FLAGS Framework for Comparative Analysis of Federated Learning Algorithms

Ahnaf Hannan Lodhi**, Barış Akgün, Öznur Özkasap

Department of Computer Engineering. Koç University, Rumelifeneri Sarıyer, 34450, Turkey

Abstract

Federated Learning (FL) has become a key choice for distributed machine learning. Initially focused on centralized aggregation, recent works in FL have emphasized greater decentralization to adapt to the highly heterogeneous network edge. Among these, Hierarchical, Device-to-Device and Gossip Federated Learning (HFL, D2DFL & GFL respectively) can be considered as foundational FL algorithms employing fundamental aggregation strategies. A number of FL algorithms were subsequently proposed employing multiple fundamental aggregation schemes jointly. Existing research, however, subjects the FL algorithms to varied conditions and gauges the performance of these algorithms mainly against Federated Averaging (FedAvg) only. This work consolidates the FL landscape and offers an objective analysis of the major FL algorithms through a comprehensive cross-evaluation for a wide range of operating conditions. In addition to the three foundational FL algorithms, this work also analyzes six derived algorithms. To enable a uniform assessment, a multi-FL framework named FLAGS: Federated Learning AlGorithms Simulation has been developed for rapid configuration of multiple FL algorithms. Our experiments indicate that fully decentralized FL algorithms achieve comparable accuracy under multiple operating conditions, including asynchronous aggregation and the presence of stragglers. Furthermore, decentralized FL can also operate in noisy environments and with a comparably higher local update rate. However, the impact of extremely skewed data distributions on decentralized FL is much more adverse than on centralized variants. The results indicate that it may not be necessary to restrict the devices to a single FL algorithm; rather, multi-FL nodes may operate with greater efficiency.

Keywords: Federated Learning, Hierarchical, Device-to-Device (D2D), Gossip, Cluster, Framework

1. Introduction

The critical challenges associated with centralization in Machine Learning (ML), including data aggregation, privacy and security risks, and economy of resources have accelerated interest in decentralized learning approaches. Federated Learning (FL) is one such learning framework that enables disjoint devices to learn collaboratively without the need to share data. The process entails devices to train models locally and share them for aggregation. Since its inception by Google researchers, Federated Learning (FL) proposed in [1] as FedAvg has been the foremost avenue for distributed Deep Learning (DL). Gboard-Google keyboard and Siri-Apple smart assistant, are real-world applications that have benefited tremendously from FL [2].

While FL is a major avenue towards democratized learning, there still exists a strong proclivity in FL research towards centralized orchestration. Access to updates from a large number of devices, a global model, considerable computational and storage resources have traditionally made a strong case for Centralized FL (FedAvg). However, there exist a number of challenges [2] to such an operation including adversarial attacks, malicious participation, privacy preservation, heterogeneous devices and participation, non-Independently and Identically Distributed (non-IID) data, and communication issues. Simultaneously, next generation networks [3] envision considerable serverless interaction between the devices. Advances, such as application specific data rates, Network Function Virtualization (NFV) and network slicing have enabled researchers to propose various FL algorithms based on the permissible device interactions (see Fig.1 for the depiction of these links). The proposed methods under various settings show multiple

*This work was supported by the Koç University and Iş Bank Artificial Intelligence (KUIS AI) Center Research Award and in part by the TUBITAK 2247-A Award 121C338. Funding Agency: TUBITAK 2247-A Award (Project No: 121C338).

**Corresponding author at: Graduate School of Science and Engineering, Koç University, Rumeli Feneri, Sarıyer, Istanbul 34450, Turkey

Email addresses: alodhi18@ku.edu.tr (Ahnaf Hannan Lodhi), baakgun@ku.edu.tr (Barış Akgün), oozkasap@ku.edu.tr (Öznur Özkasap)

Manuscript accepted for publishing at Elsevier Internet of Things (https://doi.org/10.1016/j.iot.2022.100638)

advantages over Centralized FL (FedAvg). Furthermore, current and future devices come equipped with multiple communication links. This fact negates the limitation of operating a single FL algorithm at all times. The challenge then arises regarding the selection of FL algorithm to employ under various conditions. The comparative characterization of FL algorithms conducted in this work is aimed at bridging this gap. The analysis conducted here paves the way for establishing economy of operation for these algorithms for various operating scenarios in a particularly heterogeneous Network Edge.

Contributions. This work aims to contrast the performance of the major FL algorithms under some of the most prevalent challenges to distributed learning at the network edge, i.e., non-IID data, noisy communication and asynchronous aggregation. The main goal is to pave the way for a multi-FL system where devices may select the most suitable FL algorithm depending on the operating characteristics.

1. A detailed comparative evaluation of the fundamental Federated Learning algorithms namely (a) *Hierarchical FL (HFL)* (b) *Device-to-Device FL (D2DFL)* (c) *Gossip FL (GFL)* (d) *Centralized FL (FedAvg)* has been conducted. In order to expand the scope of evaluation, this work also considers other FL algorithms employing multiple aggregation strategies jointly. These include: (a) *Hierarchical Device-to-Device FL (HD2DFL)* (b) *Hierarchical Gossip FL (HGFL)* (c) *Clustered Device-to-Device FL (CD2DFL)* (d) *Inter-Cluster FL (iCFL)* (e) *Inter-Cluster Device-to-Device FL (iCD2DFL)*

   Their performance is tested under ideal and noisy Device-Device (D2D), Device-Edge (D2E) and Device/Edge-Cloud (D2C/E2C) communication links. The results indicate that even in the presence of non-IID data, decentralized FL performs with a marginal loss in performance compared to centralized variants despite operating entirely in the absence of any global information. Extremely skewed data distributions, however, greatly impact decentralized FL drastically reducing convergence rate as well as accuracy.

2. A detailed study of these algorithms under various levels of device participation (such as active, inactive, stragglers) has been conducted. Decentralized FL shows marginal loss in performance when compared with centralized FL in the presence of degraded participation. Decrease

in device participation and extremely skewed data distributions have a confounding affect on all the algorithms, more so on the decentralized ones.

3. An analysis using modified Few-Shot Learning, with considerably more local updates before aggregation, has also been conducted for the FL algorithms under consideration. Decentralized algorithms with show a greater loss in accuracy than the centralized ones. However, clustered operation helps mitigate the absence of a global server while restricting all communications to the device level.

4. Finally, this work develops the FLAGS framework[1] to simulate multiple FL algorithms. The framework is designed to support ease of configuration of multiple FL algorithms and provides numerous options for generating the edge network topology for a realistic assessment.

The remainder of this paper is organized as follows: Section-2 provides some of the related developments in FL. An overview of the system model and FL is presented in Section-3. The FLAGS framework and its modules have been explained in Section-4. Details of the FL algorithms have been provided in Section-5. This is followed by a description of the experiments and performance analysis in Section-6 and 7. The paper concludes with a summary of the important findings in Section-8.

2. Related Work

The FL process is envisioned for a highly heterogeneous environment. The search of optimal application of FL has resulted in various FL algorithms designed to uniquely benefit from edge network topology. The impact of some of the most prevalent among these must first be characterized in order to adapt FL accordingly. To this end, some of the representative works that have focused on the FL algorithms and simulation frameworks have been covered in this section.

2.1. Federated Learning Algorithms

Most of the current research in Federated Learning leans heavily towards centralization. However, research indicates that careful selection of design

[1] `https://github.com/ahnaflodhi/FLAGS-FL`

parameters may yield competitive results for decentralized FL as well. [4] describes a fully decentralized FL algorithm with IID data distributed across users interacting according to a directed graph. In [5], the authors present a gossip-aggregation framework for FL. The research reports more favorable results from gossip learning based FL with uniform data distribution across the nodes. More recently, [6] suggests the application of the Decentralized Stochastic Gradient Descent (*DSGD*) [7] for D2D belief aggregation in the presence of wireless impairments. [8] proposes a Consensus-based Federated Averaging (CFA) for dense IoT networks. Their analysis suggests that serverless cooperation between devices may also yields results comparable to FedAvg. Extending their work, the authors in [9] evaluate Consensus-based Federated Averaging (CFA) and CFA-Gradient Exchange (CFA-GE) for dense IoT networks with D2D interaction. Their results support the original hypothesis indicating that although slow to converge, CFA and CFA-GE achieve performance similar to FedAvg with communication restricted only to device level. In [10], the authors try to tackle the problem of data heterogeneity in a decentralized fit learning setting. The authors propose a peer-to-peer model exchange method with model fusion using Mutual Knowledge Transfer.

Other research has also proposed hybrid FL algorithms employing hierarchical aggregation in conjunction with D2D interaction. In [11], the authors perform divergence based client grouping in a Hierarchical Federated Learning (HFL) scenario. [12] provide latency analysis of a hierarchical FL system operating in a heterogeneous cellular network where the local aggregation is conducted at Mobile Base Stations. The impact of HFL on training time and energy consumption is investigated in [13]. The same work also indicates that a trade-off between latency and computation may be achieved resulting in better performance than centralized FL. A two-time scale Hybrid FL model is proposed in [14] which compliments device-device communication with hierarchical server based aggregations. This work introduces a control algorithm scheduling global aggregations, local interactions and learning rate to achieve a convergence rate of $\mathcal{O}(1/t)$. The work in [15] uses gossip interaction between the devices before allowing them to upload their models to the hierarchical servers. The resultant FL algorithm provides near-optimal results even in the presence of reduced communication frequency and volume. [16] proposes clustering devices in the same network location using the corresponding mobile edge nodes for local aggregation operations. Their suggested method in conjunction with a cosine-similarity based device filtering

attains higher convergence speeds using less number of local updates. Then [17] experiment with FL using clustered devices where the Cluster Heads, selected from the devices, are awarded reputation scores by the members and may communicate with their one-hop neighbors. The results indicate that the proposed configuration improves network efficiency.

The work in [18] compares the communication efficiency of split learning and FL. This research evaluates the learning techniques for increasing client population as well as increasing the size of the global dataset. The results indicate that split learning favors the former whereas the latter is better served by FL. In [19], the authors compare FedAvg, Federated Stochastic Variance Reduced Gradient (FSVRG) and CO-OP FL algorithms to investigate the impact of non-IID distribution of various FL optimization techniques. However, the comparison only considers centralized FL aggregation for the mentioned optimization schemes. The results indicate that FedAvg fares better than the other two techniques for non-IID data distribution. A detailed empirical analysis of federated and gossip learning conducted in [20] shows that gossip performs competitively with federated learning.

The aforementioned and most remaining body of FL works only use centralized FL as baseline. It therefore, remains to be seen where different FL algorithms may show similar or better performance while being more efficient in other aspects such as communication, energy etc.

2.2. Existing Frameworks

Benchmarking FL takes on greater significance as the envisioned operating environment is highly diverse and different from centralized or silo-based distributed learning. It is therefore necessary that an effective benchmark be capable of allowing heterogeneity in device behavior, data distribution and connectivity. Additionally, the benchmark should be able to support different cooperation schemes that form the basis of different FL algorithms.

Various benchmarking frameworks have taken different approaches to satisfy the aforementioned objectives. These have been aimed at achieving rapid prototyping [21], higher scalability [22, 23] and realistic data heterogeneity [24, 25]. Among these options, FedML [26] additionally offers topology management by allowing configuration of multiple FL algorithms. However, it is pertinent to note that these frameworks are designed to simulate either FedAvg or a single FL algorithm for a given environment. This indicates that greater work is involved in simulating various FL algorithms to capture more accurate performance details.

