Data-Centric Client Selection for Federated Learning over Distributed Edge Networks

Rituparna Saha, Sudip Misra, *Fellow, IEEE,* Aishwariya Chakraborty, *Student Member, IEEE,* Chandranath Chatterjee, Pallav Kumar Deb, *Student Member, IEEE,*

Abstract—This work presents an efficient data-centric client selection approach, named DICE, to enable federated learning (FL) over distributed edge networks. Prior research focused on assessing the computation and communication ability of the client devices for selection in FL. On-device data quality, in terms of data volume and heterogeneity, across these distributed devices is largely overlooked. The obvious outcome is the selection of an improper subset of clients with poor quality data, which inevitably results in an inefficient trained model. With an aim to address this problem, in this work, we design DICE which prioritizes the data quality of the client devices in the selection phase, in addition to their computation and communication abilities, to improve the accuracy of FL. Additionally, in DICE, we introduce the assistance of vicinal edge devices to account for the lack of computation or communication abilities in certain devices without violating the privacy-preserving guarantees of FL. Towards this aim, we propose a scheme to decide the optimal edge device, in terms of latency and workload, to be selected as the helper device. The experimental results show that DICE improves convergence speed for a given level of model accuracy. Further, the simulation results show that DICE reduces delay by at least 16%, energy consumption by at least 17%, and packet loss by at least 55% compared to the existing benchmarks while prioritizing the on-device data quality across clients.

Index Terms—Federated Learning, Client Selection, Offloading, Distributed Edge Networks, Fog Computing

1 Introduction

The advent of edge computing [1], [2] has led to the conceptualization of a novel distributed learning paradigm termed as *Federated learning (FL)* [3], which aims to preserve users' data privacy and confidentiality at the edge of the network. Unlike the traditional model training approach, FL decouples model training from direct access to the on-device training dataset. In essence, FL performs *local model training* at the edge devices (clients) and exchanges the updated model parameters to the centralized server to carry out the *global model aggregation* operation [4]. Thus, FL removes the need to exchange the on-device data with the server [3]. This decoupled model training approach addresses several challenges of the traditional centralized training approach[5], such as, (a) availability of the huge amount of real-world training data eases the data collection process, (b) local model training at distributed edge devices reduces the requirement of high computation power at the centralized server, and (c) exchange of model parameters, instead of sensitive data, with the server helps to maintain user privacy [6]. Thus, FL aids in bringing code to the data, instead of data to the code [7]. This privacy-preserving nature of FL also makes it useful for many data-sensitive use cases such as healthcare and banking. Despite its many advantages, the FL framework is posed with different challenges, such as (1) *communication overheads* due to iterative communication between different clients and the centralized server, (2) *heterogeneity of client devices* in terms of uneven distribution of data, varying computing capability, and network channel condition, and (3) *privacy issues at different levels*, e.g., inference attacks during the model training phase, leakage of private information from the trained model, manipulation of trained model output, and dependency on centralized global aggregator node.

Existing research works [8], [9] have successfully established that the selection of an optimum set of clients helps to reduce the communication latency and convergence time of the training operation. These works design the client selection problem of FL either based on the current network connection status and residual computation capability [8] or by predicting the next state of the network parameters for individual users [9]. These works do not consider the data quality across client nodes during the client selection process in FL. However, in machine learning, the quality of data at the device end is an important aspect. In an IoT scenario, data quality is defined in various ways. Some of the existing works [10] define data quality in terms of completeness, timeliness, and accuracy, whereas some others [11] point towards heterogeneity of data, missing or dirty data due to network constraints or security issues. The checking of data quality helps in increasing the model training efficiency and accuracy, and therefore, needs to be considered during the client selection process. Additionally, FL is usually performed with the help of numerous heterogeneous client devices, which differ in terms of network connectivity such as 3G, 4G, and 5G, and computation ability measured in terms of CPU capacity and usage and residual mem-

- *Rituparna Saha, Aishwariya Chakraborty, and Pallav Kumar Deb are with the Indian Institute of Technology Kharagpur, India. Email: (rituparnasaha@, aishwariyach@)iitkgp.ac.in, pallv.deb@gmail.com*
- *Sudip Misra is with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. Email: sudipm@cse.iitkgp.ac.in*
- *Chandranath Chatterjee is with the Department of Agriculture and Food Engineering, Indian Institute of Technology Kharagpur, India. Email: cchatterjee@agfe.iitkgp.ac.in*

ory. Moreover, such device heterogeneity poses different challenges, such as asynchronous local updates and client dropout [12], which ultimately affect the model efficiency. Furthermore, situations may arise in which certain client devices having exceptionally good quality data suffer from poor communication or computation capabilities. In such cases, it is a loss to not utilize good-quality data for FL.

An intuitive solution to the aforementioned problems is to utilize neighboring edge devices for communicating the model parameters to the server in case of communication impairment or for computation offloading in case of a lack of computation ability. However, both of these approaches require sharing of sensitive data or model parameters with neighboring devices which, in turn, results in various privacy threats. Primarily, these approaches face two types of privacy threats [13], [14] – (a) *inference during learning* which implies that a malicious client may have the intention to infer other clients' private information during the training process at the client end, resulting in the need to ensure *privacy on computation*; and (b) *inference over the outputs* which implies that a malicious client may try to infer information from the ML model parameters, thereby necessitating the *privacy on models*. Therefore, the main aim of this work is *to design a client selection process for FL training while considering the clients having good data quality but poor computation or communication ability, without breaking the privacy-preserving promises of FL.*

Towards this objective, in this work, we follow the fog-based FL framework, FogFL [15], in which a layer of fog devices is introduced to reduce the dependency on the centralized cloud server and works by random client selection approach without concern about the quality of data across the clients. Following this architecture, we devise a data-centric client selection scheme, DICE, to improve the efficiency of FL. Data quality checking is computationally intensive and incurs high computation overhead across resource-constrained devices in terms of computation and memory constraints. Further, the increased size of the dataset imposes high latency during data quality checking. Michailiodou et al. [10] design an algorithm that efficiently places the tasks across the edge and cloud devices to minimize the latency and maximize the level of data quality checking. Compared to this work, we consider data quality as a combination of data heterogeneity and data volume and aim to design a client selection algorithm for FL which prioritizes the clients having good data quality instead of having poor computation and communication ability. Thus, to incorporate computation or communication-deprived clients with good data quality into the FL operation, we introduce two alternative approaches – (a) For computation-constrained devices - we employ a computation offloading scheme to nearby edge devices. (b) For communication-impaired devices – we propose a scheme to transfer the local model parameters to the server through a neighbor device. Therefore, the main *contributions* of this work are as follows:

- We identify the problem of insufficiency of computation and communication abilities of client devices with high-quality local dataset and study its effects on the performance of FL over distributed edge networks.
- We propose a scheme to quantify the training dataset quality and the computation and communication capacity of the devices, and identify their capabilities by classifying them into groups based on these values.
- For computation/communication-impaired client devices with a good quality dataset, we propose the assistance of neighboring edge devices and design a scheme to select the most suited one based on the incurred delay.
- For executing FL with high accuracy and efficiency, we design another scheme that selects the optimal set of edge devices with high data quality, while increasing the convergence speed and ensuring low delay, energy consumption, and packet loss.
- For experimental evaluation, we design a lab-scale testbed to collect the real values of parameters such as energy consumption and processing time and use these values to conduct extensive simulations.

