. 2023.
- [39] V.-D. Nguyen, S. K. Sharma, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Efficient federated learning algorithm for resource allocation in wireless IoT networks," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3394–3409, Mar. 2021.
- [40] L. Xiao and S. P. Boyd, "Fast linear iterations for distributed averaging," *Syst. & Control. Lett.*, vol. 53, no. 1, pp. 65–78, Sep. 2004.
- [41] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, Apr. 2020.
- [42] S. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and Y. J. Guo, "Multi-timescale decentralized online orchestration of software-defined networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2716–2730, Dec. 2018.
- [43] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.
- [44] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *ArXiv preprint ArXiv 1708.07747*, Aug. 2017.
- [45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.
- [46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR)*, Miami, FL, Jun. 2009.
- [47] H.-Y. Chen and W.-L. Chao, "FedBE: Making bayesian model ensemble applicable to federated learning," in *Proc. Int. Conf. Learn. Representations (ICLR)*, May 2021.
- [48] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-IID data silos: An experimental study," in *Proc. IEEE Int. Conf. Data Eng. (ICDE)*, Kuala Lumpur, Malaysia, May 2022.
- [49] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu, "FedDC: Federated learning with non-IID data via local drift decoupling and correction," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR)*, New Orleans, LA, Jun. 2022.
- [50] S. Hosseinalipour, S. Wang, N. Michelusi, V. Aggarwal, C. G. Brinton, D. J. Love, and M. Chiang, "Parallel successive learning for dynamic distributed model training over heterogeneous wireless networks," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 222–237, Feb. 2024.
- [51] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.
- [52] X. Gu, K. Huang, J. Zhang, and L. Huang, "Fast federated learning in the presence of arbitrary device unavailability," in *Proc. Conf. Neural Info. Proc. Syst. (NeurIPS)*, Dec. 2021.
- [53] H. Wang and J. Xu, "Friends to help: Saving federated learning from client dropout," in *Proc. of IEEE Int. Conf. Acoust, Speech and Signal Process. (ICASSP)*, Seoul, Korea, Apr. 2024.



Chuiyang Meng (S'23) received the B.Eng. degree from the Harbin Institute of Technology, China, in 2018 and the M.Sc. degree from the Johns Hopkins University, U.S., in 2020. He is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering, The University of British Columbia (UBC), Vancouver, Canada. His research interests include distributed machine learning, network for AI, and optimization in wireless networks.



Ming Tang (S'16, M'18) is an Associate Professor in the Department of Computer Science and Engineering at Southern University of Science and Technology, Shenzhen, China. She received her Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, China, in 2018. She was a postdoctoral research fellow in The University of British Columbia, Vancouver, Canada, from 2018 to 2021. Her research interests include mobile edge computing, federated learning, and network economics. She received the 2022 Best Paper Award from *IEEE Transactions on Mobile Computing*. She recently serves as an Associate Editor in *IEEE Internet of Things Journal* and *IEEE Open Journal of the Communications Society*.



Mehdi Setayesh (S'20, M'24) received the B.Sc. and M.Sc. degrees from the Sharif University of Technology, Tehran, Iran, in 2014 and 2016, respectively, and the Ph.D. degree from The University of British Columbia (UBC), Vancouver, Canada, in 2024. From 2020 to 2023, he was a Mitacs Accelerate intern with Rogers Communications Canada Inc. He is currently a Senior Researcher with Huawei Noah's Ark Lab, Canada. From 2020 to 2023, he received the Graduate Support Initiative (GSI) Award from the Faculty of Applied Science at UBC. His research interests include network for AI (Net4AI), AI for network (AI4Net), machine learning, algorithm design, and optimization in wireless communication systems. He received the Best Paper Award at the *IEEE GLOBECOM* 2020.



Vincent W.S. Wong (S'94, M'00, SM'07, F'16) received the B.Sc. degree from the University of Manitoba, Canada, in 1994, the M.A.Sc. degree from the University of Waterloo, Canada, in 1996, and the Ph.D. degree from The University of British Columbia (UBC), Vancouver, Canada, in 2000. From 2000 to 2001, he worked as a systems engineer at PMC-Sierra Inc. (now Microchip Technology Inc.). He joined the Department of Electrical and Computer Engineering at UBC in 2002 and is currently a Professor. His research areas include protocol design, optimization, and resource management of communication networks, with applications to 5G/6G wireless networks, Internet of things, mobile edge computing, smart grid, and energy systems. Dr. Wong is the Editor-in-Chief of *IEEE Transactions on Wireless Communications*. He has served as an Area Editor of *IEEE Transactions on Communications* and *IEEE Open Journal of the Communications Society*, an Associate Editor of *IEEE Transactions on Mobile Computing* and *IEEE Transactions on Vehicular Technology*, and a Guest Editor of *IEEE Journal on Selected Areas in Communications*, *IEEE Internet of Things Journal*, and *IEEE Wireless Communications*. Dr. Wong was the General Co-Chair of *IEEE INFOCOM* 2024; Tutorial Co-Chair of *IEEE GLOBECOM* 2018; Technical Program Co-Chair of *IEEE VTC2020-Fall* and *IEEE SmartGridComm* 2014; and Symposium Co-Chair of *IEEE ICC'18*, *IEEE SmartGridComm* ('13, '17) and *IEEE GLOBECOM'13*. He received the 2022 Best Paper Award from *IEEE Transactions on Mobile Computing* and Best Paper Awards at the *IEEE ICC* 2022 and *IEEE GLOBECOM* 2020. He has served as the Chair of the IEEE Vancouver Joint Communications Chapter and IEEE Communications Society Emerging Technical Sub-Committee on Smart Grid Communications. He was an IEEE Communications Society Distinguished Lecturer from 2019 to 2020 and is an IEEE Vehicular Technology Society Distinguished Lecturer for the term of 2023–2025. Dr. Wong is a Fellow of the IEEE and the Engineering Institute of Canada (EIC).