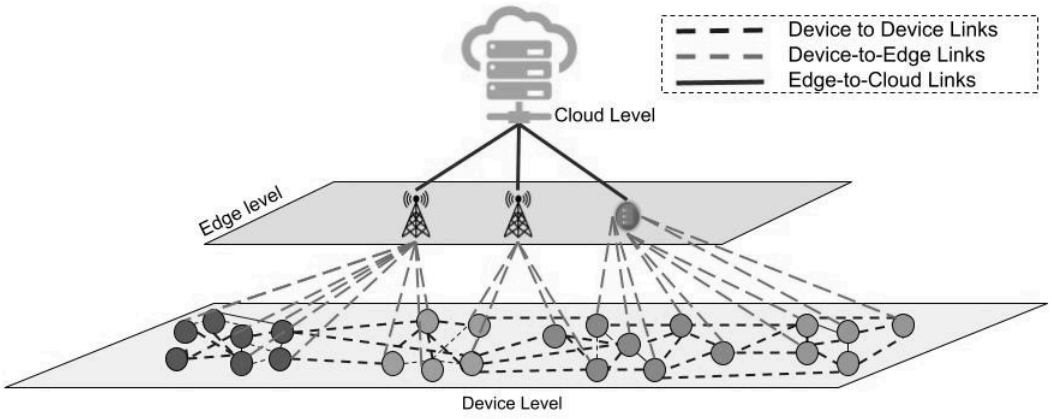


Figure 1: Network topology including Edge network levels and communication links.

This work presents the performance of key FL algorithms distinguished in their ability to utilize various network levels (Fig. 1). In order to achieve nuanced evaluation and ensure *extensibility*, this work presents the **FLAGS** framework. The framework allows each network entity, whether a node or a server, to separately emulate a realistic behavior. By implementing a range of functionality associated with each entity, this framework allows diverse network interactions enabling multiple FL algorithms to be simulated easily. Each device has a wide array of associated functions to enable multiple operating conditions and device behaviors. During the experiments, the FL algorithms are subjected to some of the key challenges facing general FL research. The empirical analysis of the performance of these FL algorithms is conducted by varying the operating parameters such as device participation, communication noise and data distribution.

3. System Model

Federated Learning offers an efficient means of distributed learning at the Edge Network. This section first establishes a *System Model* for the interaction between network entities such as nodes and servers connected by various communication links. The mathematical framework behind the FedAvg algorithm is also presented which is extended in subsequent sections to describe major FL algorithms.

3.1. Network Topology

Each of the FL algorithms in this work is evaluated over the same system model to ensure uniformity of conditions. The system model is defined using a set of nodes, edge servers and a global cloud server. A stochastic formulation is used to establish the interaction between the nodes at the device layer as well as between the device and hierarchical servers at the edge layer. A set of $\boldsymbol{N}$ static nodes are randomly distributed in $\boldsymbol{C}$ geo-proximal clusters ($\mathcal{C}$) representing natural grouping. Each cluster may be associated with its own Cluster Head randomly selected from the cluster. Cluster Head selection and Cluster membership are an active area of research, however, beyond the scope of this paper. However, the behavior in the presence of one has been replicated. One node from each cluster $\mathcal{C}_i$ is designated at random as a Cluster Head. This work intends to evaluate the performance under Clustered FL operation and thus assumes automatic membership of proximal nodes.

Each node k is assumed to have a degree n_k and possesses knowledge about its neighborhood. However, it is pertinent to note that a device's neighborhood is not restricted to its cluster and devices can be connected with those in other clusters. This ensures that the network topology at the device layer is *reachable* without involving an entity from a higher edge layer. This layout forms an undirected graph $\mathcal{G} = (\boldsymbol{N}, \xi)$ with ξ representing device links or graph edges at the device layer. The links density at the device level is controlled by a set of two parameters (γ, υ).

The probability of a link between geographically proximal devices lying within a cluster $\mathcal{C}_i$ is:

$$\underset{m,n \in \mathcal{C}_i}{P(\xi_{mn})} = \{\gamma \mid 0 < \gamma < 1\}$$

Similarly, the probability of a device-level link between two nodes belonging to distinct clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ is:

$$\underset{p \in \mathcal{C}_i, q \in \mathcal{C}_j}{P(\xi_{pq})} = \{\upsilon \mid 0 < \upsilon \leq \gamma\}$$

The parameters υ and γ are used to alter the network topology from a random placement to a highly clustered setting depending upon the target environment. For these parameters, a setting of $(\gamma \to 1,\ \upsilon << \gamma)$ indicates that the proximal groups are densely connected with almost each device sharing a link

Param	Description
$\boldsymbol{\theta^t}$	Global model parameters at time t
$\boldsymbol{\theta_k^t}$	Model parameters of node k at time t
$\boldsymbol{\theta^*}$	Optimal parameters of the global model
$\boldsymbol{\mathcal{L}}$	Global loss function
$\nabla\mathcal{L}$	Gradient of the loss
$\boldsymbol{x}, \boldsymbol{y}$	Input feature vector, reference output
α_k	Learning rate at node k
η_k	Aggregation weight associated with node k
$\mathcal{D}$	Global dataset partitioned among the nodes
$\mathcal{D}_k$	Dataset associated with node k
$\boldsymbol{N}$	Number of nodes in the network
l_k	Local loss function associated with node k
$\xi_{l,m}$	Edge link between nodes l and m
$\mathcal{C}_i$	Cluster Head (CH) of the cluster i
$\boldsymbol{C}$	Number of geo-proximal clusters in the network
d_k	Neighbor cooperation probability of node k
p_k	Edge server cooperation probability of node k

Table 1: List of parameters used in the system model and the algorithmic description.

with the other within the group. On the other hand, only a few edges exist between devices between different groups indicating a reduced probability of the devices' ability to communicate directly with devices that are farther off. In contrast, $(\gamma \to 1, \upsilon \to \gamma)$ indicates a densely connected device layer where each device is connected to a large number of other devices. Finally, $(\gamma \to 1, \upsilon = 0)$ indicates that devices only maintain connections within a proximal group without any inter-group connectivity.

The framework can simulate both ideal and noisy links with zero-mean Gaussian noise ($\mathcal{N}(0, \sigma)$). Each device has a probability d_k of participating in a neighborhood aggregation and p_k for an edge or cloud server-based aggregation. These probabilities reflect the device status (active/inactive) as well stragglers that may choose not participate in certain aggregation rounds. Furthermore, the nodes are assumed to be fully capable of undertaking local learning operations including adjusting hyperparameters.

3.2. Federated Learning Methodology

The global objective in a Federated Learning setting is a weighted sum of the local loss functions:

$$\boldsymbol{\mathcal{L}}(\boldsymbol{\theta^t}) = \sum_{k=1}^{\boldsymbol{N}} \eta_k \mathcal{L}_k(\boldsymbol{\theta_k^t}) \tag{1}$$

where $\boldsymbol{\mathcal{L}}(\boldsymbol{\theta}^t)$ represents the global loss function and $\mathcal{L}_k(\boldsymbol{\theta_k})$ and η_k are the local loss function and weight associated with node k at time t. The FL objective is to minimize the weighted sum of the individual loses of $\boldsymbol{N}$ devices

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^M} \boldsymbol{\mathcal{L}}(\boldsymbol{\theta^t}) = \min_{\boldsymbol{\theta}} \sum_{k=1}^{\boldsymbol{N}} \eta_k \mathcal{L}_k(\boldsymbol{\theta_k^t}) \tag{2}$$

Each $\mathcal{L}_k(\boldsymbol{\theta_k^t})$ at time t is associated with an M-dimensional model parameterized by $\boldsymbol{\theta_k^t} \in \mathbb{R}^M$ and the weight η_k for node k. The initial parameters of the model, $\boldsymbol{\theta}^t$, are shared with the devices through a server. The local loss at a device is calculated through the training as follows

$$\mathcal{L}_k(\boldsymbol{\theta_k^t}) = \frac{1}{|\mathcal{B}_k|} \sum_{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathcal{B}_k} l(\boldsymbol{x_i}, \boldsymbol{y_i}; \boldsymbol{\theta}_k^t) \tag{3}$$

where $l(\mathbf{x}_i, \mathbf{y}_i; \boldsymbol{\theta_k^t})$ represents the loss of the machine learning task (e.g cross-entropy, mean squared error) calculated for the local model for the current model parameters. This is calculated for each point $(\boldsymbol{x_i}, \boldsymbol{y_i})$ in the minibatch $\mathcal{B}_k$ sampled from the local dataset $\mathcal{D}_k$. Using the principles of Distributed Stochastic Gradient Descent (DSGD), each device with the learning rate α_{kr} for a round r, updates its model as:

$$\boldsymbol{\theta}_k^{t+1} = \boldsymbol{\theta}_k^t - \alpha_k^t \nabla \mathcal{L}_k^t(x_i, y_i; \boldsymbol{\theta}_k^t) \tag{4}$$

where $\nabla \mathcal{L}_k^t(x_i, y_i; \boldsymbol{\theta}_k^t)$ are the gradients calculated by node k during the local update round t and α_k^t is the learning rate associated with kth node at time t. The updated model parameters from Eq. 4 or the gradients calculated above are then shared with the server. The global model is obtained through the weighted aggregation of the model parameters

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \sum_{k=1}^{\boldsymbol{N}} \eta_k \boldsymbol{\theta}_k^t \tag{5}$$

For an alternate FL scheme employing gradient sharing, Eq. 5 can be formulated as:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \alpha^{t} \sum_{k=1}^{\boldsymbol{N}} \eta_k \nabla \mathcal{L}_k^t(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\theta}_k^t) \tag{6}$$

where $\boldsymbol{\theta}^t$ refers to the parameters of the global model used at the start of round t and α^t is the learning rate employed by the server. The learning rate α_t is typically kept smaller [27] compared with centralized learning on large datasets. As evident from Eq. 2-6, the entire Federated Learning process bypasses the need for the devices to share data at any stage, offering a strong privacy advantage over centralized forms of ML. It instead relies on the communicating model parameters indicated in Eq. 5 and Eq. 6 to the local server. The server is also required to share the aggregated model with the connected nodes for the subsequent rounds adding to the overall communication volume needed to execute FL successfully. Such servers are typically located at the network cloud level, offering multiple services. Thus, frequent communication to and from these servers becomes an extremely expensive operation in addition to straining the latency requirements of various edge services in addition to subjecting the shared models to additional adversarial risks.Table reftab:params presents the list of parameters used in this work.

4. FLAGS Framework

The FL algorithms are envisioned for a highly heterogeneous setting with limited prior information. Accurate assessment of their performance requires that the simulation framework provide repeatable and uniform conditions, support easily configurable FL algorithms, possess flexibility to implement new ones and cater to multiple types of communication links. This work develops **F**ederated **L**earning **A**l**G**orithms **S**imulation (**FLAGS**) framework with the intent of making it easier to configure multiple FL algorithms under a wide range of operating conditions.