2 Related work

In this section, we briefly discuss existing literature on FL. The client management part of the FL framework is improvised in different ways, such as maintaining a client registry to hold information of participants, selecting clients based on predefined criteria [3], [16], [17], and grouping client devices based on certain characteristics [18]. These processes not only improve the model performance but also increase reliability, optimize resource usage and improve convergence speed [12]. However, these techniques face different challenges such as increased computation and communication costs and reduced data privacy. On the other hand, to manage client models, a few existing works, viz., [19], [20], proposed model compression techniques to reduce data size during transmission between the central server and client devices. Although these approaches improve communication efficiency and model quality, they are faced with issues such as loss of information and extra computation cost. Further, FL suffers from significant delays in synchronized parameter aggregation due to edge devices with weak computation capacity. In order to handle the straggler devices in the FL framework, Wang *et al.* [21] proposed an asynchronous collaboration scheme that again suffers from significant performance loss. In this regard, Xu *et al.* [17] proposed ELFISH, a resource-aware FL framework with a synchronous aggregation scheme to solve the computational straggler problem. Moreover, to solve the problem of different model upload times per device, Chen *et al.* [22] and Gu *et al.* [23] conducted an asynchronous model aggregation technique to increase the global model aggregation speed per round.

Another serious issue of the conventional FL framework is that the model training and aggregation are fully coordinated by a central server which results in issues such as single point of failure and network congestion. In this regard, Lalitha *et al.* [24] and Jiang *et al.* [25] proposed fully decentralized model aggregation algorithms. Again, Liu *et al.* [26] proposed a hierarchical server-edge-client

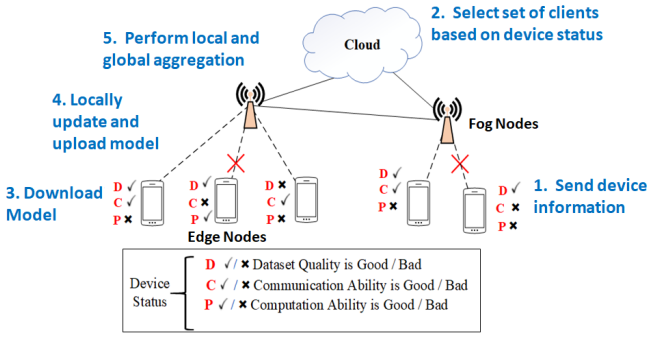


Fig. 1: System Model

architecture to improve both the computation and communication efficiency of the FL training process. However, the proposed framework raises questions regarding system reliability and security. In this context, Xu *et al.* [13] proposed an efficient privacy-preserving FL approach *HybridAlpha* using functional encryption. On the other hand, Nguyen *et al.* [27] and Pokhrel *et al.* [28] proposed the integration of FL with blockchain, whereas Kim *et al.* [29] proposed BlockFL where the local model updates are exchanged based on their corresponding rewards. Additionally, Bao *et al.* [30] and Weng *et al.* [31] proposed blockchain-based FL with incentive registry to record the performance of each client and provide incentives to increase client's participation. In this context, another renowned problem related to FL is the client selection problem. In the FL training process, a fraction of clients is get selected based on the allocated bandwidth and dynamic status of the clients. Therefore, there is a great impact of the client selection decision in each round on the global model convergence time and accuracy [32]. The existing researches design the client selection problem based on communication perspective [8], energy consumption factor [33]. However, current research mainly designs the client selection process of FL assuming pre-known local training time and the device performance which always limits the participation of poor-performance clients. Moreover, Ren *et al.* [34] proposed a DRL-based real-time computation offloading scheme for the IoT devices, where FL is used to train the multiple DRL agents deployed across multiple edge nodes. Prathiba *et al.* [35] proposed federated Q-learning-based FLOR algorithm for computation offloading and resource management and Ji *et al.* [36] proposed EAFL to alleviate the straggler problem of FL by partial computation offloading to the edge server.

Synthesis: Most of the aforementioned existing literature on FL considers the selection of clients either randomly or based on some predefined criteria, and do not consider the quality of data while selecting the set of clients for FL. Moreover, it is highly challenging to address privacy concerns and communication and computation cost in FL, while selecting a client having good data quality and poor computation or communication ability. Hence, to solve this pressing problem, we design a data-centric client selection approach for FL without hindering the privacy rules of FL.

3 System Model

We consider a fog-based FL framework [15] comprising a set of client devices (edge nodes) $E = \{E_1, E_2, E_3, .., E_N\}$, a set of fog nodes $\mathcal{F} = \{F_1, F_2, F_3, ..., F_M\}$, where $(M < N)$, and a centralized cloud server **S**, as shown in Fig. 1. In the fog-based FL framework, the cloud server is responsible only for client selection and storing the final global model, with no further intervention during FL training. Each round of FL starts with the selection of K edge nodes based on some predefined criteria, where $K < N$. We consider that each edge node E_i is associated with a vicinal fog node F_j and consists of a local dataset D_i in which $a \in D_i$ represents each training data sample. a has two components (X_a, Y_a), where X_a is the input data vector, and Y_a is the corresponding label of the output.

The sequence diagram, as shown in Fig. 2, manifests the working principle of the fog-based FL framework. In Fig. 2, the transition arrows numbered around (1) to (6) depict the different steps. (1) denotes the request for the participation of each client node to the centralized server to start the communication round of FL. (2) denotes the initiation of the client selection process at the centralized server, after receiving acknowledgment and the device information from the interested set of clients. (3) denotes the execution of the local update process at each client end after downloading the global model from the server. (4) represents the implementation of the local aggregation process at the fog nodes after receiving the local updates from the vicinal client nodes. (5) denotes the execution of the global aggregation process at an optimal fog node after the predefined number of rounds of client selection, local update, and local aggregation process. (6) denotes the upload of the global model after the completion of the whole FL framework.

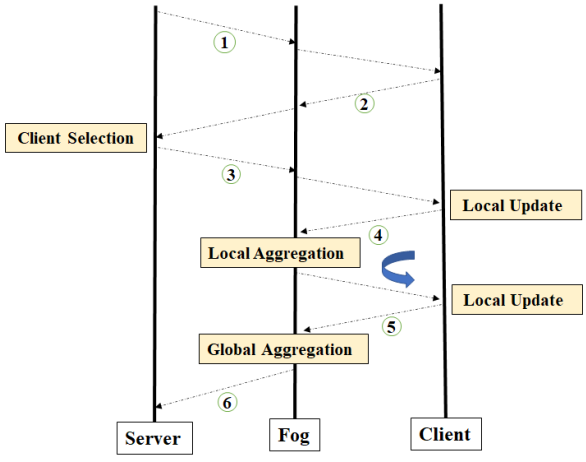


Fig. 2: Different steps of Fog-based FL Framework

During the *local update* phase, the client calculates its loss function for each data sample. We denote the local loss function of client E_i by $l_a(X_a, Y_a, w)$ or $l_a(w)$ for each data sample $a \in D_i$, where w denotes the vector of weights [3]. It is calculated as follows:

$$L(w_i) = \frac{1}{n_i} \sum_{a \in D_i} l_a(w), \tag{1}$$

where $n_i = |D_i|$ and θ denotes the local accuracy which satisfies the condition $0 \leq \theta \leq 1$. Each E_i updates its local parameters w_i using gradient descent algorithm as follows,

$$w_i(t) = w_i(t-1) - \eta \nabla L(w_i(t, B_i)), \tag{2}$$

where t denotes the number of iterations and B_i is the size of the mini-batch [3]. The learning rate η is always positive. Edge node E_i transmits the updated local parameters $w_i(t)$ to its vicinal fog node F_j to perform the *local aggregation* [15] as follows,

