Reinforcement based Communication Topology Construction for Decentralized Learning with Non-IID Data

Yi-Cheng Lin[§], Jian-Jhih Kuo[†], Wen-Tsuen Chen[§], and Jang-Ping Sheu[§]
[§]Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[†]Dept. of Computer Science & Information Engineering, National Chung Cheng University, Chiayi, Taiwan
E-mail: s108062646@m108.nthu.edu.tw, lajacky@cs.ccu.edu.tw, {wtchen,sheujp}@cs.nthu.edu.tw

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685241

***Abstract*—Federated Learning (FL) allows Internet-of-Things (IoT) devices to train a global model collaboratively and circumvent the security issue. However, the current FL framework has three main drawbacks, the *huge network overhead*, *single point of failure*, and *accuracy degradation in non-independent-and-identically-distributed (non-IID) data distribution*. We propose a novel Deep Reinforcement Learning (DRL) based Decentralized Learning (DL) framework, DeepSelect, to 1) reduce the network overhead of conventional FL, 2) construct a good communication topology adaptively to mitigate the effect of non-IID data, and 3) accelerate the DL training by balancing the effects of hitting time (HT) and data bias. Moreover, DeepSelect with a subtly-designed DRL agent is reusable with different levels of non-IID data distributions. To the best of our knowledge, this paper is the first one to indicate that proper neighbor selection for exchanging parameters (not raw data) can counterbalance the data bias's effect and improve the DL convergence with non-IID data. The experiment results show that DeepSelect can reduce 18%–51% training rounds than the other heuristics on FashionMNIST and CIFAR-10 with non-IID data distributions.**

***Index Terms*—Deep Reinforcement Learning, Decentralized Learning, Communication Topology, Federated Learning**

I. Introduction

The Internet-of-Things (IoT) is an important paradigm in 5G, which has a large number of devices and provides ubiquitous services in our daily life and generate tremendous data (e.g., voices, photos, positions). Many Machine Learning (ML) models are thus developed to analyze user data and make suitable strategies, while training the models centrally causes user data's security issue. To jointly benefit from the immense user data and circumvent the security issue, Federated Learning (FL) is proposed to distribute the training process into the user devices [1]. However, FL has the following drawbacks. 1) FL suffers from a single point of failure. If the parameter server crashes, the model aggregation will be terminated as well as the training process accordingly. 2) The network overhead between the user devices and the central parameter server is severe since extensive model parameters are uploaded to the central parameter server [2]. 3) The users' data distributions are usually non-independent-and-identically-distributed (non-IID) [3]. The skewed data distributions may seriously degrade the model accuracy and postpone the training convergence [4].

Decentralized Learning (DL) emerges to overcome the first two drawbacks of FL [5]. Specifically, each node performs *on-device* training and exchanges the model parameters with its neighboring nodes according to the given *communication topology* [6], [7]. Therefore, only few data exchanges occur among nodes, and thus the central parameter server is no longer needed. The network overhead is mitigated accordingly [2]. However, the third drawback is still challenging for DL. Researchers have spent a lot of effort handling the non-IID problem in the literature. Chiu *et al.* proposed an aggregate strategy named Federated Swapping to swap the training models among end-devices in each round [3]. Wang *et al.* presented the concept of choosing some nodes with the suitable data distribution to counterbalance the global model [4]. However, the above methods are designed only for FL. To the best of our knowledge, no method focuses on the non-IID issue for DL, thereby providing the motivation of this paper to make the first attempt to explore how to mitigate the non-IID issue for DL.

TABLE I
Effect of different HT on number of rounds

Target Accuracy	82%
Topology 1 (HT: 32)	112
Topology 2 (HT: 44)	169
Topology 3 (HT: 18)	99

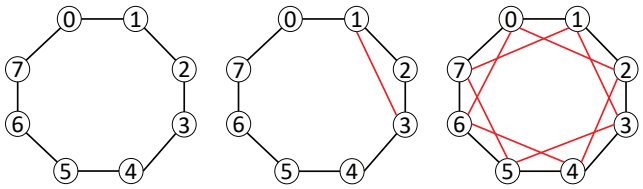
TABLE II
Effect of neighbor selection on number of rounds