4.1. FLAGS: Features

FLAGS is a lightweight FL prototyping framework allowing for an accurate assessment of multiple FL algorithms for highly realistic network topologies. Inherent in FLAGS are different levels of device participation (controlled by parameters p_k for participation in central aggregation and d_k for controlling neighborhood aggregation), device selection mechanism for

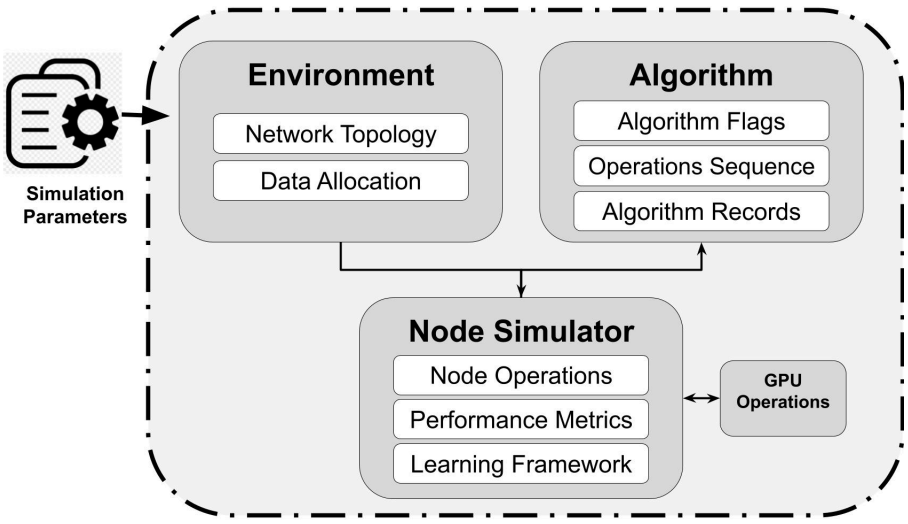


Figure 2: FLAGS Framework Architecture

nodes and servers as well as generating multiple data distributions including IID, non-IID and extremely-skewed non-IID data partitions. It also supports synchronous and deadline-based asynchronous operation to replicate heterogeneous device behavior where devices training progresses at different paces. The highlight of FLAGS lies in its capability to configure and subject multiple FL algorithms to a realistic multi-tiered environment which allows users to propose the best environment-FL fit.

4.2. FLAGS Architecture

FLAGS is developed so that multiple FL algorithms may be simulated using a single framework and tested under a unified set of conditions. The FLAGS architecture is highly modular and uses three different functional blocks to control the respective set of operations, as depicted in Fig. 2. These are the *Environment, Algorithm* and *Node* modules.

The *Environment* block is responsible for configuring the network topology for FL operations. It controls the generation of the network topology and the respective links while providing access to related information. Once configured, this module enables interaction within various entities of the network. The device layer is configured with the devices grouped into $\boldsymbol{C}$ proximal cluster. Each of the devices is also connected to the respective server at the Edge Layer. These hierarchical servers, in turn, are then connected

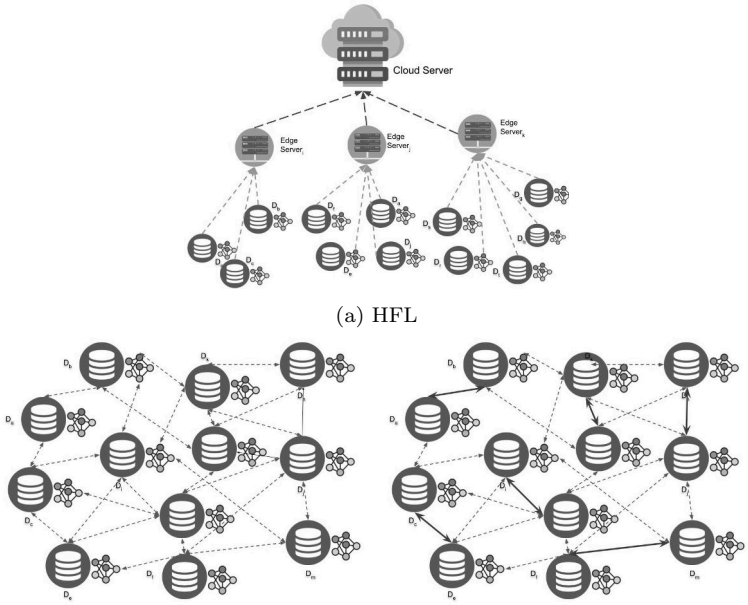
Flags	D2D Aggregation	Edge Aggregation	Cluster Aggregation	Inter-Cluster Aggregation
FedAvg	'CServer'	False	False	False
HFL	False	True	False	False
D2DFL	'D2D'	False	False	False
GFL	'Random'	False	False	False
HD2DFL	'D2D'	True	False	False
HGFL	'Random'	True	False	False
CFL	False	False	True	False
iCFL	False	False	True	True
iCD2DFL	'D2D'	False	True	True

Table 2: FL Algorithms with the respective *flag* settings for the **FLAGS** framework

to a global server representing the cloud level. The overall configuration in the environment module affords greater flexibility in conjunction with the corresponding *flags*. This feature allows multiple FL algorithms to be simulated by just changing the *flags* configuration. This distinguishes the *FLAGS* framework from others since those have been designed to simulate a single topology.

The *Algorithm* module manages the configuration and sequence of operations for all FL algorithms. It makes use of a set of *flags* to generate unique interaction between various elements of the environment. FLAGS implements four different flags to generate the required interaction: (a) Device Aggregation (b) Edge Aggregation (c) Cluster Aggregation and (d) Inter–Cluster Aggregation. The Device Aggregation flags control whether a node is allowed to share information with its peers or not. This may be within the devices' own neighborhoods or through random pairings including multi-hop interactions. The Edge Aggregation, when enabled, enables the devices to share their local models with the hierarchical servers. Cluster and Inter-Cluster Aggregations have been included in the framework to support expandability for FL scenarios with device clusters. Employing a flag-based mechanism allows the framework to remain configurable for addition of more FL algorithms.The *flag* settings for the various FL algorithms covered within this work have been shared under Table 2.

Finally, the *Node* block implements all device level operations including training, testing and communication while maintaining records all perfor-



(a) HFL

(b) D2DFL

(c) GFL

Figure 3: Main Federated Learning Algorithms. Dashed lines represent links between various entities whereas only solid lines in (c) depict active established-pair links in GFL.

mance metrics. This module enables the nodes to communicate with both Environment and Algorithm blocks to obtain the required topological and permissible-interaction information. The *devices* and *servers* both are generated using this block imbuing them with a range of functionality available to modern devices for conducting the learning operations.

5. Federated Learning Algorithms

Federated Learning was initially conceived as a two-tiered learning framework alternating between clients/nodes performing the training and the cloud servers aggregating the local models to yield a global model. In order to increase the efficiency, the envisaged FL was further spread over the Edge Network to benefit not only from the Edge devices capabilities but also address the challenges inherent with centralization. This evolution has yielded various FL algorithms suitable to a variety of scenarios including vehicular-networks, dense IoT networks, cross-silo operations between relatively large data centers. These algorithms employ three different types of communica-

Algorithm 1: Hierarchical Federated Learning

Input: Initial Model parameters $\boldsymbol{\theta}_0$, $\mathcal{S}$ hierarchical servers with device association

Output: Global model parameters $\boldsymbol{\theta}_{t\to\infty}$

Initialization by global server: $\boldsymbol{\theta}_0$ received by each hierarchical server S_i and broadcast among the connected nodes

for *Nodes* $k = 1 \ldots\ldots \boldsymbol{N}$ ***(in parallel)*** **do**

  Perform Local update for e epochs

  $\boldsymbol{\theta_k^{t+\tau}} = \boldsymbol{\theta}_k^t - \alpha_k^t \nabla \mathcal{L}_k(x_m, y_m; \boldsymbol{\theta}_k^t)$

  TX: With a probability p_k, send the $\boldsymbol{\theta}_k^{t+\tau}$ to the hierarchical server for aggregation

for *Servers* $\mathcal{S}_i \; \forall \; i = 1 \ldots\ldots \mathcal{K}$ ***(in parallel)*** **do**

  AGG: Aggregate the models received from the n_i nodes

  $\boldsymbol{\theta}_{S_i}^{t+T} = \boldsymbol{\theta}_{S_i}^t + \sum_{j \in n_i} w_j \boldsymbol{\theta}_j^{t+\tau}$

if *round % f == 0* **then**

  for *Servers* $S_i \; \forall \; i = 1 \ldots\ldots \mathcal{K}$ ***(in parallel)*** **do**

    Upload models to the global server

  Global Aggregation

  $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \sum_{j \in \mathcal{K}} \eta_j \boldsymbol{\theta}_j^{t+T}$

  Send model back to Edge Servers S_i

  Edge Servers S_i broadcast model parameters $\boldsymbol{\theta}^{t+1}$ to all nodes connected respectively

tion links as shown in Fig. 1 associated with different communication costs: (a) Device-to-Device (D2D) (b) Device-to-Edge (D2E) (c) Edge-to-Cloud (E2C)

D2D link are limited to the device level whereas D2E links connect the nodes with the edge servers. Finally, E2C links connect the edge servers to the core network. While D2D links and D2E links operate independently, E2C links require the latter for information to be transmitted from the devices to the edge server for onward transmission to the core network. All subsequent communication analyses in this work use this fact to establish effectiveness of a particular form of communication in FL operations.

A key aspect in major FL literature is the absence of cross-configuration

comparison and evaluation of FL algorithms. Such an analysis promises to identify optimal employment of these FL algorithms based on the operating environment and nature of the problem being addressed. This section first provides a description of these algorithms followed by details of the experiments in the subsequent sections.

5.1. Hierarchical Federated Learning

Hierarchical Federated Learning (HFL) introduces intermediate model aggregation closer to the data origin using an edge server [13]. This FL algorithm introduces $\mathcal{F}$ aggregation layers between the nodes and the global cloud server, $\mathcal{F}$ increasing toward the cloud layer. Cellular base stations or Mobile Edge Computing (MEC) Servers are envisioned to realize this role [28] for their respective connected devices. Each network level, as depicted in Fig. 3(a), aggregates the models received from the previous layer and passes them to the next layer in the hierarchy. This scheme offers reduction in the overall upstream communication as well progressively offloading the computational load at the edge network.

Each hierarchical layer $\mathcal{F}$ houses $\mathcal{K}_{\mathcal{F}}$ edge servers. Each server at layer $\mathcal{F} = 1$ aggregates models from the devices linked to it respectively. The pseudocode for HFL have been laid out in Alg. 1. The initial model parameters i.e. $\boldsymbol{\theta}_0$ are initialized by the global server $\mathcal{S}_g$ and shared with the nodes. The nodes in turn update the model over a mini-batch of size $\mathcal{M}$ to yield $\boldsymbol{\theta}_k^{t+1}$. After completing e local updates, each node shares its updated parameters with the hierarchical server S_i with a probability p_k. The servers aggregate the received models which is then shared with the servers in the next layer/global server after f aggregation rounds. At this stage, the global server aggregates the input models from the $\mathcal{K}$ hierarchical servers and generates the global model $\boldsymbol{\theta}^{t+\tau}$ which are then shared through the local servers with the associated nodes.