$$w_j(t) = \frac{\sum_{i=1}^{q_j} n_i w_i(t)}{n_j} \tag{3}$$

Here, q_j denotes the number of clients managed by fog node F_j. Considering binary variable $x_{i,j}$ to denote the association between client E_i with F_j, we have $q_j = \sum_{i=1}^{K} x_{i,j}$. Moreover, n_j denotes the total number of data samples of the clients under F_j, hence we have $n_j = \sum_{i=1}^{q_j} n_i$. We consider that, in the proposed framework, ε rounds of client selection and local update and aggregation are performed. Thereafter, the cloud server chooses the *global aggregator* node among the available fog nodes [15]. Towards this aim, the cloud server uses two parameters – latency and workload – to determine the optimal fog node for performing global aggregation. The following equation is used to calculate global aggregation parameters denoted by $\tilde{w}(t)$:

$$\tilde{w}(t) = \frac{\sum_{j=1}^{M} w_j(t)}{M} \tag{4}$$

where $w_j(t)$ represent the local aggregation parameters of each F_j. The primary assumptions made in this work are:
1. The ML task is a classification task and the FL algorithm uses SGD.
2. The FL framework follows the fog-based FL architecture.
3. The end devices are considered to be static edge nodes that are assigned to the same fog node throughout the FL training. Therefore, before the FL optimization process, the fog association problem is solved.
4. The centralized server is responsible only for client selection and storing the final global model, and does not intervene otherwise.

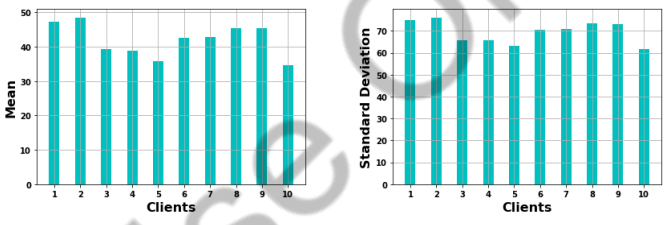
In the following subsections, we discuss the data, delay, and energy model for the fog-based FL model. Table 1 enlists the key notations used in this work.

3.1 Data Model

We define data quality of $D_i(t)$ across each client E_i using two important parameters – data heterogeneity (D^{het}) and data volume (D^{vol}). D^{het} across the clients is visible due to their varying device usage patterns and is defined by the data distribution which can be exhibited by the statistical metrics – mean and standard deviation. In this work, we consider data quality in terms of the distribution of the cleaned data across each client. Thus, we assume that the data is cleaned using techniques such as duplicate data removing, structured error fixing, outliers filtering, and missing data handling, prior to use in FL. Fig. 3a and 3b show the data heterogeneity across client data using the two metrics

TABLE 1: Key Notations

Notations	Description
M	Total number of fog nodes
N	Total number of client devices
f_i	CPU frequency at E_i
z_i	CPU cycles at E_i
B_i	Bandwidth at E_i
h_i	Channel gain
p_i	Transmission power at c_i
N_0	Background noise
$d_{i,j}$	Distance between E_i and f_j
s_i	Training model size at E_i
D_i	Dataset at E_i
$w_i(t)$	Local parameter at F_j
$w_j(t)$	Local aggregation parameter at F_j
$\tilde{w}(t)$	Global aggregation parameter at F_j



(a) Mean vs Clients (b) Standard Deviation vs Clients

Fig. 3: Non-IID Data Heterogeneity Across Clients

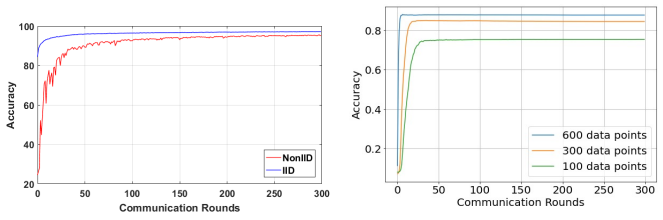
mean and standard deviation, respectively. We generate the results using the MNIST dataset and perform the IID and non-IID data partition following the base work [3] which presented the federated averaging algorithm (FedAvg). Hsu *et al.* [37] showed how the FL performance degrades with high data distributions across clients. In this paper, we consider the mean value to define the data distribution or data heterogeneity. Further, we consider D^{vol} as another parameter which is defined as the number of data samples across clients. Again, Fig. 4a and 4b depict the performance of the FL for data heterogeneity and varying volume of data, respectively. Fig. 4a manifests how the presence of data heterogeneity (non-IID data) degrades the performance of FL compared to the non-heterogeneity data (IID data). On the other hand, Fig. 4b infers how the increasing numbers of data points improve the performance of FL.

3.2 Delay Model

Each round of FL carries out local updation across each client E_i and communicates the intermediate updates periodically to the local server (fog node). We consider that all edge devices have the same bandwidth capacity B. To evaluate the communication delay at edge node E_i, we calculate the data rate r_i as follows:

$$r_i = Blog_2(1 + \frac{p_i h_i^2}{N_0}) \tag{5}$$

where p_i denotes the transmission power density, h_i denotes channel gain, and N_0 is the noise power [38]. Thus, the transmission delay δ_i^T and propagation delay $\delta_{i,j}^P$ are calculated as follows:



(a) For Varying Data Heterogeneity

(b) For Varying Data Volume

Fig. 4: FL Training Performance

$$\delta_i^T = \frac{s_i}{r_i} \quad \text{and} \quad \delta_{i,j}^P = \frac{d_{i,j}}{c} \tag{6}$$

where s_i denotes the size of transmitted model information, $d_{i,j}$ denotes distance of end device E_i to F_j and c denotes the speed of light. Thus, the total communication delay $\delta_{i,j}^{comm}$ of E_i is calculated as follows:

$$\delta_{i,j}^{comm} = \delta_i^T + \delta_{i,j}^P \tag{7}$$

On the other hand, the computation delay δ_i^{comp} at E_i is evaluated based on the computation capacity of the device using the following equation:

$$\delta_i^{comp} = \frac{z_i}{f_i} \tag{8}$$

where z_i is total number of CPU cycles needed to execute D_i and f_i denotes the CPU frequency of E_i.

3.3 Energy Consumption Model

The transmission energy consumption $\mathcal{E}_i^{comm}(t)$ of E_i for dataset D_i to communicate with the j^{th} local aggregator node at a time t is calculated as follows:

$$\mathcal{E}_i^{comm}(t) = p_i \delta_i^T, \tag{9}$$

where p_i denotes the transmission power of E_i. On the other hand, the computation energy consumption $\mathcal{E}_i^{comp}(t)$ for each local updation at E_i with the help of the following equation:

$$\mathcal{E}_i^{comp}(t) = \frac{\alpha_i}{2} z_i f_i^2 \tag{10}$$

where $\frac{\alpha_i}{2}$ is the effective capacitance coefficient of E_i, z_i is total number of CPU cycles needed to execute D_i and f_i denotes the CPU frequency of E_i [39].