Target Accuracy	80%
IID	41
non-IID CBA(2,0)	286
non-IID R	395

To better understand the rationale behind the DL, the experiments are conducted as follows. Firstly, we observe the relation between the convergence rate and the hitting time (HT). The experiment setting is detailed in Section IV. Figs. 1(a), 1(b), and 1(c) are a 2-regular topology (Topology 1), a 2-regular topology with one more edge (Topology 2), and a 4-regular topology (Topology 3) with HT of 32, 44, and 18, respectively. The effect of different HT on model accuracy (with independent and identically distributed (IID) distribution) is shown in Fig. 2(a). Topology 3 (red line) clearly outperforms the others after the 300-round training since it has the smallest HT among them. Specifically, Table I shows that Topology 3 requires 99 rounds to achieve the 82% accuracy. It is worth noting that the difference of one edge (i.e., Topologies 1 and 2) may have a great impact on HT and convergence rate. To reach the target accuracy, Topology 1 takes 112 rounds, which is only 67% of the rounds required by Topology 2 (i.e., 169 rounds). The above results conform to the observation that the convergence in DL is asymptotically proportional to the HT of the given communication topology [8]. Thus, an adequate method for communication topology construction should consider the topology structure carefully.

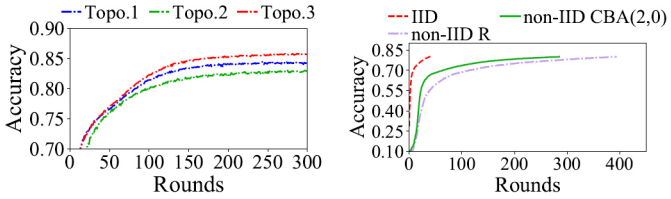
Secondly, we wonder whether neighbor selection on the communication topology may make a great impact on conver-

978-1-7281-8104-2/21/$31.00 ©2021 IEEE



(a) Topology 1 (b) Topology 2 (c) Topology 3

Fig. 1. An illustration of the cases of the communication topology.



(a) Effect of HT on accuracy (b) Effect of neighbor select. on acc.

Fig. 2. Motivating experiment results.

gence when the users' data distributions are non-IID. To verify the effect, we deploy 32 nodes on a 2-regular communication topology to train a classification model for the FashionMNIST dataset. In Fig. 2(b), the purple line (non-IID R) represents that the neighbors for each node are randomly assigned, while the green line (non-IID CBA(2,0)) adopts the subtly-selected neighbors for each node to counterbalance the effect of data bias (detailed in section III-A). Table II explicitly shows that non-IID R needs 395 rounds to achieve 80% accuracy, while non-IID CBA(2,0) only takes 286 rounds to reach the same accuracy, which reduces 28% of the number of training rounds and implies neighbor selection can mitigate the non-IID issue.

By the above experiments and observations, we find accelerating the convergence of DL in natural non-IID environments has the following new challenges. 1) Trade-off between topology regularity and neighbor selection. The more regular communication topology usually leads to a smaller HT and a better convergence rate. However, moderately adding a suitable set of edges into the communication topology to counterbalance the effect of data bias could facilitate the convergence rate, even if such a communication topology violates the regularity and increase the HT. 2) Trade-off between the network overhead and the degree of regular topology.[1] Increasing the degree of regular topology can decrease the HT and thus speed up the convergence. Nevertheless, the network overhead will increase accordingly since the number of neighbors of each node increases. Fig. 3 shows the relationship between the degrees of regularity and the network overhead, where the solid lines denote the accuracy of different regular topologies while the dotted lines indicate the cumulation of network overhead. Although the 16-regular topology achieves the highest accuracy after 500 rounds, it also causes the highest network overhead. Overall, this problem is very challenging since it has to jointly examine the HT (i.e., regularity) and the data bias to subtly balance their effects when tailoring an adequate communication topology to speed up the convergence.

To address the above challenges and solve the mentioned drawbacks of conventional FL, we propose the DL Framework with **Deep** Reinforcement Learning based Neighbor **Select**ion (DeepSelect) and present a new optimization problem named Efficient **Co**mmunication Topology **Co**nstruction in IoT for **D**ecentralized **L**earning (CoCoDL). Given 1) a set of training nodes, 2) a training model, and 3) the network information, CoCoDL asks for a set of links to construct a communication topology to speed up the convergence. To solve CoCoDL, DeepSelect contains a Deep Reinforcement Learning (DRL) based control agent to construct the communication topology on the fly to 1) reduce the network overhead of conventional FL, 2) construct a communication topology adaptively that can mitigate the influence of non-IID data, and 3) accelerate the DL training by balancing the effects of HT and data bias. The contributions of this paper are summarized as follows.

1) To the best of our knowledge, this paper is the first one to indicate that proper neighbor selection for exchanging parameters (not raw data) can counterbalance the data bias's effect and improve the DL convergence with non-IID data.
2) A DRL-based agent for DL along with DeepSelect is proposed to dynamically determine the set of suitable edges to overcome the issue caused by non-IID data distributions.
3) The performance of DeepSelect based on DRL is shown to make better actions than the other heuristic edge construction algorithms and outperform them in two well known datasets, FashionMNIST and CIFAR-10 with non-IID data.
4) DeepSelect with subtly-designed DRL agent is reusable with different levels of non-IID data distributions.