5.2. Device-to-Device Federated Learning

With Device-to-Device (D2D) communication [29], FL can be operated without the requirement of a central aggregator [9]. In a purely serverless FL framework, the nodes communicate with their immediate neighbors and resort to local aggregation Fig. 3(b). At a given time t, the nodes share their locally learned parameters $\boldsymbol{\theta}_k^t$ with their neighborhood n_k. The recipients in turn perform aggregation of the received models combining the incoming

Algorithm 2: Device-to-Device Federated Learning

Input: Model parameters θ_k for each node, degree n_k
Output: D2D Aggregated Models

for *Nodes* $k = 1 \ldots\ldots \boldsymbol{N}$ ***in parallel*** **do**
  Update local model for e epochs
  $\boldsymbol{\theta_k^{t+\tau}} = \boldsymbol{\theta}_k^t - \alpha_k^t \nabla \mathcal{L}_k(x_m, y_m; \boldsymbol{\theta}_k^t)$
  Send model to 1-hop neighbors n_k with probability d_k
  Receive models from n_k 1-hop neighbors
  Perform weighted aggregation for the received models
  $\boldsymbol{\theta}_k^{t+1} = \boldsymbol{\theta}_k^{t+\tau} + \sum_{j \in n_k} \eta_j \boldsymbol{\theta}_j^{t+\tau}$

parameters $\boldsymbol{\theta}_k^t$ with corresponding weights $\{\eta_j \forall j \in n_k\}$ to generate $\boldsymbol{\theta}_k^{t+1}$: The entire operation of D2DFL is depicted in Alg. 2.

Algorithm 3: Gossip Federated Learning

Input: Model parameters $\boldsymbol{\theta}_k$, optimization parameters
Output: Gossip-aggregated model

for *Nodes* $k = 1 \ldots\ldots \boldsymbol{N}$ ***in parallel*** **do**
  $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ Model-Update$(\boldsymbol{\theta}_k^(t))$
  $\boldsymbol{\theta}_k^{t+1} \longleftarrow$ *Gossip Exchange* $(\theta_k^{t+\tau})$
Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`
  for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**
    $\xi_b = (x_b, y_b)$ is the data sample pairs in b
    $\boldsymbol{\theta_l^{t+\tau}} = \boldsymbol{\theta}_l^t - \alpha^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$
  return $\boldsymbol{\theta}_l^{t+\tau}$
Function `Gossip Aggregate(`$\boldsymbol{\theta}_l^t$`):`
  Perform aggregation handshake with a random node
  Exchange models for aggregation between the pair
  Aggregate received model
  $\boldsymbol{\theta}_l^{t+T} = \boldsymbol{\theta}_l^{t+\tau} + \eta_j \boldsymbol{\theta}_j^{t+\tau}$
  return $\boldsymbol{\theta}_i^{t+T}$

5.3. Gossip Federated Learning

Gossip communication is another form of decentralized communication wherein the nodes communicate with randomly selected peer nodes. Gossip Federated Learning (GFL) was among the early fully decentralized FL frameworks to have been proposed [30]. For a network with $\boldsymbol{N}$ nodes, the nodes form random pairs with other nodes from the network forming a gossip pair ($i,\ j\ \forall\ \ i,\ j\ \in\ \ \boldsymbol{N}$) as shown in Fig. 3(c). Once the connection has been established, the pair exchange their locally updated models $\boldsymbol{\theta}_i^t$, $\boldsymbol{\theta}_j^t$ and in turn aggregate with the received model parameters to generate $\boldsymbol{\theta}_i^{t+1}$ and $\boldsymbol{\theta}_j^{t+1}$. The process results in considerably curtailed communication costs. The details of the operation have been provided in Alg. 3.

5.4. Hierarchical Device-to-Device Federated Learning

Hierarchical D2DFL employs aggregation at the device levels, edge servers and cloud servers. Thus the D2D, D2E and E2C links all are employed to enable such operation [14]. The proposed mechanism allows for devices to perform local updates on their model parameters $\boldsymbol{\theta}_k^t$ and share them with their n_k neighbors leading to device level aggregation. Simultaneously, the devices also share the model parameters with the connected servers $\mathcal{S}_i$. The servers in turn aggregate the received models and share along the hierarchy $\mathcal{F}$ eventually leading up to the global server $\mathcal{S}_g$ as indicated in Fig. 3(d). The global model $\boldsymbol{\theta}^{t+\tau}$ gets disseminated back to the nodes through the respective servers and the process is continued till convergence. The details of the entire process has been outlined in Alg. 4. While greater cooperation is envisioned with this algorithm, the learning requires communication at both device and upstream level.

5.5. Hierarchical Gossip Federated Learning

Hierarchical FL, as described in Alg.1, allows device layer to interact with Edge Servers to aggregate local models resulting in sub-global models. These models are then passed onto global server for generating the global model $\boldsymbol{\theta}^t$. HD2DFL builds on this configuration by allowing devices layer entities to interact among themselves before sharing the local aggregated models with the higher edge layers. The performance as shall be seen in Sec-7 indicate better performance albeit at the communication volume which is almost the joint sum of HFL and D2DFL. This observation led the researchers in [15] to introduce gossip steps at the device level instead of the full neighborhood

aggregation as indicated in Alg 5.

Algorithm 4: Hierarchical-D2D Federated Learning

Input: Initial Model parameters $\boldsymbol{\theta}_0$, Neighborhood information $\mathcal{G}_k = \{\boldsymbol{N}, \xi_k\}$, Hierarchical servers K with device association

Output: Global model parameters $\boldsymbol{\theta}_{t\to\infty}$

Initialization by global server: $\boldsymbol{\theta}_0$ received by each hierarchical server S_i and broadcast among the connected nodes n_i

for *Node* $k = 1 \ldots\ldots \boldsymbol{N}$ ***in parallel*** **do**

  $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow Model\ Update(\boldsymbol{\theta}_{\boldsymbol{k}}^{\boldsymbol{t}})$

  $\boldsymbol{\theta}_k^{t+\delta} \longleftarrow\ D2D\ Aggregate(\boldsymbol{\theta}_{\boldsymbol{k}}^{\boldsymbol{t+\tau}}, n_k)$;

if *aggregation round* $\%\ f == 0$ **then**

  for *Servers* $S_i\ \forall\ i = 1 \ldots\ldots H$ ***(in parallel*** **do**

    Sample q from the associated nodes and request parameters

    Aggregate received parameters: $\boldsymbol{\theta}_{S_i}^{t+\tau} = \boldsymbol{\theta}_{S_i}^{t} + \sum_{j\in q} w_j \boldsymbol{\theta}_j^t$

  for *Servers* $S_i\ \forall\ i = 1 \ldots\ldots H$ ***in parallel*** **do**

    Upload model $\boldsymbol{\theta}_{S_i}^t$ to the global server

  Global Aggregation : $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \sum_{j\in S_i} \eta_j \boldsymbol{\theta}_j^t$

  Send model back to nodes through respective Edge Servers S_i

Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`

  for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**

    $\xi_b = (x_b, y_b)$ is the data sample pairs in b

    $\boldsymbol{\theta}_{\boldsymbol{l}}^{\boldsymbol{t+\tau}} = \boldsymbol{\theta}_l^t - \alpha_l^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$

  return $\boldsymbol{\theta}_l^{t+\tau}$

Function `D2D-Aggregate(`$\boldsymbol{\theta}_k^t,\ n_l$`):`

  Exchange model with 1-hop neighbors n_l with probability d_l

  Perform weighted aggregation for the received models

  $\boldsymbol{\theta}_l^{t+\delta} = \boldsymbol{\theta}_l^{t+\tau} + \sum_{j\in n_l} \eta_j \boldsymbol{\theta}_j^{t+\tau}$

  return $\boldsymbol{\theta}_l^{t+\delta}$

The nodes communicate at the device level using random pairings resulting in gossip communication. These random node pairs exchange local models during this stage to perform local aggregation yielding $\boldsymbol{\theta}_k^{t+\tau}$ for $k = 1 \ldots\ldots \boldsymbol{N}$ network nodes. Subsequently, these gossip-aggregated models

are shared with the edge servers for hierarchical aggregation and subsequently cloud aggregation respectively.

Algorithm 5: Hierarchical-Gossip Federated Learning

Input: Initial Model parameters $\boldsymbol{\theta}_0$, Neighborhood information $\mathcal{G}_k = \{\boldsymbol{N}, \xi_k\}$, Hierarchical servers K with device association

Output: Global model parameters $\boldsymbol{\theta}_{t\to\infty}$

Initialization by global server: $\boldsymbol{\theta}_0$ received by each hierarchical server S_i and broadcast among the connected nodes n_i

for *Node* $k = 1 \ldots\ldots \boldsymbol{N}$ ***in parallel*** **do**

  $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ *Model Update* $(\boldsymbol{\theta}_k^t)$

  $\boldsymbol{\theta}_k^{t+\delta} \longleftarrow$ *Gossip Exchange* $(\boldsymbol{\theta}_k^{t+\tau})$

if *aggregation round* $\%\ f == 0$ **then**

  for *Servers* $S_i\ \forall\ i = 1 \ldots\ldots H$ (***in parallel*** **do**

    Sample q from the associated nodes

    for $j \in$ *Sampled Nodes* **do**

      Share model parameters $\boldsymbol{\theta}_j^{t+\delta}$ with the server

    Aggregate the received parameters

    $\boldsymbol{\theta}_{S_i}^{t+T} = \boldsymbol{\theta}_{S_i}^{t+\delta} + \sum_{j\in q} w_j \boldsymbol{\theta}_j^{t+\delta}$

  for *Servers* $S_i\ \forall\ i = 1 \ldots\ldots H$ (***in parallel*** **do**

    Upload model $\boldsymbol{\theta}_{S_i}^{t+T}$ to the global server

  Global Aggregation : $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \sum_{j\in S_i} \eta_j \boldsymbol{\theta}_j^{t+T}$

  Send model back to the nodes through respective Edge Servers S_i

Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`

  for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**

    $\xi_b = (x_b, y_b)$ is the data sample pairs in b

    $\boldsymbol{\theta_l^{l+\tau}} = \boldsymbol{\theta}_l^t - \alpha_l^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$

  return $\boldsymbol{\theta}_l^{t+\tau}$

Function `Gossip Aggregate(`$\boldsymbol{\theta}_l^t$`):`

  Perform aggregation handshake with a random node

  Exchange models for aggregation between the pair

  Aggregate received model : $\boldsymbol{\theta}_l^{t+\zeta} = \boldsymbol{\theta}_l^{t+\tau} + \eta_j \boldsymbol{\theta}_j^{t+\tau}$

  return $\boldsymbol{\theta}_l^{t+\zeta}$

5.6. Clustered Federated Learning

Clustered FL (CFL) builds on the fully decentralized Federated Learning while aiming to reduce communication volume by clustering operations. The nodes in proximal locations are grouped into Clusters formed around a Cluster Head (CH). The devices become a Cluster-Member (CM) by associating with a Cluster Head $\mathcal{C}_i$, itself a device. During the training phase, the device perform local updates of their own models i.e $\boldsymbol{\theta}_k^t$. At the aggregation stage, each CM i shares its local model with the CH i.e $\mathcal{C}_i$, with a probability d_k. Subsequently, all the received models are aggregated to generate the sub-global model $\boldsymbol{\theta}_{\mathcal{C}_i}^t$. The aggregated model is shared with the respective CMs of i^{th} cluster. The process continues until an acceptable threshold is reached. The details of the entire operation are presented in Alg. 6.