Problem Scenario

The primary goal of FL is to learn a global model with help of multiple local models trained by multiple efficient candidates, selected as participants at each round such that they help to converge the training operation fast. However, the varying nature of wireless channels and model uploading deadlines restricts the selection of candidates. On the other hand, the two parameters, computation time and energy consumption of the clients contradict each other, i.e., for achieving a given level of accuracy within a lower (higher) computation time, a higher (lower) amount of energy is required. Although most of the existing client selection approaches [8], [9] for FL focus on the computation and communication ability of end devices, these works do not consider the quality of available on-device dataset which impacts the accuracy and efficiency of the trained model. Further, the incorporation of computation or communication-impaired devices into FL introduces several additional privacy threats. This necessitates the design of a data-centric client selection scheme to decide the optimal set of user devices to participate in each FL round while considering the device privacy requirements. Therefore, in this work, we propose DICE, an intelligent user selection scheme, as discussed in the following sections.

4 DICE: The Proposed Data-Centric Client Selection Approach For FL

In this section, we discuss the various steps of the proposed scheme, DICE. Initially, at the beginning of each FL round, like the traditional FL approach, the centralized server sends the request message to the clients. Subsequently, the interested clients send a request message to the server, which is comprised of the client's device status. Eventually, the centralized server randomly selects a client device, if the data quality score of the device is satisfactory then evaluates the computation and communication abilities of the client device and classifies them into different groups. Thereafter, the cloud server selects a helper device for the computation and communication-deprived clients. Afterward, the cloud server selects the optimal subset of devices. In the following, we discuss the three important steps of the proposed scheme, a) classification of the client device, b) selection of helper device and c) optimal client subset selection, in detail as follows.

4.1 Classification of Client Device

We define a few parameters, viz., data quality score, computation score, and communication score, to quantify the capabilities of the client devices. After selecting a client device based on its data quality score, the client device is classified based on its computation and communication score. We define the data quality score, computation score and communication score as in Definitions 1, 2, and 3, respectively. Towards this aim, we assume that the client devices are static in nature and we evaluate two types of parameters – *static* parameters which remain constant throughout a communication round, such as computation and communication delay, and computation and transmission energy consumption, and *dynamic* parameters which may vary during a communication round, such as residual memory and RSSI. Motivated by the works of Huang *et. al.* [9], we consider the prediction of the next state value of the dynamic parameters using the LSTM network model.

***Definition* 1.** *Data Quality Score $D_i^{score}(t)$ of a client device $E_i \in E$ at t^{th} round is defined in terms of the ratio of the data volume $D_i^{vol}(t)$ and the heterogeneity of data $D_i^{het}(t)$ generated by the device. Mathematically,*

$$D_i^{score}(t) = \alpha_1 \frac{D_i^{vol}(t)}{D_i^{het}(t)}, \tag{11}$$

where α_1 is the normalization constant such that $\alpha_1 = \frac{D^{het'}}{D^{vol'}}$.

Due to the degradation of FL performance with the increasing data distribution and the improved FL performance with the increasing data volume, we formulate the D_i^{Score} in such a way that $D_i^{vol}(t)$ is proportional to D_i^{Score} and $D^{het}(t)$ is inversely proportional to D_i^{Score}.

***Definition* 2.** *Computation score $\mathcal{P}_i^{score}(t)$ of a client E_i at round t is defined based on the ratio of predicted residual memory $\mathcal{M}_i^{pred}(t)$ to the pre-determined computation delay $\delta_i^{comp}(t)$ and energy consumption $\mathcal{E}_i^{comp}(t)$. Mathematically, we have:*

$$\mathcal{P}_i^{score}(t) = \frac{\frac{\mathcal{M}_i^{pred}(t)}{\mathcal{M}'}}{\alpha_2 \frac{\delta_i^{comp}(t)}{\delta^{comp'}} + (1-\alpha_2)\frac{\mathcal{E}_i^{comp}(t)}{\mathcal{E}^{comp'}}}, \quad (12)$$

where α_2 is a constant, which is used to control the importance of computation delay and computation energy. $\delta^{comp'}$, $\mathcal{E}^{comp'}$ and $\mathcal{M}'$ denote the maximum value of computation delay, computation energy and residual memory, respectively.

We formulate the $\mathcal{P}_i^{score}(t)$ in Eq. 12 in such a way as the computation ability of an edge device is increased with the high residual memory and also decreases with the high computation delay and computation energy,

***Definition* 3.** *Communication score $\mathcal{C}_i^{score}(t)$ of a client E_i at round t is defined based on the ratio of predicted RSSI value $\mathcal{R}_i^{pred}(t)$ to pre-determined communication delay $\delta_i^{comm}(t)$ and energy consumption $\mathcal{E}_i^{comm}(t)$. Mathematically, we have:*

$$\mathcal{C}_i^{score}(t) = \frac{(\mathcal{R}_i^{pred}(t)/\mathcal{R}')}{\alpha_3 \frac{\delta_i^{comm}(t)}{\delta^{comm'}} + (1-\alpha_3)\frac{\mathcal{E}_i^{comm}(t)}{\mathcal{E}^{comm'}}}, \quad (13)$$

where α_3 is a constant to control the importance of communication delay and communication energy. $\delta^{comm'}$, $\mathcal{E}^{comm'}$, and $\mathcal{R}'$ denote the maximum value of communication delay, communication energy and RSSI, respectively.

We formulate the $\mathcal{C}_i^{score}(t)$ in Eq. 13 in such a way as the communication ability of an edge device increases with the high RSSI value and decreases with the high communication delay and communication energy consumption,

The cloud server randomly selects a client $E_i \in E$ having $D_i^{score}(t) \leq D^{thresh}$ and determines it's group based on the calculated $\mathcal{C}_i^{score}(t)$ and $\mathcal{P}_i^{score}(t)$. The properties of the four groups are as follows:

- G_1 consists of clients with good computation and communication ability, i.e, $\mathcal{P}_i^{score}(t) \leq \mathcal{P}^{thresh}$ and $\mathcal{C}_i^{score}(t) \leq \mathcal{C}^{thresh}$.
- G_2 represents the group having clients with good computation but poor communication ability, i.e, $\mathcal{P}_i^{score}(t) \leq \mathcal{P}^{thresh}$ and $\mathcal{C}_i^{score}(t) \geq \mathcal{C}^{thresh}$.
- G_3 corresponds to the group having clients with good commutation but poor computation ability, i.e, $\mathcal{P}_i^{score}(t) \geq \mathcal{P}^{thresh}$ and $\mathcal{C}_i^{score}(t) \leq \mathcal{C}^{thresh}$.
- G_4 exhibits the clients with poor computation and communication ability, i.e, $\mathcal{P}_i^{score}(t) \geq \mathcal{P}^{thresh}$ and $\mathcal{C}_i^{score}(t) \geq \mathcal{C}^{thresh}$.

4.2 Selection of Helper Device

We define the criteria for the selection of a particular neighbor device $E_j \in E$ to help the edge device $E_i \in G_2$ or $E_i \in G_3$ or $E_i \in G_4$, $\forall i \neq j$, where E_i and E_j both are associated with the same fog device F_k. We formulate a parameter, termed as the priority $pri(E_j)$, for each neighbor device E_j, as defined in Definition 4, and the device with the minimum priority value is selected as the potential helper device for E_i.

***Definition* 4.** *Priority $pri(E_j)$ of neighbor edge device E_j is calculated based on the delay incurred due to communication and computation while helping the device E_i and the workload across the neighbor device E_j. Mathematically, we have*

$$pri(E_j) = \gamma * (\delta_{i,j}^{comm}(t) + \delta_{j,k}^{comm}(t)) + (1-\gamma) * \Omega_j(t), \quad (14)$$

where γ denotes the control variable, $\delta_{i,j}^{comm}(t)$ denotes communication latency between E_i and E_j, $\delta_{j,k}^{comm}(t)$ denotes communication latency between E_j and F_k and $\Omega_j(t)$ denotes the workload at E_j (in terms of data size in bit).