II. Preliminaries, Motivation, and Problem Model

A. Decentralized Learning (DL)

DL is an emerging learning framework where each participant node $i \in K$ locally solves the optimization problem with their own loss function $f_i : \mathcal{X} \to \mathbb{R}$ [2]. Let $f : \mathcal{X} \to \mathbb{R}$ denote the global loss function with the form $f(x) := \frac{1}{|K|}\sum_{i\in K} f_i(x)$. The nodes aim to collaboratively find the optimal parameters $x^* \in \mathbb{R}^d$ that minimizes the global loss function, i.e.,

$$f(x^*) = \min_{x\in\mathbb{R}^d} \frac{1}{|K|}\sum_{i\in K} f_i(x). \tag{1}$$

Algorithm 1 illustrates the detailed steps in DL. First, all nodes compute local stochastic gradient descent (SGD) [9] in parallel (line 2). Next, each node sends the updated model parameters (or gradient) to their neighboring nodes in the communication topology and receives the models from them (line 4). Then, each node follows the Lazy Metropolis-based update (see Definition 1) to merge the received models (line 5). The above operations will be repeated until the stopping criteria match.

Nedic et al. have shown that the convergence of the model is proportional to the HT [8]. For ease of reading, the calculation of the HT of a topology G is briefly described as follows. First, we have to derive the Lazy Metropolis-based Communication Matrix $M(G)$ (see Definition 1). Then, $M(G)$ can be used to

[1]For a regular topology, the degree of regular topology denotes the number of each node's neighbors.

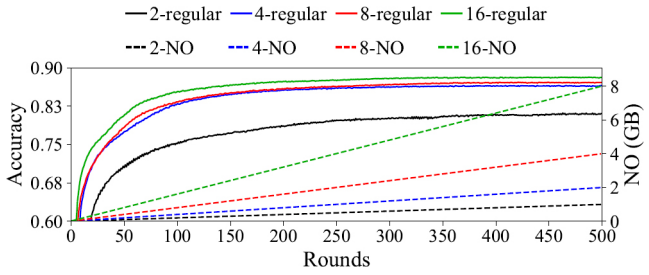


Fig. 3. Effect of different degree of regular topology on accuracy and network overhead (NO) with non-IID data distributions in FashionMNIST dataset.

Algorithm 1 Decentralized learning (DL)

Input: x_i^0, $i \in K$: Initial values of each node; $G = (K, E)$: communication topology; η: SGD step size;

1: **for** $t = 0$ to $T - 1$ **do** in parallel for all nodes $i \in K$
2: $y_i^t \leftarrow x_i^t - \eta \nabla f_i(x_i^t)$;
3: **for** each node $i \in K$ **do**
4: Send y_i^t and receive y_j^t to/from node j if $(i, j) \in E$;
5: $x_i^{t+1} \leftarrow \sum_{j \in K} m_{ij} y_j$;
6: **end for**
7: **end for**

compute the HT Matrix $H(G)$ (see Definition 2). Lastly, the hitting time of G is the largest entry in $H(G)$.

Definition 1 (Lazy Metropolis-based Communication Matrix [8])**.** *Given a set of training nodes K and communication topology $G = (K, E)$. The entries m_{ij} of the communication matrix $M(G) \in [0, 1]^{|K| \times |K|}$ are defined as*

$$m_{ij} = \begin{cases} 1 - \sum_{k \in K \setminus \{i\}} m_{ik}, & \text{if } i = j; \\ \frac{1}{2max\{deg(i), deg(j)\}}, & \text{else if } (i, j) \in E; \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Definition 2 (Hitting Time Matrix)**.** *Given a topology G with its communication matrix $M(G)$ from eq. (2), the entries h_{ij} of the hitting time matrix $H(G) \in \mathbb{R}^{|K| \times |K|}$ are define as*

$$h_{ij} = \begin{cases} 0, & \text{if } i = j; \\ 1 + \sum_{k \in K, k \neq j} m_{ik} \cdot h_{kj}, & \text{otherwise.} \end{cases} \quad (3)$$

where h_{ij} is the hitting time from node i to node j.

B. Influence of non-IID data and Neighbor Selection

Recall that Section I introduced an experiment to show that non-IID data distribution may slow down the convergence in DL. In Fig. 2(b), DL only needs to take 41 rounds to reach the target accuracy for the IID situation, while it takes more than 250 rounds to reach the same accuracy for the non-IID data distribution. On the other hand, Fig. 2(b) further shows that different neighbor selection will lead to different convergence results, even if the two communication topologies have the same nodes and the same HT. The topology with a random neighbor selection requires 395 rounds to reach the 80% accuracy, while the topology with a proper neighbor selection for counterbalancing the data bias only needs 286 rounds to reach the same accuracy.