Algorithm 6: Clustered Federated Learning

Input: Model parameters θ_k for each node, degree n_k
Output: D2D Aggregated Models

Devices $i = 1.....\boldsymbol{C}$ chosen as Cluster Head (CH) of cluster $\mathcal{C}_i$
for *free nodes* $k = 1.....\boldsymbol{N} - \boldsymbol{C}$ **do**
    Join $\mathcal{C}_i \ \forall \ i \in \boldsymbol{C}$ as a Cluster-Member (CM)
for *Cluster Head* $\mathcal{C}_i \forall \ i = 1 \ldots\ldots \boldsymbol{C}$ *in parallel* **do**
    Send model to CM $\{p \ : \ p \in \mathcal{C}_i\}$
    for *Nodes* $k \ = \ 1 \ \ldots\ldots \boldsymbol{N}$ ***in parallel*** **do**
        $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ Model-Update $(\boldsymbol{\theta}_k^t)$
    Receive models from CM $\{p \ : \ p \in \mathcal{C}_i\}$ with probability d_p
    Perform weighted aggregation of received models
    $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1} = \boldsymbol{\theta}_{\mathcal{C}_i}^{t+\tau} + \sum_{j \in C_i} \eta_j \boldsymbol{\theta}_j^{t+\tau}$
Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`
    for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**
        $\xi_b = (x_b, y_b)$ is the data sample pairs in b
        $\boldsymbol{\theta_l^{t+\tau}} = \boldsymbol{\theta}_l^t - \alpha_l^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$
    return $\boldsymbol{\theta}_l^{t+\tau}$

5.7. Clustered Device-to-Device Federated Learning (CD2DFL)

Clustered D2DFL (CD2DFL) builds on the fully decentralized Federated Learning while aiming to reduce communication volume by clustering oper-

ations. The devices join a cluster by associating with a Cluster Head $(CH)_i$, itself a device. The Cluster-Members (CMs) in this case, continue local updates and D2D aggregating models over their respective neighborhoods. After a number of local aggregation rounds, each CH orchestrates a cluster-level aggregation wherein the CMs share their respective models with the CH. Once aggregated, the CMs receive the aggregated model $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1}$ from the CH and continue with the D2DFL. Alg. 7 outlines the overall procedure for CD2DFL.

Algorithm 7: Clustered Device-to-Device Federated Learning

Input: Model parameters θ_k for each node, degree n_k
Output: D2D Aggregated Models

Devices $i = 1.....\boldsymbol{C}$ chosen as Cluster Head (CH) of cluster $\mathcal{C}_i$
for *free nodes* $k = 1.....\boldsymbol{N} - \boldsymbol{C}$ **do**
    Join $\mathcal{C}_i \ \forall \ i \in \boldsymbol{C}$ as a Cluster-Member (CM)
for *Cluster Head* $\mathcal{C}_i \forall \ i = 1 \ldots\ldots \boldsymbol{C}$ *in parallel* **do**
    Send model to CM $\{p \ : \ p \in \mathcal{C}_i\}$
    for *Nodes* $k \ = \ 1 \ \ldots\ldots \boldsymbol{N}$ ***in parallel*** **do**
        $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ Model-Update $(\boldsymbol{\theta}_k^t)$
        $\boldsymbol{\theta}_k^{t+T} \longleftarrow$ D2D-Aggregate $(\boldsymbol{\theta}_k^{t+\tau}, n_k)$
    Receive models from CM $\{p \ : \ p \in \mathcal{C}_i\}$ with probability d_p
    Perform weighted aggregation
    $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1} = \boldsymbol{\theta}_{\mathcal{C}_i}^{t+T} + \sum_{j \in C_i} \eta_j \boldsymbol{\theta}_j^{t+T}$
Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`
    for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**
        $\xi_b = (x_b, y_b)$ is the data sample pairs in b
        $\boldsymbol{\theta_l^{t+\tau}} = \boldsymbol{\theta}_l^t - \alpha_l^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$
    return $\boldsymbol{\theta}_l^{t+\tau}$
Function `D2D-Aggregate(`$\boldsymbol{\theta}_l^t, \ n_l$`):`
    Exchange models with 1-hop neighbors n_l with probability d_l
    Perform weighted aggregation : $\boldsymbol{\theta}_l^{t+\delta} = \boldsymbol{\theta}_l^{t+\tau} + \sum_{j \in n_l} \eta_j \boldsymbol{\theta}_j^{t+\tau}$
    return $\boldsymbol{\theta}_l^{t+\delta}$

5.8. Inter-Cluster Device-to-Device Federated Learning

Clustered FL in the presence of D2D interactions significantly improves local cooperation. In this regard, the devices ability to interact within its neighborhood and subsequent consolidation at Cluster Head (CH) results in a greater participation in the overall aggregation scheme. In order to further expand this functionality, this algorithm allows the Cluster Heads to communicate the other CH's using a gossip mechanism. While the rest of the learning and aggregation process remains similar to CD2DFL, the aggregation at the cluster head level is followed by the CH exchanging the locally aggregated cluster model $\boldsymbol{\theta}^t_{\mathcal{C}_i}$ with other CH $\{\mathcal{C}_j | i \neq j\}$ using randomized gossip. Each CH $\mathcal{C}_i$ is allowed a limited number of gossip steps before sharing the gossip-aggregated model $\boldsymbol{\theta}^{t+\tau}_{\mathcal{C}_i}$ with its respective CMs. As shown in Alg. 8, this allows the both of communication to remain restricted to proximal locations while still enabling models learned in farther clusters to be acquired.

5.9. Centralized to Decentralized Spectrum

The FedAvg is on the centralized end of the FL algorithms spectrum whereas GFL, D2DFL and CD2DFL are on the decentralized end. HFL, HD2DFL and HGFL avail a mix of centralized and decentralized operations. iCFL and iCD2DFL, despite being fully decentralized, mimic hierarchical behavior at the device level. The boundaries between centralized and decentralized operations FL algorithms thus remain fluid allowing present and

future algorithms to benefit from both types of interactions.

Algorithm 8: Inter-Cluster Device-to-Device Federated Learning (iCD2DFL)

Input: Model parameters θ_k for each node, degree n_k
Output: D2D Aggregated Models

Devices $i = 1.....\boldsymbol{C}$ chosen as Cluster Head (CH) of cluster $\mathcal{C}_i$
for *free nodes* $k = 1.....\boldsymbol{N} - \boldsymbol{C}$ **do**
    Join $\mathcal{C}_i \ \forall \ i \in \boldsymbol{C}$ as a Cluster-Member (CM)
for *Cluster Head* $\mathcal{C}_i \forall \ i = 1 \ldots\ldots \boldsymbol{C}$ *in parallel* **do**
    Send model to CM $\{p \ : \ p \in \mathcal{C}_i\}$
    for *Nodes* $k \ = \ 1 \ \ldots\ldots \boldsymbol{N}$ ***in parallel*** **do**
        $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ Model-Update $(\boldsymbol{\theta}_k^t)$
        $\boldsymbol{\theta}_k^{t+T} \longleftarrow$ D2D-Aggregate $(\boldsymbol{\theta}_k^{t+\tau}, n_k)$
        Share model with CH $\mathcal{C}_i$ with probability d_k
    Aggregate models received from CM $\{p \ : \ p \in \mathcal{C}_i\}$
    $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+\beta} = \boldsymbol{\theta}_{\mathcal{C}_i}^{t+T} + \sum_{j \in \mathcal{C}_i} \eta_j \boldsymbol{\theta}_j^{t+T}$
    $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1} \longleftarrow$ *Gossip Exchange* $(\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1})$
Function `Model-Update(`$\boldsymbol{\theta}_s^t$`):`
    for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_s$ **do**
        $\xi_b = (x_b, y_b)$ is the data sample pairs in b
        $\boldsymbol{\theta_s^{t+\tau}} = \boldsymbol{\theta}_s^t - \alpha_s^t \nabla \mathcal{L}_s(\xi_b; \boldsymbol{\theta}_s^t)$
    return $\boldsymbol{\theta}_k^{t+\tau}$
Function `D2D Aggregate(`$\boldsymbol{\theta}_l^t, \ n_l$`):`
    Exchange models with 1-hop neighbors n_l with probability d_l
    Perform weighted aggregation : $\boldsymbol{\theta}_l^{t+\delta} = \boldsymbol{\theta}_l^{t+\tau} + \sum_{j \in n_l} \eta_j \boldsymbol{\theta}_j^{t+\tau}$
    return $\boldsymbol{\theta}_l^{t+\delta}$
Function `Gossip Aggregate(`$\boldsymbol{\theta}_{\mathcal{C}}^t$`):`
    for $g = 1 \ldots\ldots \zeta$ *gossip rounds* **do**
        Perform aggregation handshake with a randomly selected $\mathcal{C}_j$
        Exchange and aggregate models between the pair
        $\boldsymbol{\theta}_i^{t+1} = \boldsymbol{\theta}_i^{t+\tau} + \eta_j \boldsymbol{\theta}_j^{t+\tau}$
    return $\boldsymbol{\theta}_i^{t+1}$

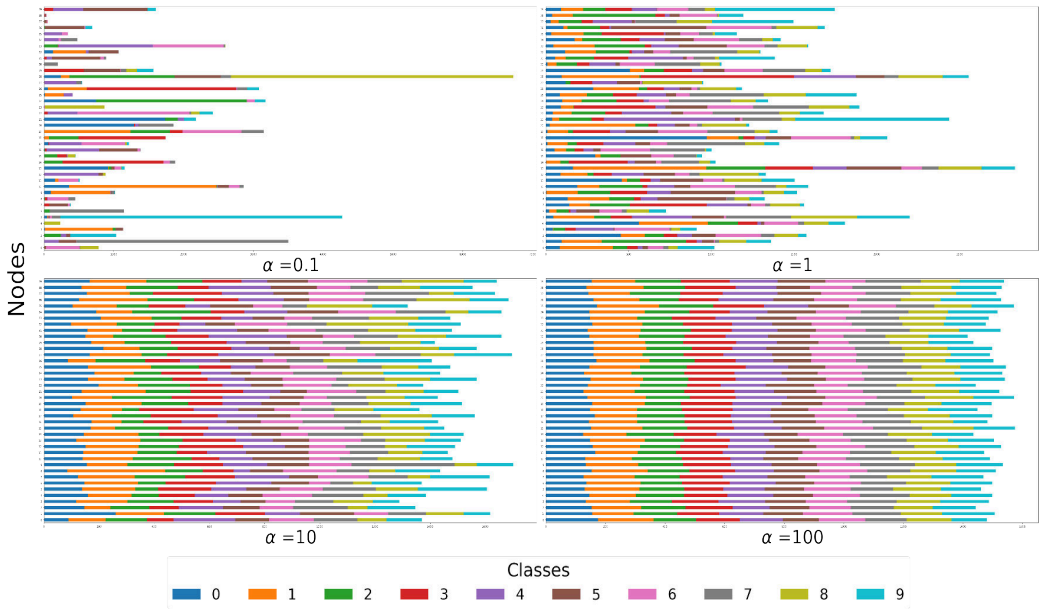


Figure 4: Synthetic data distributions for MNIST dataset across 40 nodes derived from Dirichlet Distribution for different α values.

6. Experiments and Evaluation

A uniform evaluation strategy was devised to conduct an objective assessment of the various FL algorithms. The experiments performed in this work use the MNIST [31] and FashionMNIST datasets. [32] A non-IID data distribution remains of primary interest in this work. The experimentation included subjecting the FL algorithms to ideal and noisy communication, probabilities of participation, data distribution and aggregation frequency. Additionally, these algorithms were also subjected to Few-Shot Learning details of which have been shared subsequently.

Non-IID Distribution. The non-IID data partitions are generated using the Dirichlet Distribution [33] parametrized by its concentration parameter α. The value of α controls the degree of non-IID sampling spread across the clients with lower values resulting in higher imbalance. Lower values of α result in more skewed datasets as shown in Fig. 4. A value of α between 1 and 10 results in a typically non-IID data distribution whereas lower values of $\alpha < 1$ result in extreme non-IID distributions. On the other end, values of $\alpha > 100$ result in increasingly IID distributions. The evaluation in this work

uses $\alpha = 0.1$ and $\alpha = 1.0$ for generating highly non-IID distributions.