If $E_i \in G_2$, the value of γ is set to be greater than 0.5; if $E_i \in G_3$, it is set to be less than 0.5, whereas if $E_j \in G_4$, it is set to be equal to 0.5. Note that, in case the chosen helper device also does not possess sufficient computation or communication capacity, the next best helper device is selected.

4.3 Optimal Client Subset Selection

The next step is to select the optimal subset of client devices. It must be noted that the devices in G_1 train their local models using their local dataset and communicate their model parameters with their vicinal fog devices by themselves, while utilizing the Differential Privacy (DP) technique [40] to ensure privacy. However, to participate in the FL training process, the devices in G_2 require help from a neighboring device for communication offloading, and the devices G_3 and G_4 require help from a neighboring device for computation offloading. Thus, the devices in G_2 need to share their trained model parameters, whereas those in G_3 and G_4 are required to share their local data with the neighboring device. Therefore, to maintain privacy, for the devices in G_2, the same DP scheme is utilized to prevent model inversion attack, whereas the devices in G_3 and G_4 ensure data privacy using a lightweight homomorphic encryption [41] scheme. Thus, the devices in G_1 enjoy the maximum level of privacy followed by those in G_2, G_3, and G_4. However, the exact methods to ensure privacy is out of the scope of this work. In the following subsection, we discuss the selection of the most suitable vicinal edge device as a helper device for the devices in G_2, G_3, and G_4.

Now, to select the optimal subset of clients, we define the net gain or utility $\mathcal{U}_i$ based on the $D_i^{score}(t)$, $\mathcal{P}_i^{score}(t)$ and $\mathcal{C}_i^{score}(t)$, using the following equation. It must be noted that, during the evaluation of $\mathcal{U}_i$ for $E_i \in G_2$ or $E_i \in G_3$ or $E_i \in G_4$, $\mathcal{P}_j^{score}$ and $\mathcal{C}_j^{score}(t)$ of helper device E_j are considered.

$$\mathcal{U}_i = \frac{D_i^{score}(t)}{\beta \mathcal{C}_i^{score}(t) + (1-\beta)\mathcal{P}_i^{score}(t))} \quad (15)$$

where β is a control variable used to regulate the relative importance of communication and computation score. The main objective is to select an optimal set of K numbers of

clients, denoted as $X \in E$, where $|X| = K$, which maximizes the total gain or utility. We formulate this problem mathematically as follows,

$$\begin{aligned} \arg\max_{X \in E} \quad & \sum x_i(t)\mathcal{U}_i(t), \quad \forall E_i \in E, i \in N \\ \text{s.t.} \quad & C_1 : \sum_{i=1}^{N} x_i(t) \leq K, \quad x_i(t) \in \{0,1\} \\ & C_2 : x_i(t)\mathcal{M}_i^{pred}(t) \leq \mathcal{M}' \\ & C_3 : x_i(t)\mathcal{R}_i^{pred}(t) \leq \mathcal{R}' \\ & C_4 : x_i(t)\delta_i^{comm}(t) \leq \delta^{comm'} \\ & C_5 : x_i(t)\delta_i^{comp}(t) \leq \delta^{comp'} \\ & C_6 : x_i(t)\mathcal{E}_i^{comm}(t) \leq \mathcal{E}^{comm'} \\ & C_7 : x_i(t)\mathcal{E}_i^{comp}(t) \leq \mathcal{E}^{comp'}, \end{aligned} \tag{16}$$

where $x_i(t)$ is the decision variable which denotes whether E_i is selected or not at round t. C_1 indicates the maximum number of selected clients for the FL operation. C_2 and C_3 denote predicted residual memory and RSSI value constraints, respectively. C_4 and C_5 denote the maximum permissible communication and computation delay, respectively, to execute the FL algorithm at round t. Further, C_6 and C_7 denote the communication and computation energy constraints, respectively. We solve this optimization problem using a greedy approach, as shown in Algorithm 1.

Algorithm 1 `DICE Algorithm`

INPUT: E
OUTPUT: X
PROCEDURE:

Initialize an empty set X
Set $temp = 0$
for $I = 1 : T$ **do**
  Set $j = 1$ and $sum = 0$
  Initialize an empty set E'
  while $j < K$ **do**
    Randomly select a client $E_i \in E$
    Calculate $D_i^{score}(t), \forall E_i \in E$
    if $D_i^{score}(t) \leq D^{thresh}$ **then;**
      Calculate $\mathcal{P}_i^{score}(t)$ and $\mathcal{C}_i^{score}(t)$ for E_i
      Identify group of E_i
      if E_i belongs to G_2 or G_3 or G_4 **then**
        Select helper device $E_j \in E$
        Calculate $\mathcal{P}_j^{score}(t)$ and $\mathcal{C}_j^{score}(t)$ for E_j
      end if
      Calculate $\mathcal{U}_i$ using Eq. 15
      Set $sum \leftarrow sum + \mathcal{U}_i$
      Set $E' \leftarrow E' \cup \{E_i\}$
    end if
    Set $j \leftarrow j + 1$
  end while
  if $temp \leq sum$ **then**
    Set $temp \leftarrow sum$
    Set $X \leftarrow E'$
  end if
end for
Return X

Analysis of The Proposed Scheme

We select the optimal set of clients $X \in E$ which have the highest utility as shown in Equation 16. It may be possible that the set $X(t)$ of selected end nodes at t time instant may vary from the set $X(t+1)$ of selected nodes at $t+1$ time instant, i.e. $X(t) \neq X(t+1)$. Fig. 5 shows the utility value across the selected clients at an iteration t for different values of control variable β such as 0.2, 0.5, and 0.8. Fig. 5 depicts that, with an increasing value of β, the utility value varies less.

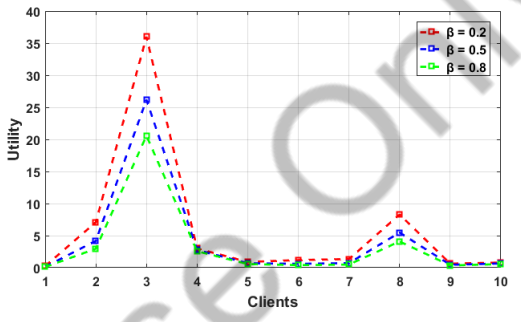


Fig. 5: Utility Values for Different Value of β

We also determine the complexity of the **Algorithm 1** in this section. The outer for loop to find the optimum set $X \in E$ is executed T times, hence has complexity $\mathcal{O}(T)$. The while loop in Step 6 to randomly select K number of clients, has complexity $\mathcal{O}(K)$. Step 7-11 to calculate and verify the D_i^{score} then calculate P_i^{score} and C_i^{score} value, and identify the group of E_i has complexity $\mathcal{O}(1)$. Then Step 13 which selects a helper device E_j with the $min(pri(E_j))$ for E_i belongs to G_2 or G_3 or G_4, incurs complexity of $\mathcal{O}(log(N_1))$ for N_1 number of neighbour nodes across E_i, where $N_1 < N$. Next, the Step 16-20 incurs complexity $\mathcal{O}(1)$. Thus, the total complexity of the algorithm is $\mathcal{O}(T \times Klog(N_1))$.