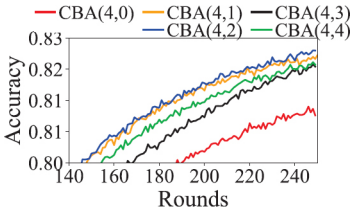


Fig. 4. Effect of HT and data bias

TABLE III
DATA BIAS AND HT WITH DIFFERENT c IN CBA(4,c)

CBA(4,C)	HT	Data Bias
$c = 0$	227.69	143.94×10^6
$c = 1$	236.06	143.80×10^6
$c = 2$	241.06	143.51×10^6
$c = 3$	248.39	143.09×10^6
$c = 4$	250.58	142.53×10^6

Therefore, we wonder which topology is more beneficial to speed up the convergence for non-IID data distributions, 1) a regular topology (i.e., lower HT) that may not consider the effect of data bias or 2) a non-regular topology (i.e., higher HT) that can counterbalance the effect of data bias. To explore the relation, we devise a heuristic named counterbalance bias algorithm (CBA) to add some edge into a given regular topology. The edge addition can help each node counterbalance the effect of its data bias on its local model parameters by aggregating the subtly-selected neighbors' local model parameters but at the price of higher HT. To calculate the data bias of each node, we treat each node and its neighbors as a set, then calculate the data bias of each set (eq. (5) in Definition 3) as the node's data bias. The heuristic starts from a 4-regular topology. Then, it iteratively selects the node with the largest data bias, say v, and connects v to the other node which is not in v's set but can decrease v's data bias the greatest. For ease of presentation, let CBA$(d, 0)$ denote the original d-regular topology, and let CBA$(d, 1)$ be CBA$(d, 0)$ plus one more edge selected by CBA, and so on.

Fig. 4 compares the convergence of the topologies with different additional edges selected by CBA, where 32 nodes are exploited to train the CNN model with non-IID data distributions on FashionMNIST dataset in 250 rounds. Table III shows the HT and data bias of each node in different topologies generated by CBA. We can find that the CBA$(4, 0)$ has the smallest HT and the largest data bias while achieving the worst accuracy. In contrast, the CBA$(4, 4)$ has the largest HT and the smallest data bias while reaches the third highest accuracy. It implies that a smaller data bias benefits the convergence. However, the CBA$(4, 2)$ with the second smallest HT and the third largest data bias can lead to the best accuracy. There may be a nonlinear relationship among HT, data bias, and convergence. Thus, in Section III-B, we propose a DRL-based method to explore the implicit relationship to select neighbors.

C. System Model and Problem Formulation of CoCoDL

We assume that the DL framework exploits a set of nodes K in an IoT system, where a platform is set up to monitor all participant nodes [10], to train the model collaboratively. Different from conventional FL, the platform does not collect and aggregate the model. Instead, the platform only determines the communication topology. Since the platform exchanges little information with nodes, it does not become the training bottleneck. Suppose each node transmits its data distribution (i.e., the amount of data for each class) to the platform in the beginning. The platform determines the communication topology based on the data distributions and the other information

collected later. Each node $i \in K$ has limited bandwidth B_i and computing C_i and thus can only construct E_i edges. CoCoDL is as follows:

$$\text{minimize} \quad \frac{1}{|K|} \sum_{i \in K} f_i(x^t) \tag{4a}$$

$$\text{subject to} \quad E_i \cdot \frac{Y}{B_i} + \frac{Y}{C_i} \leq \mathcal{T} \qquad \forall i \in K \tag{4b}$$

$$E_i \in [2, |K| - 1] \cap \mathbb{Z}, \qquad \forall i \in K \tag{4c}$$

$$\sum_{j \in K \setminus \{i\}} y_{ij}^t \leq E_i, \qquad \forall i \in K \tag{4d}$$

$$y_{ij}^t = y_{ji}^t, \quad y_{ij}^t \in \{0, 1\}, \quad \forall i, j \in K, i \neq j \tag{4e}$$

Eq. (4a) aims at the model parameter x^t to minimize the average loss of each training node. Eq. (4b) indicates that the degree of each node i (i.e., E_i) multiplied by transmission time (i.e., Y/B_i, where Y is model size and B_i is bandwidth limit of node i) and then add model training time (i.e., Y/C_i, where C_i is computing capability of node i) must be no more than a given time threshold $\mathcal{T}$. Eq. (4c) limits the degree of each node to range from two to the number of connectable nodes. Eq. (4d) implies that at most E_i edges can be selected for the each node $i \in K$, where y_{ij}^t denotes if edge (i, j) is selected for node i at time t. The last one shows the constraints of y_{ij}^t.