Extremely Skewed non-IID Distribution. This work also employs an extremely skewed 2-class and 3-class non-IID cases where each node is trained using data samples from two and three classes only respectively. Each node is randomly assigned the determined number of classes and the individual class data is grouped into *shards* of 50 images each. The nodes are then assigned a randomly chosen number of shards sampled from their respective classes.

Neural Network and Development Environment. The DNNs used for evaluations based on MNIST and Fashion MNIST datasets are comprised of two 2D–Convolutional blocks followed by a Dropout and two linear layers. The modes are instantiated with the same weights while training is conducted using the SGD optimizer with a learning rate $\eta = 0.01$. The entire framework has been developed in Python whereas the learning algorithms were implemented using PyTorch. The neural network trained for MNIST and Fashion MNIST contains approximately 2 million parameters oth neural networks use Rectified Linear Unit (ReLU) activations with all layers except the final layer which uses Log-Softmax activation to generate the class probabilities.

Training Regime. The training environment assumes a network of 40 nodes, all initiated with similar weights. Our framework allows for each node to control the number of local updates as well as the learning rate and other hyperparameters. However, for this work, only variation in number of training epochs has been conducted across the devices.

Operating Characteristics. The network topology generated for the experiments consists of $\boldsymbol{N} = 40$ nodes that are divided into $\mathcal{C} = 7$ clusters. In order to ensure *reachability*, $\gamma = 0.95$ and $\upsilon = 0.1$ have been used. The former ensures that each device has a direct link to 95% of the devices from the same cluster. Additionally, 10% of each devices' neighbors lie outside the cluster. This ensures that the groups have sufficiently dense intra-device connections as well as having links with nodes outside the current group. This resulting network graph remains *reachable*. The experiments conducted during the course of this work encompass two different participation scenarios: (a) Upstream participation with probability p_k and (b) Neighborhood participation with probability d_k The participation probabilities of $p_k = \{0.9, 0.6\}$ and $d_k = \{0.9, 0.6\}$ have been used during the course of this work. The two

conditions have also been tested jointly to mimic scenarios of frequent stragglers and communication limitations.The channel conditions for both D2D and D2E communication have been subjected to zero-mean Gaussian noise $\mathcal{N}(\mu = 0, \sigma^2)$[2] with $\sigma^2 = 0.01, 0.0025, 0.0001$.

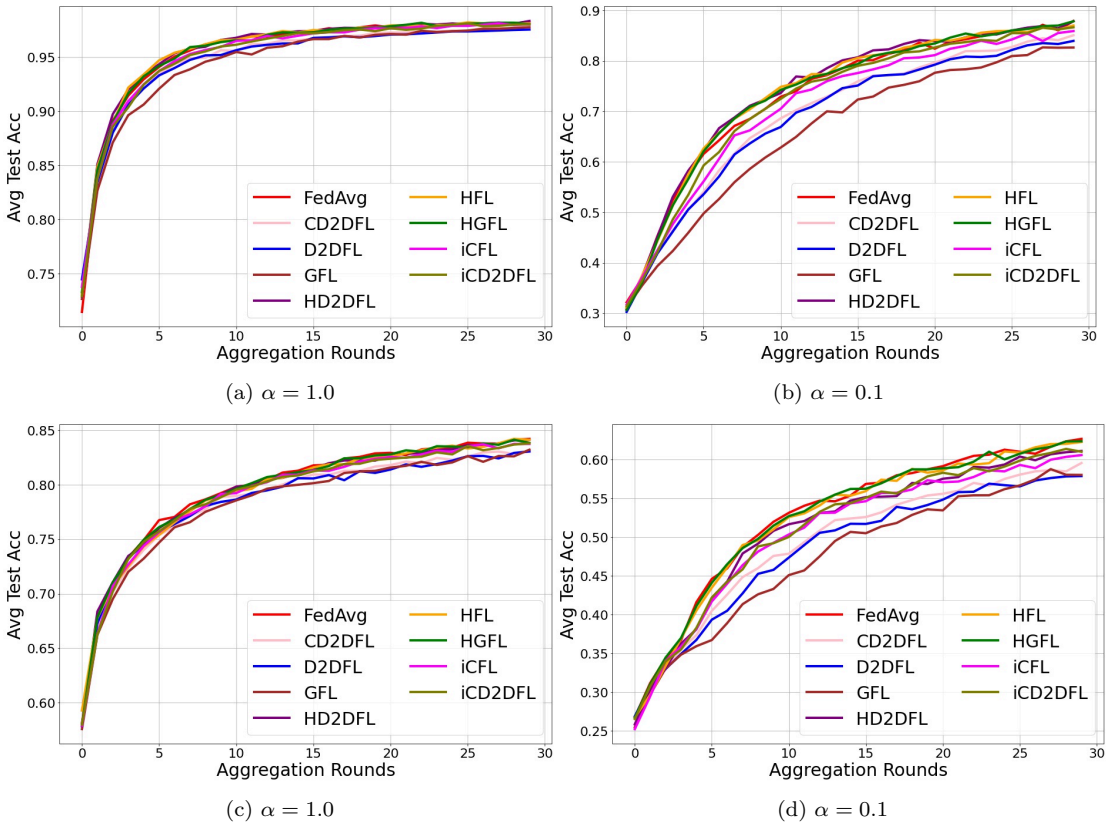
Few-Shot Federated Learning. One-Shot FL [34] suggests training local models to completion before sharing with the global server. The resulting mechanism enables significant reduction in communication frequency. Extending the idea, this work subjects the FL algorithms to Few-Shot FL. The nodes perform significantly more local updates before sharing their models. The scheme works with significantly reduced aggregation rounds and communication at both device and edge level. However, it also results in higher client drift.

Asynchronous Operation. FL envisions autonomy of operation at the device level. By extension, this implies that not all of the devices perform same number of update operations during the training phase. This may be caused by device being active (or inactive) or a straggler. This work assumes that the devices are required to work with a deadline after which they are required to share their respective models for aggregation. This deadline-based asynchronous aggregation behavior is replicated for the FL algorithms by allowing each device to undergo a range-limited random number of training epochs.

7. Performance Analysis

The following section encapsulates the findings for each of the mentioned simulation conditions. The final test accuracies of each algorithm, averaged over three runs and over all the nodes have been reported in Figures 5-9. Since, one of the aims of decentralized FL is to reduce the communication cost, we also present the number of messages for each link type for each algorithm in Fig. 10.

[2] $\boldsymbol{N}(0, \sigma^2)$ noise instead of more detailed fading models and frequency-selective channel impairments has been considered since the primary aim remains to gauge the impact of noisy updates being used for aggregation.



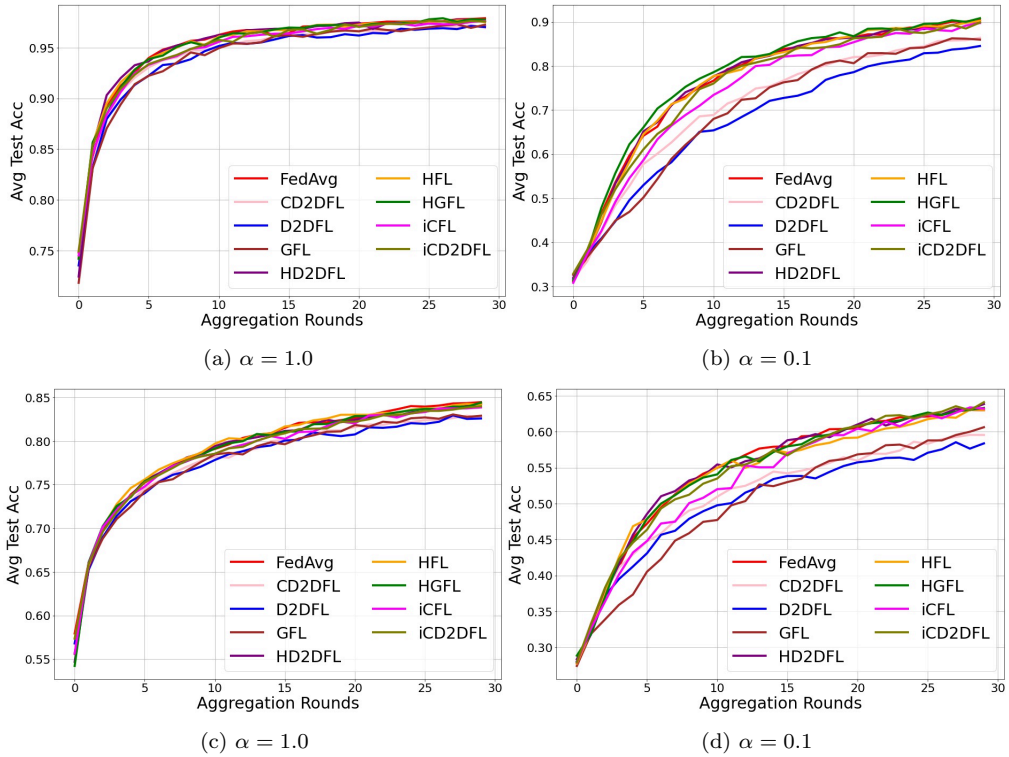
(a) $\alpha = 1.0$ (b) $\alpha = 0.1$

(c) $\alpha = 1.0$ (d) $\alpha = 0.1$

Figure 5: Average Test Accuracy for (a)-(b) MNIST and (c)-(d) FashionMNIST for maximal participation $p_k, d_k = 0.9$ with asynchronous communication for Non-IID distributions with Dirichlet parameter $\alpha = 10$ and $\alpha = 0.1$

7.1. FL with Maximal Participation and Ideal Communication

The ideal scenario assumes noise-free communication and a 90% device participation probability i.e. $p_k, d_k > 0.9$. As expected, greater non-IID distribution across device adversely affects the convergence rate and performance. The impact is reflected across all algorithms. However, results for FashionMNIST depict more loss in performance. This indicates a strong correlation between the performance degradation jointly due to a DNN's ability to learn and greater non-IID levels. The results in Fig. 5 indicate that decentralized FL performs on par with their more centralized counterparts. The results also show that as distributions become more non-IID, GFL, D2DFL and CD2DFL lag in learning performance when compared to the other algo-