5 Performance Evaluation

For the performance evaluation of DICE, we perform extensive experiments and compare the results with other benchmark solutions in terms of different performance metrics. Towards this aim, we design a testbed and collect the real values of several parameters. Thereafter, we conducted numerical simulations[1] using these values and comparing the results. The details of the performance analysis are presented below.

5.1 Experimental Setup

For all the five schemes, we consider that the edge nodes are equipped with the MNIST dataset. These devices train and test the multi-layer perceptron (MLP) model with 2 hidden layers (2NN), each with 400 neurons using the ReLU activation function. We consider a non-IID partition of the dataset across each client following the works of McMahan *et al.* [3]. Fig. 3a and 3b show the data heterogeneity across

1. The simulation code can be found at https://github.com/Rituparna30/DICE.

TABLE 2: Simulation Parameters

Parameters	Value
Simulation iteration	100
Number of Edge devices	100, 200, 300
Number of Fog devices	6, 12, 18
Simulation area	$5km \times 5km$
Model size	7.7 Mb
Noise	-100 dB
Communication range of edge nodes	350 m
CPU frequency of edge nodes	1.5 GHz
Transmission power of edge nodes	2.2 W
Average execution time of edge nodes at each round to locally update	15.57 sec
Average energy consumption of edge nodes at each round to locally update	7 mAh
Bandwidth between fog and edge nodes	20 MHz

the clients using the mean and standard deviation of non-IID data, respectively. Thereafter, we design the simulation environment in Matlab R2018a while considering a large number of working nodes and varying client fractions, fog nodes, and a number of local aggregations. During the simulation, we also use real average energy consumption and execution time values of the edge nodes (Raspberry Pi) for on-device training. We design the simulation environment with a $5km \times 5km$ coverage area with a varying number of edge nodes 100, 200, and 300. The ratio of the fog nodes to the edge nodes is maintained at 0.06 at all times. The detailed simulation parameters are listed in Table 2.

5.2 Benchmark Schemes

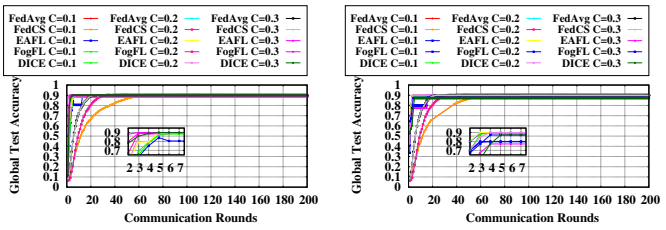
We consider the three variants of FL framework as benchmark schemes – a) traditional cloud-based FL framework (FedAvg) [3] which follows random selection of clients, b) random client selection-based computation offloading scheme for edge-assisted FL framework (EAFL) [36], c) client selection approach for cloud-based FL framework (FedCS) [8] which follows client selection through maintaining certain deadline for model update and upload, and d) traditional Fog-based FL framework (FogFL) [15] which follows random client selection approach. In the first three schemes, the whole training process is coordinated by a centralized server, whereas in the rest schemes, the centralized server is only responsible for the client selection part. However, the aforementioned approaches neither consider the data quality, nor the possibility of incorporating computation or communication-deprived client devices in FL. Table 3 depicts the comparison of different features with the benchmarks.

5.3 Performance Metrics

We evaluate the performance of the DICE based on the following metrics. We also evaluate the scalability of DICE by varying the number of edge and fog nodes.

Learning Parameters: We evaluate the global test accuracy of the FL training model at each round. In this work, we aim to improve the accuracy of the trained model by introducing clients having high-quality dataset in FL.

Network Parameters: We evaluate the network parameters such as average delay (total update and upload time),



(a) For Local Aggregation = 10 (b) For Local Aggregation = 20

Fig. 6: Global Test Accuracy

energy consumption, and packet loss for all participating clients during each FL round, as discussed in Section 3.

Reliability: We define system reliability as the fraction of the number of clients who complete the training operation within the specified deadline at each FL round. This is an essential system quality parameter that captures the applicability of the proposed scheme. If the majority of the selected clients are unable to deliver within a deadline, the FL procedure suffers.

Data Quality of Selected Client Devices: We evaluate the data quality of selected client devices at each communication round in terms of data score using Equation 11. The selection of high-data-quality clients ensures the efficiency of the system.

5.4 Results and Discussions

In this section, we discuss the results and observations of the simulations.

5.4.1 *Impact on Learning Parameters*

We analyze the global test accuracy of the FL frameworks for different client fractions $C = 0.1, 0.2$, and 0.3 with the varying number of local aggregations such as 10 and 20, as shown in Figs. 6a and 6b, respectively. Both results manifest that the fog-based FL frameworks (FogFL and DICE) converge faster than the cloud-based FL frameworks (FedAvg and FedCS) due to the integration of intermediate fog nodes. The fog nodes reduce the total number of global aggregations by performing local aggregations. Additionally, the incorporation of higher-quality data also contributes to the faster convergence time. Moreover, the results exhibit that, in all types of environments, the test accuracy result significantly improves with the increase in client fractions ($c \geq 0.1$). This can be attributed to the increase in the variety of the training data which reduces overfitting.

5.4.2 *Impact on Network parameters*

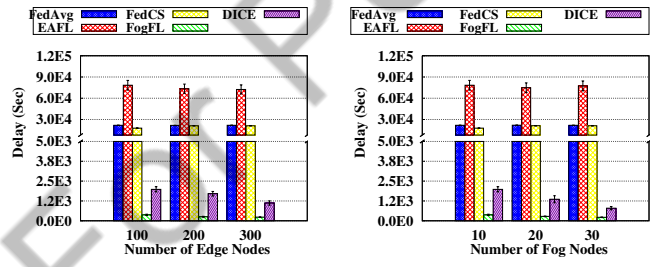
As mentioned earlier, we consider three parameters – delay, energy consumption, and packet loss for network performance evaluation.

Delay: We observe the average delay in each FL round for varying numbers of edge and fog nodes for the five schemes, as shown in Fig. 7a and 7b, respectively. For all five schemes, the average delay is the combination of local computation delay across edge nodes and updated model transmission delay from the edge nodes to the centralized

TABLE 3: Comparison of Different Features With the Benchmarks

FL Framework	Architecture	Computation Offloading	Optimize Delay	Optimize Energy Consumption
FedAvg [3]	Cloud-based	No	No	No
EAFL [36]	Cloud-based	Yes	Yes	No
FedCS [8]	Cloud-based	No	Yes	No
FogFL [15]	Fog-assisted	No	Yes	Yes
DICE	Fog-assisted	Yes	Yes	Yes

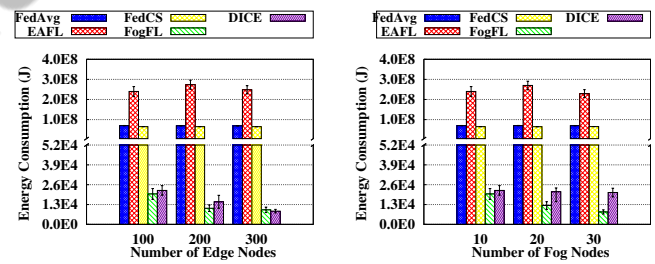
server or fog node. Therefore, the figures manifest that the fog-based FL frameworks experience a lower average delay than the cloud-based FL frameworks due to higher transmission delay in the case of the latter. Moreover, among the three cloud-based FL frameworks EAFL comprises a high average delay due to the addition of another type of transmission delay in order to perform the computation offloading by the transmission of partial data from each edge node to the central server. Besides, among the fog-based FL systems, DICE exhibits a high delay compared to FogFL. This can be attributed to the integration of a data-centric client selection strategy with the fog-based FL framework which adopts a policy of offloading computation or communication to a neighbor device for the selected resource-constrained client devices. Further, in DICE, for the communication-constrained client nodes, the average delay is the combination of the computation delay across client nodes and two types of transmission delay – one from the client node to the helper node and another from the helper node to the fog node. Again, for the computation-deprived client nodes, the average delay is the combination of the transmission delay from the client node to the helper node and the computation delay across the client node. Besides, the computation-deprived client nodes also consider the transmission delay from the helper node to the fog node, in case of a better communication score across the helper node than the client node. From the figures, we also observe that the delay of the proposed fog-based system decreases with the increase in the total number of edge nodes and fog nodes. Overall, DICE reduces delay by 91%, 97%, and 91% compared to the FedAvg, EAFL, and FedCS, respectively, and increases the delay by 18% to FogFL.