III. DL Framework with DRL-based Neighbor Selection (DeepSelect)

DeepSelect's one-episode training has two phases. The 1^{st} phase constructs a 2-regular topology as the communication topology based on the data bias of each node. In the 2^{nd} phase, the DRL agent adaptively adds or removes edges on the communication topology constructed by the 1^{st} phase during the DL. After training, we introduce the DRL agent's workflow.

A. Construct the 2-regular topology based on the data bias

Recall that neighbor selection based on users' data distributions for constructing the regular topology can mitigate the effect of data bias. Therefore, each node $i \in K$ uploads its data distribution S_i to the platform in the beginning.[2] Then, the platform examines the data bias of each node (see Definition 3) to select the suitable neighbors for each node to form a 2-regular topology.

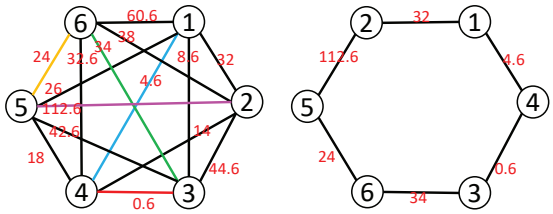
Definition 3 (Data Bias)**.** *Given a set of data with a distribution P and labels H. The data bias of P is defined as*

$$DataBias(P) = \sum_{l=1}^{H} (c_l - \overline{c})^2 \tag{5}$$

where $\overline{c} = \sum_{l=1}^{H} c_l / H$ denotes the average amount of data per label, and c_l denotes the amount of data in each label l.

Specifically, the process is presented step by step as follows.

[2]The nodes do not upload the raw data directly. Instead, they upload only their data distributions for data privacy concern. Moreover, the data distributions can be further disturbed by adding random noises (i.e., deferential privacy (DP) [11], [12]) for privacy concern. User privacy can be secured to some extent.



(a) Auxiliary complete graph (b) Constructed 2-regular topo.

Fig. 5. Example of Step 1

Step 1: Sets up an auxiliary complete graph $G_c = (K, L)$, where the weight of each edge $(i, j) \in L$ is set to the data bias (Definition 3) of the set of $\{i, j\}$. For example, given a set of nodes $K = \{1, 2, 3, 4, 5, 6\}$, the auxiliary complete graph is constructed as shown in Fig.5(a). Take edge $(1, 6)$ for example. Assume that there are three labels, and node 1's data distribution is $[3, 6, 5]$ while the node 6's data distribution is $[2, 9, 9]$. From eq. 3, the data bias of edge $(1, 6)$ (i.e., the set $\{1, 6\}$) is $(5 - 11.3)^2 + (15 - 11.3)^2 + (14 - 11.3)^2 = 60.6$. The weight of $(1, 6)$ is 60.6.

Step 2: Pick a starting point and add an edge with the smallest data bias into the edge set L. For example, the node 1 is picked as the starting point, and edge $(1, 4)$ with the data bias of 4.6 (i.e., blue line) is selected.

Step 3: Continue to select an unselected edge with the smallest data bias from the last covered node and add it to L. Repeat Step 3 until the 2-regular topology is constructed. For example, the edge $(4, 3)$ with the data bias of 0.6 is selected from node 4 (i.e., red line). The next picked edge is $(3, 6)$ with the data bias of 34 (i.e., green line). The last two edges selected are $(6, 5)$ with the data bias 24 and $(5, 2)$ with the data bias 112.6 (i.e., yellow and purple lines).

Step 4: Finally, to construct the 2-regular topology, the edge connecting the first covered and the last nodes are selected. For example, the constructed 2-regular topology follows the selection order $1, 4, 3, 6, 5, 2$ as shown in Fig. 5(b).

B. Deep Q-Network for Neighbor Selection

After the 1^{st} phase generates a d-regular topology, the 2^{nd} phase uses a DRL agent to select a valid action (i.e., add or remove an edge) in each round to change the communication topology for DL. DeepSelect trains a Deep Q-Network (DQN) as the DRL agent with the *Adam optimizer* based on the state, action, reward, and new state from the replay buffer. The state, action, and reward are described exhaustively as follows.