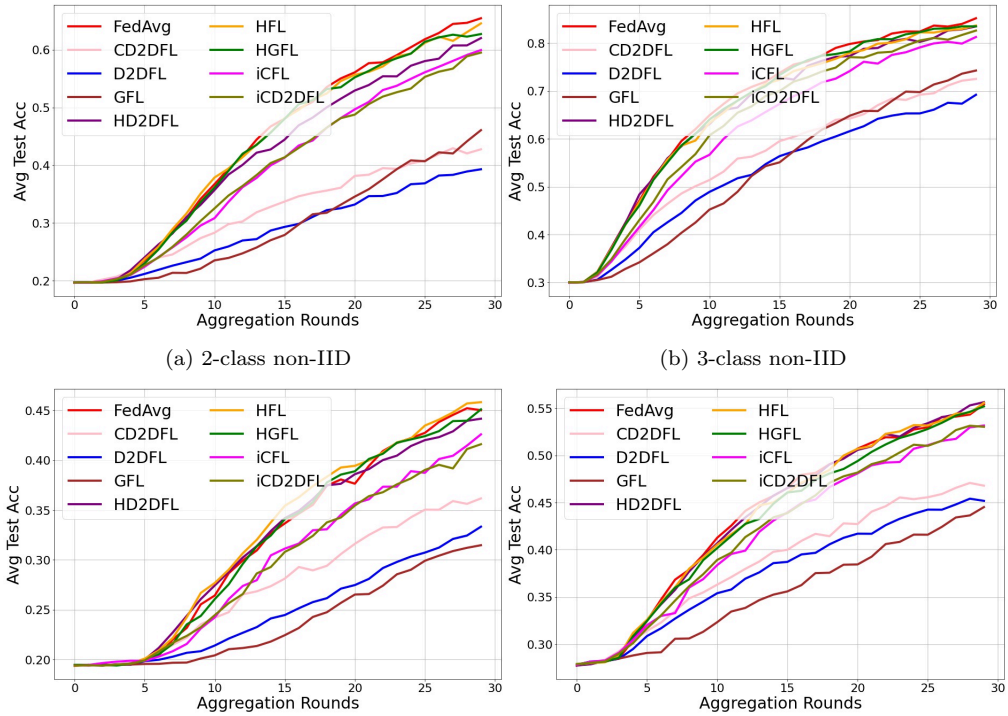
(a) $\alpha = 1.0$ (b) $\alpha = 0.1$

(c) $\alpha = 1.0$ (d) $\alpha = 0.1$

Figure 6: Average Test Accuracy for (a)-(b) MNIST and (c)-(d) FashionMNIST for limited participation $p_k, d_k = 0.6$ with asynchronous communication for Non-IID distributions with Dirichlet parameter $\alpha = 1.0$ and $\alpha = 0.1$

rithms. However, GFL also incurs the least communication volume among all the considered FL schemes since it only uses the least costly communication link as shown in Fig. 10.

The decentralized algorithms also indicate a slightly slower convergence rate exacerbated as the training progresses. However, clustered operations, particularly iCD2DFL and iCFL indicate better convergence and consistently perform better than D2DFL and GFL. This indicates that inter-cluster cooperation orchestrated by these algorithms helps in the learning performance and may help in arresting overtraining even in the absence of global information. This is particularly evident from iCFL where D2D interaction at the device level is absent.



(a) 2-class non-IID

(b) 3-class non-IID

(c) 2-class non-IID

(d) 3-class non-IID

Figure 7: Average Test Accuracy for (a)-(b) MNIST and (c)-(d) FashionMNIST for limited participation $p_k, d_k = 0.6$ with for extremely skewed (2-class and 3-class) Non-IID distributions.

7.2. FL with Limited Device Participation

The results in Fig. 6 present the results with reduced device participation. Less number of devices joining an aggregation round may be caused due to stragglers or busy devices. All the algorithms suffer from convergence issues as the device participation is decreased. This implies that data distribution and device participation have confounding effects on the learning performance of these algorithms. The difference in performance of the D2DFL, GFL and CD2DFL becomes more pronounced than the rest as the training proceeds. The performance suffers clearly from overtraining compounded by lower number of devices as well highly non-IID data. The convergence rate of these three algorithms slows more evidently than the others as the training proceeds. On the other hand, inter-cluster operations allow iCFL and iCD2DF to perform better than the others even with 60% device

participation.

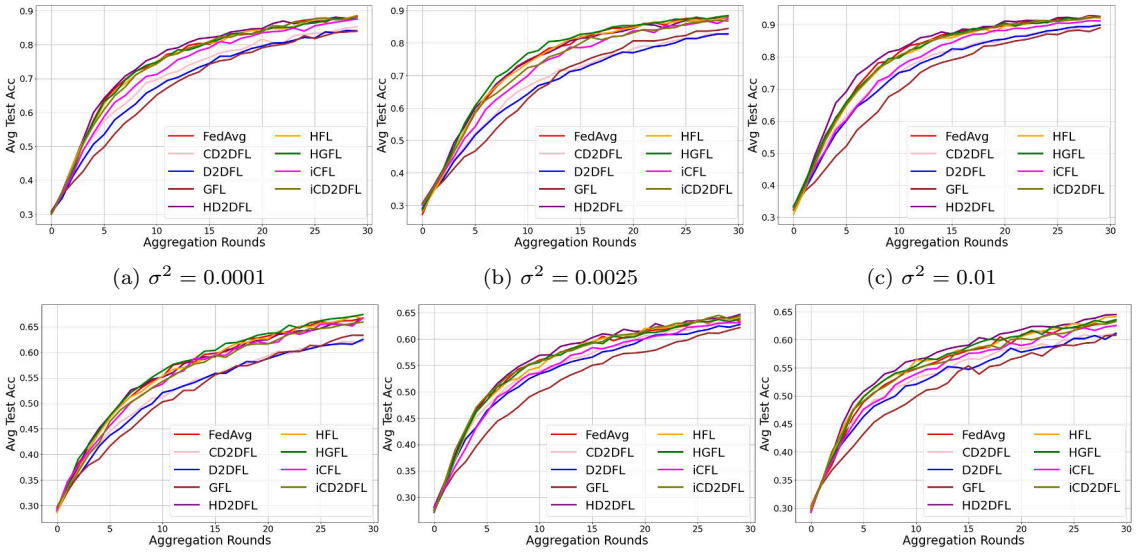
The results from Fig. 6 are reinforced by testing the FL algorithms under limited device participation and extremely skewed data distribution as shown in Fig. 7. When subjected to 2-class and 3-class non-IID data distributions, the localized operation in D2DFL, GFL and CD2DFL suffers more than the others. Lesser number of classes result in almost 20% performance difference between these algorithms and the rest as indicated in Fig. 7(a) and (c). The performance of iCFL and iCD2DFL, however, remains remarkably robust even in the presence of these extremely adverse conditions. Both algorithms, supported by inter-CH operation, perform within 5% of the centralized and hierarchical algorithms. The convergence rate of these algorithms retains its trajectory while other decentralized algorithms show greater divergence as the training proceeds. It may therefore be inferred that gossip operations by the CHs partially offset the degradation caused by the lesser device participation and extremely non-IID data and acts akin to DropOut operations in a DNN. Table. 3 provides an accuracy-based performance overview of all the algorithms both with maximal and limited participation conditions.

Algorithm	$p_k = 0.6, d_k = 0.6$		$p_k = 0.9, d_k = 0.9$	
	$\alpha = 1.0$	$\alpha = 0.1$	$\alpha = 1.0$	$\alpha = 0.1$
FedAvg	0.9794	0.8736	0.9786	0.8810
D2DFL	0.9758	0.8454	0.9752	0.8849
HFL	0.9805*	0.9022*	0.9808	0.9257*
HD2DFL	0.9798*	0.8980	0.9831*	0.9189
GFL	0.8744	0.3824	0.9753	0.8398
HGFL	0.9801*	0.9015*	0.9800	0.9241
iCFL	0.9762	0.8972	0.9791	0.9142
CD2DFL	0.9694	0.8651	0.9730	0.8855
iCD2DFL	0.9803*	0.8983	0.9800	0.9162

Table 3: Accuracy comparison of FL algorithms for two sets of p_k, d_k values and different values of the Dirichlet parameter α (lower values indicate increased non-IID distribution). Highest accuracies within $\Delta = 0.001$ have been marked with asterisk (*)

7.3. FL with Noisy Communication

The largest application of Federated Learning is envisioned in wireless spectrum. However, wireless media is subject, among other challenges, to



(a) $\sigma^2 = 0.0001$ (b) $\sigma^2 = 0.0025$ (c) $\sigma^2 = 0.01$

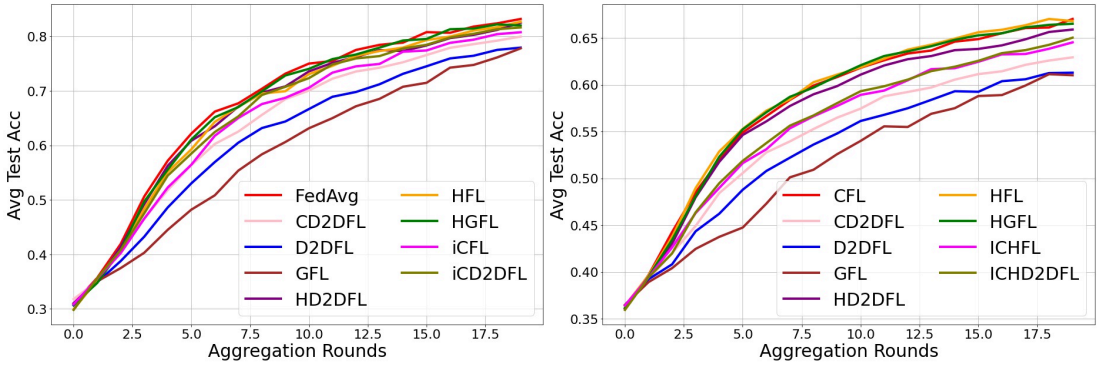
(d) $\sigma^2 = 0.0001$ (e) $\sigma^2 = 0.0025$ (f) $\sigma^2 = 0.01$

Figure 8: Average Test Accuracy for (a)-(c) MNIST and (d)-(f) FashionMNIST for $\boldsymbol{N}(0, \sigma^2)$ noisy communication with $p_k, d_k = 0.9$ and Dirichlet parameter $\alpha = 0.1$.

a significant presence of noise. With FL requiring to exchange hundreds of thousands of parameters, presence of noise may cause the convergence to slow down considerably. The FL algorithms during the current research were subjected to the presence of Gaussian Noise $\mathcal{N}(0, \sigma^2)$ both at the device communication level and during D2E and E2C communication. From Fig. 8 it can be observed that the addition of noise at the aggregation stage mildly slows down convergence. HGFL progressively shows reduction in performance as σ^2 is increased. However, clustering operation in iCFL and iCD2DFL allow them to closely follow the performance by HD2DFL and HFL. In contrast, GFL shows reduced convergence as compared to D2DFL. However, as training progresses, D2DFL and CD2DFL both also indicates a plateauing convergence rate.