(a) Delay vs #Edge Nodes (b) Delay vs #Fog Nodes

Fig. 7: Variation of Delay with Number of Edge and Fog Nodes

Energy Consumption: We observe the average energy consumption of the edge nodes at each round of FL for both varying numbers of edge and fog nodes from Fig. 8a and 8b in comparison to the benchmarks. The average energy consumption across the five schemes comprises the computation energy across the client nodes and transmission energy due to model transmission from the client to the centralized server or fog node. The figures manifest that the DICE-based FL outperforms the cloud-based frameworks and reduces energy consumption by 97%, 98%, and 97% compared to the FedAvg, EAFL, and FedCS, respectively. Whereas, DICE-based FL increases energy consumption by 11% than FogFL. Moreover, from the figures 8a and 8b, we also manifest that the energy consumption reduces further with the increasing number of edge and fog nodes due to the data-centric client selection strategy and also, due to faster convergence of the FL training operation. Further, we also separately exhibit the average energy consumption value of edge nodes across different groups in Fig. 9, where the first bar denotes the energy consumption of G_1, the second bar denotes the energy consumption of G_2 and the third bar for G_3 and G_4. The Fig. 9 manifests that G_2, G_3 and G_4 score high energy consumption than G_1. This is because DICE-based FL considers computation and communication energy consumption across the helper nodes to support the computation and communication-deprived clients, respectively, unlike the other schemes which support the G_1 type of edge nodes only.



(a) Energy Consumption vs #Edge Nodes (b) Energy Consumption vs #Fog Nodes

Fig. 8: Variation of Energy Consumption With Number of Edge and Fog Nodes

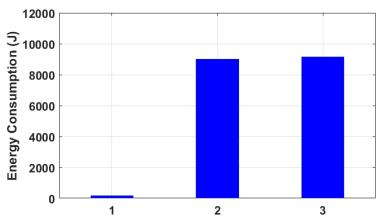
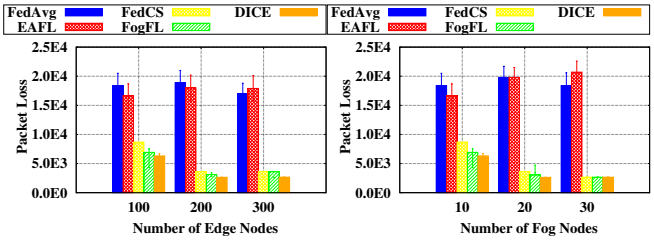


Fig. 9: Average Energy Consumption of Edge Nodes Across Different Groups

Packet Loss: We observe the average packet loss of the five schemes for varying numbers of edge and fog nodes from Figs. 10a and 10b, respectively. Packet loss is primarily

caused due to network congestion. Both the Figs. 10a and 10b manifest that DICE substantially reduces packet loss by 65%, 62%, and 27% compared to the benchmarks FedAvg, EAFL, and FedCS, respectively, and 10% for FogFL, which implies a reduction in the network congestion. This can be attributed to the fact that the RSSI at the client devices is considered in the case of calculation of the communication score of the devices, which in turn results in the selection of devices with higher RSSI, thereby decreasing the packet loss in the case of DICE.



(a) Packet Loss vs #Edge Nodes

(b) Packet Loss vs #Fog Nodes

Fig. 10: Variation of Packet Loss with number of Edge and Fog Nodes

5.4.3 Impact on Reliability

We analyze the reliability of the proposed system as the ratio of the number of edge nodes or client devices attempting a training round successfully without dropout within the deadline time to the total number of participating clients in that round. The simulation results, presented in Table **??**, show that DICE-based FL outperforms the benchmarks and shows a high number of successfully attempted clients within different decreasing deadline times, thereby proving the reliability of the system. This is expected due to the implementation of an efficient data-centric client selection approach, unlike the other benchmarks. Due to the prioritization of client devices based on data quality, the convergence of the algorithm occurs faster using DICE, and the use of supporting neighbor devices aids in the successful completion of training.

TABLE 4: Successfully Attempted Clients Ratio at single Round During Training

Time (Sec)	FedAvg	EAFL	FedCS	FogFL	DICE
700	0	0	0	1	1
2.15E+02	0	0	0.75	0.75	1
2.15E+04	0.4	0.6	1	1	1

5.4.4 Impact on Data Quality of Selected Clients

We also analyze the data quality of the selected clients using DICE and the benchmarks, based on their average data score value for varying numbers of edge devices, as shown in Fig. 11. Fig. 11 depicts that the data score value of selected edge devices using DICE is higher than other benchmarks for all cases of edge nodes. This is due to prioritizing the high data score client devices during selection for FL using DICE, unlike the benchmarks, in which the communication and computation abilities of the clients are prioritized over the data quality. This in turn leads to the selection of edge devices having good data quality but poor computation or communication abilities, which are otherwise discarded in the case of the benchmark schemes.

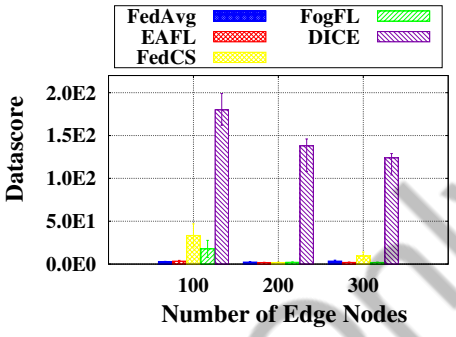


Fig. 11: Datascore vs #Edge Nodes

6 Conclusion

In this work, we proposed a data-centric client selection approach, DICE, for the fog-based FL framework. In DICE, we consider the quality of the on-device data of each client is prioritized over the client device status, viz., computation and communication ability, while selecting the optimal set of clients for FL. Additionally, a privacy-aware offloading scheme is proposed in this work to select the best neighbor device for offloading, thereby enabling computation or communication-constrained client devices having good data quality to participate in FL. We designed extensive simulations and evaluated the performance of DICE in comparison with four existing benchmarks. Simulation results showed that the proposed scheme DICE improves the accuracy of FL while ensuring low energy consumption and delay.

This work can be extended in the future while considering the active participation of the client devices in the client selection process. This work can also be extended by studying the statistical nature of the effect of different data quality parameters on the FL algorithm.

References

[1] A. Aral and I. Brandić, "Learning spatiotemporal failure dependencies for resilient edge computing services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1578–1590, 2020.