State: The state's design aims to make the DRL agent to overview 1) the current topology and 2) the feature extracted from node's model, which can reflect the node's data distribution to some extent. For the first goal, the neighbor information of each node $i \in K$ at time t is encoded into a *connection vector* $g_i^t \in \{0, 1\}^{|K|}$. For example, at time t, if node 1 has only two neighbors, nodes 0 and 2, then its connection vector is $g_1^t = \{1, 0, 1, ..., 0\}$. For the second goal, the model parameters of each node $i \in K$ at time t is reduced to a $|K|$-dimension vector by principle component analysis (PCA), i.e., $w_i^t \in \mathbb{R}^{|K|}$. Thus, the state of round t is represented by a vector $s_t = (g_0^t, ..., g_{|K|-1}^t, w_0^t, ..., w_{|K|-1}^t)$. The DRL

agent can consider the above information to construct the most suitable topology and reduce the training rounds.

Action: The DRL starts with a 2-regular topology as the based topology. It aims to approximate the optimal action-value function $Q^*(s_t, a)$. At the beginning of each round, the DRL agent selects a valid action with the highest value. Note that DRL is not allowed to remove the edge in the 2-regular topology, and thus the number of the edges that can be added or removed is $(\binom{|K|}{2} - 2 \cdot |K|)$. The actions can be classified into 3 categories, 1) construct an edge, 2) remove an edge, and 3) do nothing. Thus, the total size of the action space is $(\binom{|K|}{2} - 2 \cdot |K|) \cdot 2 + 1$. Remark that the action "construct an edge" is invalid if the edge exists in the communication topology now. Similarly, the action "remove an edge" is valid only if the edge exists in the communication topology.

Reward: Recall the HT and data bias affect the convergence significantly. The reward's design aims to make DRL agent select the action that can 1) accelerate the training, 2) lower the HT, and 3) reduce the data bias. Eq. (6) is the reward function for DeepSelect, which evaluates the result and returns a reward at the end of each round. The reward function contains the three terms, T_1, T_2, and T_3. The first term T_1 is related to the testing accuracy of the models. Factor τ is a constant, which ensures that T_1 grows exponentially. Factor ζ is the target accuracy, while ω_t denotes the average accuracy of *the two nodes with the selected action* at round t. If the action is doing nothing, the DRL agent will sample random 25% nodes for average to get the value of ω_t. The term T_1 is substantially important to the DRL agent since the reward with higher accuracy is far more than that with a lower one, which encourages the DRL agent to achieve the higher accuracy as soon as possible. The second term T_2 is about the HT, where h_t represents the HT of the communication topology at time t and h_{max} denotes the HT of the maximum degree of the regular topology that can be built under the network bandwidth limit. The term T_2 urges the DRL agent to do the action that can decrease the HT, since the convergence in DL is asymptotically proportional to the HT. The last term T_3 is data bias, where p_i denotes the data bias of each node $i \in K$ with their neighbors, and $d_i, i \in K$ is the number of data in each node. The term T_3 stimulates the DRL agent to reduce the data bias of each node. Without the assistance of the terms T_2 and T_3, DRL agent may not grasp the implicit relation between HT and data bias sooner.

$$r_t = \underbrace{(\tau^{\omega_t - \zeta} - 1)}_{T_1} - \underbrace{(\frac{h_t - h_{max}}{h_{max}})}_{T_2} - \underbrace{(\frac{\sqrt{\sum_{i=0}^{|K|-1} p_i}}{\sum_{i=0}^{|K|-1} d_i})}_{T_3} \quad (6)$$

The sum of the three terms is a negative number, which encourages the DRL agent to reach the target accuracy in fewer rounds. It is because the more rounds the DRL agent takes, the less total return it acquires.

C. The Workflow of DeepSelect

After training the DRL agent, the trained DRL agent will facilitate DeepSelect to select edges subtly. Specifically, the entire process of DL with DeepSelect is shown as follows. Suppose the nodes K train a model with size Y. Each node $i \in K$ has limited bandwidth B_i and limited computing C_i.

Step 1: All the nodes K initialize the model in the same way. Based on B_i, C_i, and model size Y, the platform first determines the maximum degree $E_i = \lfloor (\mathcal{T} - Y/C_i) \cdot (B_i/Y) \rfloor$ for each node $i \in K$ and the value of

$$d = \begin{cases} d_1 = \max_{i \in K} E_i, & \text{if } d_1 \mod 2 = 0; \\ d_2 = d_1 - 1, & \text{otherwise.} \end{cases} \quad (7)$$

Step 2: The platform selects the neighbors for each node $i \in K$ based on their initial data distributions and the value of d to construct a d-regular topology[3] as the initial communication topology.

Step 3: Each node trains the local model, sends and receives the model to/from its neighbors based on the communication topology, and aggregates the received models via Lazy Metropolis-based Communication Matrix (Definition 1).