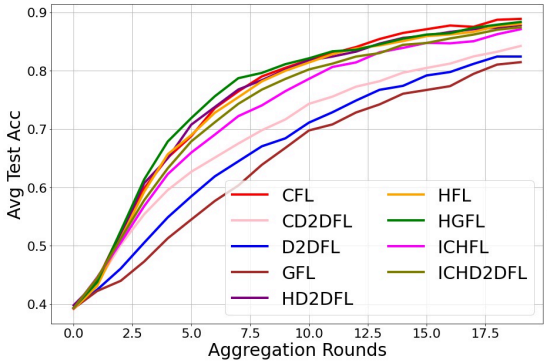
7.4. Few Shot Learning

Few-Shot FL offers one way of reducing the communication cost when applied properly. The devices undergo multiple local updates before sharing their models for aggregation. Fig. 9 shows the result of Few-Shot learning applied under maximal and limited device participation scenario with

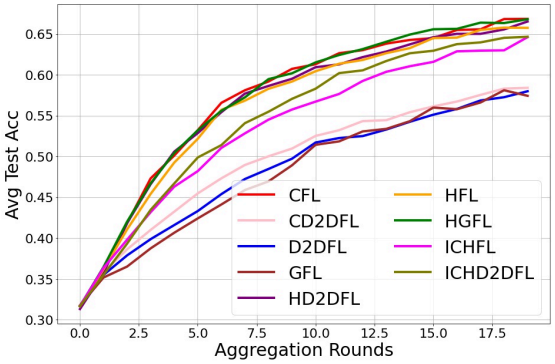


(a) $\alpha = 0.1$ MNIST

(b) $\alpha = 0.1$ FashionMNIST



(c) $\alpha = 0.1$ MNSIT



(d) $\alpha = 0.1$ FashionMNIST

Figure 9: Average Test Accuracy for Few-Shot Learning for MNIST and FashionMNIST with (a)-(b) $p_k, d_k = 0.9$ and (c)-(d) $p_k, d_k = 0.6$ with $r = 20$ aggregation rounds and epochs in range $[15, 20]$ with asynchronous communication for Non-IID distributions with Dirichlet parameter $\alpha = 0.1$

$p_k, d_k = 0.9$ in Fig. 9(a)-(b) and $p_k, d_k = 0.6$ in Fig. 9 (c)-(d). The additional difference in performance can be attributed to greater non-IIDness in figures (c) and (d). The impact of client drift on all algorithms causing the learning to plateau earlier than regular learning regime is evident when its results are compared with the ones shown in Fig. 5-6. Overall, the decentralized variants are impacted worse than the centralized and hierarchical algorithms. Additionally, the performance of D2DFL, GFL and CD2DFL lags further than the rest as training progresses. The performance of iCFL and CD2DFL while showing relatively bigger drop than when subjected to normal training scheme, still achieves an accuracy within 5% of the centralized algorithms. The overall performance suggests scenarios where Few-Shot

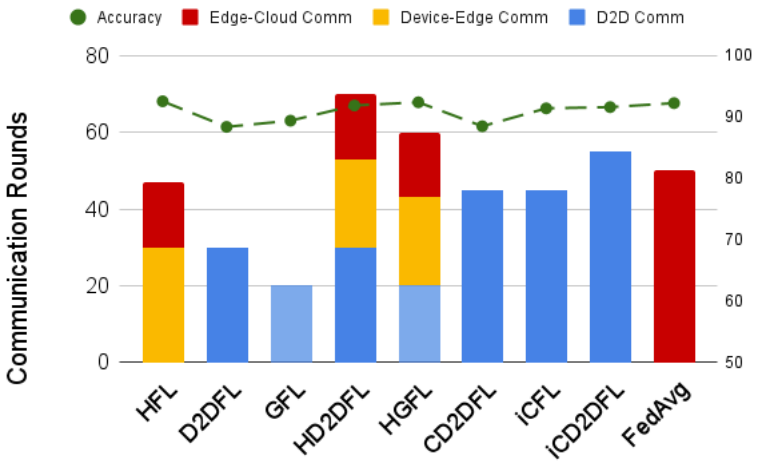


Figure 10: Average FL Algorithm Accuracy and per-Node Communication Volume for asynchronous aggregation for 30 rounds with the MNIST dataset.

learning may be considered practicable when communication costs become prohibitive especially or shared updates are infrequent.

7.5. Communication Cost and Volume

When dealing with different FL algorithms, it is essential that the difference between links employed during the operation is kept in purview. The work in [35] suggest that D2D links like LTE-Direct, WiFi Direct or DSRC are more energy efficient than Device-Edge cellular links like LTE. Furthermore, the same research also indicates that the mentioned D2D links offer reduced latency when compared with cellular communication. E2C links must be be preceded by D2E links for any communication meant for the cloud. It may therefore be inferred that the energy cost and latency of uploading to the cloud may well be more than both D2D and D2E communications. It is with this understanding that Fig.10 considers D2D links as least costly and E2C links as most expensive both in terms of energy and latency.

Fig-10 shows the communication volume required by different FL algorithms. This is calculated by identifying the number of aggregation rounds required by each of the FL algorithms and are categorized according to the target level (device, edge or cloud). The centralized variants employ more expensive cellular communication for D2E communication. The additional load on network backbone in form of E2C links is bound to significantly increase

the cost for FL albeit its improved performance. Fig.10 shows the communication volume in terms of communication rounds for various FL algorithms with the accuracy achieved by the respective algorithms under non-IID data settings. The highest performing algorithms use the most expensive forms of communication. On the contrary, decentralized FL variants indicate slower convergence but also incur less communication cost.

Comparably, decentralized scheme with inter-cluster aggregations show appreciable performance, falling slightly short of the accuracy of HGFL and HD2DFL.

8. Conclusion

This work presents a comprehensive comparison of various FL algorithms that have so far been viewed only from the perspective of Centralized FL. The FLAGS framework developed for this purpose enables configuring multiple FL architectures expeditiously. The contrasting highlight of this work is a detailed comparison of major FL algorithms under some of the most dominant challenges in this domain. frameworks. The analysis conducted here indicates that decentralized FL performs comparatively well despite no or minimal upstream communication. Even though noisy communication and irregular participation negatively impacts decentralized FL performance, such modes are still capable of achieving acceptable performance thresholds. However, extremely skewed data distributions degrade fully decentralized FL considerably more than centralized ones. The results indicate that FL modes may be used interchangeably depending on the network conditions and the communication costs. As part of the future work, we also intend to investigate the performance of the FL algorithms for feature skewed non-IID distributions. Furthermore, based on the results presented here, the next milestone would be to formulate an adaptive mechanism based on the operating and environment characteristics where the nodes may be suggested to follow a particular FL algorithm for efficient operation.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y. Arcas, Communication-efficient learning of deep networks from decentralized dataArXiv: 1602.05629 (Feb 2017).

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, arXiv:1912.04977 (2019).

[3] R. I. Ansari, C. Chrysostomou, S. A. Hassan, M. Guizani, S. Mumtaz, J. Rodriguez, J. J. Rodrigues, 5g d2d networks: Techniques, challenges, and future prospects, IEEE Systems Journal 12 (4) (2017) 3970–3984.

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

[5] I. Hegedüs, G. Danner, M. Jelasity, Gossip Learning as a Decentralized Alternative to Federated Learning, Vol. 11534 of Lecture Notes in Computer Science, Springer International Publishing, 2019, p. 74–90. doi:10.1007/978-3-030-22496-7_5.

[6] H. Xing, O. Simeone, S. Bi, Federated learning over wireless device-to-device networks: Algorithms and convergence analysis, arXiv:2101.12704 (Jan 2021).

[7] S. S. Ram, A. Nedić, V. V. Veeravalli, Distributed stochastic subgradient projection algorithms for convex optimization, Journal of optimization theory and applications 147 (3) (2010) 516–545.

[8] S. Savazzi, M. Nicoli, V. Rampa, S. Kianoush, Federated learning with mutually cooperating devices: A consensus approach towards server-less model optimization, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, p. 3937–3941. doi:10.1109/ICASSP40776.2020.9054055.

[9] S. Savazzi, M. Nicoli, V. Rampa, Federated learning with cooperating devices: A consensus approach for massive iot networks, IEEE Internet of Things Journal 7 (5) (2020) 4641–4654. doi:10.1109/JIOT.2020.2964162.

[10] C. Li, G. Li, P. K. Varshney, Decentralized federated learning via mutual knowledge transfer, IEEE Internet of Things Journal 9 (2) (2022) 1136–1147. doi:10.1109/JIOT.2021.3078543.

[11] J. Wang, S. Wang, R.-R. Chen, M. Ji, Local averaging helps: Hierarchical federated learning and convergence analysis, arXiv:2010.12998 (Mar 2021).

[12] M. S. H. Abad, E. Ozfatura, D. GUndUz, O. Ercetin, Hierarchical federated learning across heterogeneous cellular networks, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, p. 8866–8870. doi:10.1109/ICASSP40776.2020.9054634.

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

[14] F. P.-C. Lin, S. Hosseinalipour, S. S. Azam, C. G. Brinton, N. Michelusi, Semi-decentralized federated learning with cooperative d2d local model aggregations, IEEE Journal on Selected Areas in Communications (2021) 1–1doi:10.1109/JSAC.2021.3118344.

[15] A. Hashemi, A. Acharya, R. Das, H. Vikalo, S. Sanghavi, I. Dhillon, On the benefits of multiple gossip steps in communication-constrained decentralized federated learning, IEEE Transactions on Parallel and Distributed Systems 33 (11) (2021) 2727–2739.

[16] T. Wang, Y. Liu, X. Zheng, H.-N. Dai, W. Jia, M. Xie, Edge-based communication optimization for distributed federated learning, IEEE Transactions on Network Science and Engineering (2021).

[17] J. S. Ng, W. Y. B. Lim, Z. Xiong, X. Cao, J. Jin, D. Niyato, C. Leung, C. Miao, Reputation-aware hedonic coalition formation for efficient serverless hierarchical federated learning, IEEE Transactions on Parallel and Distributed Systems 33 (11) (2021) 2675–2686.

[18] A. Singh, P. Vepakomma, O. Gupta, R. Raskar, Detailed comparison of communication efficiency of split learning and federated learning, arXiv preprint arXiv:1909.09145 (2019).

[19] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, M. Jirstrand, A performance evaluation of federated learning algorithms, in: Proceedings of the second workshop on distributed infrastructures for deep learning, 2018, pp. 1–8.

[20] I. Hegedüs, G. Danner, M. Jelasity, Decentralized learning works: An empirical comparison of gossip learning and federated learning, Journal of Parallel and Distributed Computing 148 (2021) 109–124.

[21] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, P. P. de Gusmão, N. D. Lane, Flower: A friendly federated learning research framework, arXiv:2007.14390 (2020).

[22] TensorFlow Federated — tensorflow.org, `https://www.tensorflow.org/federated`.

[23] F. Lai, Y. Dai, X. Zhu, H. V. Madhyastha, M. Chowdhury, Fedscale: Benchmarking model and system performance of federated learning, in: Proceedings of the First Workshop on Systems Challenges in Reliable and Secure Federated Learning, 2021, pp. 1–3.

[24] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, A. Talwalkar, Leaf: A benchmark for federated settings, arXiv:1812.01097 (2018).

[25] Q. Li, Y. Diao, Q. Chen, B. He, Federated learning on non-iid data silos: An experimental study, arXiv preprint arXiv:2102.02079 (2021).

[26] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, et al., Fedml: A research library and benchmark for federated machine learning, arXiv:2007.13518 (2020).

[27] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, arXiv:1610.05492 (2016).

[28] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, X. Chen, Convergence of edge computing and deep learning: A comprehensive survey, IEEE Communications Surveys Tutorials 22 (2) (2020) 869–904. `doi:10.1109/COMST.2020.2970550`.

[29] M. N. Tehrani, M. Uysal, H. Yanikomeroglu, Device-to-device communication in 5g cellular networks: challenges, solutions, and future directions, IEEE Communications Magazine 52 (5) (2014) 86–92.

[30] J. Daily, A. Vishnu, C. Siegel, T. Warfel, V. Amatya, Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent, arXiv:1803.05880 (2018).

[31] L. Deng, The mnist database of handwritten digit images for machine learning research, IEEE Signal Processing Magazine 29 (6) (2012) 141–142.

[32] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv:1708.07747 (2017).

[33] Q. Li, B. He, D. Song, Model-contrastive federated learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 10713–10722.

[34] N. Guha, A. Talwalkar, V. Smith, One-shot federated learning, arXiv:1902.11175 (2019).

[35] M. Condoluci, L. Militano, A. Orsino, J. Alonso-Zarate, G. Araniti, Lte-direct vs. wifi-direct for machine-type communications over lte-a systems, in: 2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015, pp. 2298–2302. doi:10.1109/PIMRC.2015.7343681.