[2] S. Misra, A. Mukherjee, A. Roy, N. Saurabh, Y. Rahulamathavan, and M. Rajarajan, "Blockchain at the edge: Performance of resource-constrained iot networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 174–183, 2021.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[4] V. Balasubramanian, M. Aloqaily, M. Reisslein, and A. Scaglione, "Intelligent Resource Management at the Edge for Ubiquitous IoT: An SDN-Based Federated Learning Approach," *IEEE Network*, vol. 35, no. 5, pp. 114–121, 2021.

[5] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[6] S. Otoum, N. Guizani, and H. Mouftah, "On the Feasibility of Split Learning, Transfer Learning and Federated Learning for Preserving Security in ITS Systems," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–9, 2022.

[7] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards Federated Learning at Scale: System Design," *arXiv:1902.01046*, 2019.

[8] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proceedings of IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[9] H. Huang, K. Lin, S. Guo, P. Zhou, and Z. Zheng, "Prophet: Proactive candidate-selection for federated learning by predicting the qualities of training and reporting phases," *arXiv preprint arXiv:2002.00577*, 2020.

[10] A.-V. Michailidou, A. Gounaris, M. Symeonides, and D. Trihinas, "Equality: Quality-aware intensive analytics on the edge," *Information Systems*, vol. 105, p. 101953, 2022.

[11] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. Guttula, S. Mujumdar, S. Afzal, R. Sharma Mittal, and V. Munigala, "Overview and importance of data quality for machine learning tasks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3561–3562.

[12] S. K. Lo, Q. Lu, L. Zhu, H.-y. Paik, X. Xu, and C. Wang, "Architectural patterns for the design of federated learning systems," *arXiv preprint arXiv:2101.02373*, 2021.

[13] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019, pp. 13–23.

[14] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, 2021.

[15] R. Saha, S. Misra, and P. K. Deb, "Fogfl: Fog assisted federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, 2020.

[16] W. Luping, W. Wei, and L. Bo, "Cmfl: Mitigating communication overhead for federated learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 954–964.

[17] Z. Xu, Z. Yang, J. Xiong, J. Yang, and X. Chen, "Elfish: Resource-aware federated learning on heterogeneous edge devices," *arXiv preprint arXiv:1912.01684*, 2019.

[18] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, 2020, pp. 125–136.

[19] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," *arXiv preprint arXiv:2007.01154*, 2020.

[20] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[21] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[22] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data." IEEE, 2020, pp. 15–24.

[23] B. Gu, A. Xu, Z. Huo, C. Deng, and H. Huang, "Privacy-preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning," *IEEE transactions on neural networks and learning systems*, 2021.

[24] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Third workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[25] J. Jiang and L. Hu, "Decentralised federated learning with adaptive partial gradient aggregation," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 230–236, 2020.

[26] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[27] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.

[28] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4734–4746, 2020.

[29] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2019.

[30] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "Flchain: A blockchain for auditable federated learning with trust and incentive," in *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*. IEEE, 2019, pp. 151–159.

[31] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[32] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1552–1564, 2020.

[33] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient radio resource allocation for federated edge learning," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2020, pp. 1–6.

[34] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things," *IEEE Access*, vol. 7, pp. 69 194–69 201, 2019.

[35] S. B. Prathiba, G. Raja, S. Anbalagan, K. Dev, S. Gurumoorthy, and A. P. Sankaran, "Federated learning empowered computation offloading and resource management in 6G-V2X," *IEEE Transactions on Network Science and Engineering*, 2021.

[36] Z. Ji, L. Chen, N. Zhao, Y. Chen, G. Wei, and F. R. Yu, "Computation offloading for edge-assisted federated learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9330–9344, 2021.

[37] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[38] S. Misra and N. Saha, "Detour: Dynamic task offloading in software-defined fog for iot applications," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, 2019.

[39] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated Learning over Wireless Networks: Optimization Model Design and Analysis," in *Proceedings of IEEE INFOCOM*, 2019, pp. 1387–1395.

[40] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[41] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, Jul. 2020, pp. 493–506. [Online]. Available: https://www.usenix.org/conference/atc20/presentation/zhang-chengliang

Rituparna Saha is presently pursuing her Ph.D. degree from the Advanced Technology Department Centre, IIT Kharagpur, India and also working as a project fellow in Impacting Research Innovation and Technology project supported by the Ministry of Human Resource and Development. Her research interests include edge intelligence and machine learning. She received her M. Tech. and MCA degrees from the West Bengal University of Technology in 2016 and 2014, respectively. She studied B.SC. Hons. in computer science from Calcutta University in 2011.

Sudip Misra (SM'11) is a Professor at IIT Kharagpur. He received his Ph.D. degree from Carleton University, Ottawa, Canada. Prof. Misra is the author of over 350 scholarly research papers. He has won several national and international awards including the IEEE ComSoc Asia Pacific Young Researcher Award during IEEE GLOBECOM 2012. He was also the recipient of several academic awards and fellowships such as the INSA NASI Fellow Award (National Academy of Sciences, India), the Young Scientist Award (National Academy of Sciences, India), Young Systems Scientist Award (Systems Society of India), and Young Engineers Award (Institution of Engineers, India). He has also been serving as the Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING, the IEEE SYSTEMS JOURNAL, and the INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS (Wiley). He is a Guest Editor of the IEEE NETWORK Magazine. He is also an Editor/Editorial Board Member/Editorial Review Board Member of the IET NETWORKS and the IET WIRELESS SENSOR SYSTEMS. Dr. Misra has 11 books published by Springer, Wiley, and World Scientific. He was invited to chair several international conference/workshop programs and sessions. Dr. Misra was also invited to deliver keynote/invited lectures in over 30 international conferences in USA, Canada, Europe, Asia, and Africa. For more details, please visit http://cse.iitkgp.ac.in/∼smisra

Aishwariya Chakraborty (S'17) is presently pursuing her Ph.D. degree from the Advanced Technology Development Centre, Indian Institute of Technology Kharagpur, India. Her current research interests include algorithm design for IoT-based edge-clouds and distributed learning for the wireless edge. She received her M.S. and B.Tech. degree from the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur in 2019 and West Bengal University of Technology in 2015, respectively. She is a student member of IEEE and ACM. For more details, please visit https://cse.iitkgp.ac.in/∼aishchak.

Chandranath Chatterjee is a professor at Indian Institute of Technology, Kharagpur with the Department of Agricultural and Food Engineering. He received his Ph.D. degree from Indian Institute of Technology, Kharagpur, India. His research interests include surface water hydrological modeling, flood inundation modeling, remote sensing and GIS applications in hydrology, and flood forecasting using soft computing techniques. He is a past recipient of Alexander von-Humboldt Foundation Research Fellowship, Germany, 2005-06. He has received research funding support from several Indian Government funding sources. He is a life member of International Association of Hydrological Sciences (IAHS), Wallingford, U.K. He supervised the theses of several PhD and MTech scholars and has well over a hundred international journal and conference publications.

Pallav Kumar Deb is a Lead Research Engineer at Seimens, India. He completed his Ph.D. under the supervision of Dr. Sudip Misra in Smart Wireless Applications and Networking (SWAN) Lab, Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur. He also worked as an R&D consultant with SensorDrops Networks Pvt. Ltd., a STEP-incubated start-up at IIT Kharagpur. His research interests include, but are not limited to, networking, resource allocation, fog computing, constrained devices, AI/ML-based solutions, Internet of Things, Industrial Internet of Things, UAVs, Swarm of UAVs, e-health, THz communications, nano-networks.