Step 4: If the model has not converged yet, the DRL agent selects an action base on the state s_t on time t to modify the communication topology. Then, it repeats steps 3 and 4. Otherwise, the DL training stops.

IV. EVALUATION

A. Simulation Settings

Datasets and models: Our experiments are conducted with two well-known datasets, FashionMNIST and CIFAR-10.

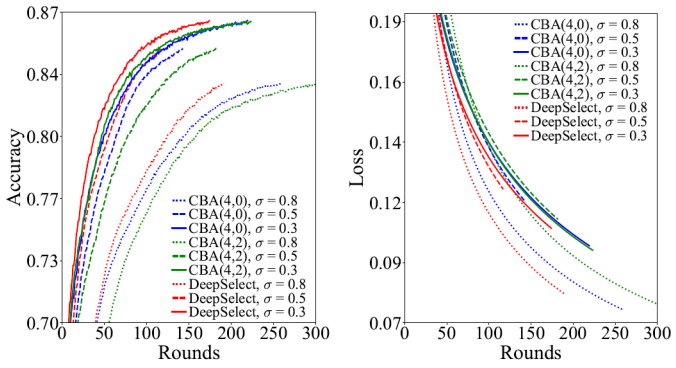
1) **FashionMNIST:** FashionMNIST consists of $60,000$ training examples and $10,000$ testing examples. We use a convolutional neural network (CNN) network with two 5×5 convolutional layers. The 1^{st} layer has 6 output channels, while the 2^{st} layer has 12. Both layers are followed by a 2×2 max pooling.
2) **CIFAR-10:** CIFAR-10 includes 50,000 images for training and 10,000 images for testing. We also use a CNN network with two 5×5 convolutional layers. The 1^{st} layer has 6 output channels, while the 2^{st} has 16. Both layers are followed by a 2×2 max pooling.

Non-IID data distributions: Following [4], we use data skewness σ to generate different levels of non-IID data distributions. For example, $\sigma = 0.8$ implies that 80% of training data belong to one label, while the remaining 20% training data belong to the other labels evenly.

Network scale and bandwidth limit: Following [13], we adopt 32 nodes (i.e., $K = 32$) to evaluate the DeepSelect's performance. We assume that nodes may use one of NB-IoT, WiFi, and LTE, and thus the each nodes' bandwidth B_i is set randomly from 10 to 80 Mbps. The nodes' training time ranges from 0.2 to 0.08 s. Thus, we set d to 4 according to Eq. (7) since the global model size Y is about 250 KB (i.e., $(1 - 0.2) \cdot 10 \cdot 1000/8/250$). Note $\mathcal{T}$ in Eq. (4b) is set to 1 s.

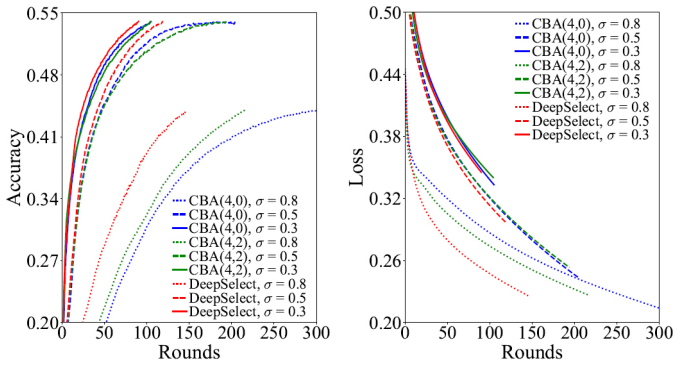
Performance metrics: Two metrics are adopted to evaluate performance: 1) loss function and 2) testing accuracy. Note that each result is averaged over 10 trials.

[3]Constructing a d-regular topology is similar to constructing a 2-regular topology, where $d \mod 2 = 0$. The only difference is at the 1^{st} phase's Steps 2 and 3 in Section III-A. It should select the edges connecting the last covered $d/2$ nodes to the $d/2$ nodes with the lower data bias.



(a) Accuracy on FashionMNIST (b) Loss on FashionMNIST

Fig. 6. Effect of different level of non-IID data on FashionMNIST dataset



(a) Accuracy on CIFAR-10 (b) Loss on CIFAR-10

Fig. 7. Effect of different level of non-IID data on CIFAR-10 dataset

Training the DRL Agent: We train the DRL agent on FashionMNIST and CIFAR-10 datasets with 32 nodes, respectively. The Deep Q-Network model in DRL agent consists of three fully connected layers. The input size is 2048, where a part of 32×32 is the communication topology, and the other part of 32×32 is the nodes' reduced model weights by sklearn.decomposition.PCA. The size of the output layer (i.e., actions) is $\binom{32}{2} - 2 \cdot 32) + 1$. The target accuracy ζ is set 80% in FashionMNIST and 43% in CIFAR-10. Each DRL agent is trained for 300 episodes, and each episode stops if the target accuracy reaches.

B. Effect of Different Levels non-IID Data Distributions

Recall that CBA(d, c) denotes the d-regular topology with c additional edges selected by CBA. We compare the performance of DeepSelect with the method CBA(d, c) proposed in section II-B. Since different levels of non-IID data distributions lead to different convergence rates, we set the different target accuracy for each different σ. For FashionMNIST with different non-IID settings $\sigma = 0.8, 0.5, 0.3$, we set our target accuracy to $83.1\%, 85\%, 86\%$, respectively, as shown in Fig. 6(a). For CIFAR-10 with different non-IID settings $\sigma = 0.8, 0.5, 0.3$, we set our target accuracy to 44%, 54%, 54%, respectively. We can find that in Fig. 7(a), CBA$(d, 0)$ performs worse than CBA$(d, 2)$ when $\sigma = 0.8$. On the other hand, in 6(a) $\sigma = 0.8$, CBA$(d, 0)$ outperforms CBA$(d, 2)$, which implies again that the accuracy, HT, and data bias have the implicit non-linear relationship. In contrast, DeepSelect can always strike the balance between data bias and HT to achieve the target accuracy with the minimum loss within the fewest rounds for the same setting as shown in Figs. 6(b) and 7(b). Table IV further shows that DeepSelect can reduce training rounds by 18%–51% while reaching the target accuracy.

TABLE IV
NUMBER OF ROUNDS TO REACH THE TARGET ACCURACY

Method	Non-IID level	FashionMNIST	CIFAR-10
CBA(4,0)	$\sigma = 0.8$	260 (1.35x)	301 (2.04x)
CBA(4,2)	$\sigma = 0.8$	301 (1.56x)	218 (1.48x)
DeepSelect	$\sigma = 0.8$	**192 (1x)**	**147 (1x)**
CBA(4,0)	$\sigma = 0.5$	144 (1.22x)	205 (1.70x)
CBA(4,2)	$\sigma = 0.5$	183 (1.55x)	195 (1.62x)
DeepSelect	$\sigma = 0.5$	**118 (1x)**	**120 (1x)**
CBA(4,0)	$\sigma = 0.3$	220 (1.25)	106 (1.16)
CBA(4,2)	$\sigma = 0.3$	224 (1.28)	105 (1.15)
DeepSelect	$\sigma = 0.3$	**175 (1x)**	**91 (1x)**

V. CONCLUSION

In this paper, we propose a new DRL-based DL framework, DeepSelect, to tackle two drawbacks of the FL framework, the *huge network overhead* and *single point of failure*. Besides, DeepSelect also mitigates the effect of the non-IID data distributions on the convergence, which is one of the most tricky issues in the *on-device* training. The paper indicates that a subtly-selected set of neighbors can help us speed up the model convergence rate substantially in non-IID environment. It should jointly balance the effect of topology regularity and neighbor selection while trade off the network overhead and degree of regular topology. Therefore, DeepSelect can reduce 18%–51% training rounds compared with the other heuristics for FashionMNIST and CIFAR-10 datasets.

REFERENCES

[1] J. Konečný *et al.*, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on PMPML*, 2016.

[2] X. Lian *et al.*, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *NIPS*, 2017.

[3] T.-C. Chiu *et al.*, "Semisupervised distributed learning with non-IID data for AIoT service platform," *IEEE Internet of Things J.*, vol. 7, no. 10, pp. 9266–9277, 2020.

[4] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on Non-IID data with reinforcement learning," in *IEEE INFOCOM*, 2020.

[5] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip Learning as a Decentralized Alternative to Federated Learning," in *IFIP DAIS*, 2019.

[6] C.-W. Ching *et al.*, "Efficient communication topology via partially differential privacy for decentralized learning," in *IEEE ICCCN*, 2021.

[7] J.-J. Kuo *et al.*, "Energy-efficient topology construction via power allocation for decentralized learning via smart devices with edge computing," *IEEE Trans. on Green Comm. and Networking (Early Access)*, 2021.

[8] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, pp. 953–976, 2018.

[9] Q. V. Le *et al.*, "On optimization methods for deep learning," in *ICML*, 2011.

[10] G. Yang *et al.*, "A health-IoT platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box," *IEEE Trans. Industr. Inform.*, vol. 10, no. 4, pp. 2180–2191, 2014.

[11] M. Abadi *et al.*, "Deep learning with differential privacy," *ACM SIGSAC CCS*, 2016.

[12] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, 2020.

[13] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *ICLR*, 2020.