HDFL: A Heterogeneity and Client Dropout-Aware Federated Learning Framework

Syed Zawad
IBM Research - Almaden
San Jose, California, USA
szawad@ibm.com

Ali Anwar
University of Minnesota
Minneapolis, Minnesota, USA
aanwar@umn.edu

Yi Zhou
IBM Research - Almaden
San Jose, California, USA
yi.zhou@ibm.com

Nathalie Baracaldo
IBM Research - Almaden
San Jose, California, USA
baracald@us.ibm.com

Feng Yan
University of Houston
Houston, Texas, USA
fyan5@central.uh.edu

***Abstract*—Cross-device Federated Learning (FL) enables training machine learning (ML) models on private data that is heterogeneously distributed over many IoT end devices without violating privacy requirements. Clients typically vary significantly in data quality, hardware resources and stability, which results in challenges such as increased training times, higher resource costs, sub-par model performance and biased training. Existing works tend to address each of these challenges in isolation, but overlook how they might impact each other holistically. We perform a first of its kind characterization study that empirically demonstrates how these properties interact with each other to impact important performance metrics such as model error, fairness, resource cost and training time. We then propose a method called HDFL based on our observations, which is the first framework to our knowledge that comprehensively considers the multiple aforementioned important challenges of practical FL systems. We implement HDFL on a real distributed system and evaluate it on multiple benchmark datasets which show that HDFL achieves better Pareto frontier compared to both the state-of-the-practice and state-of-the-art systems with up to 4-10% better model accuracy, 33% improved good-intent fairness, 63% lower cost, and 17% faster training time.**

***Index Terms*—federated learning, privacy, deep learning, fairness**

I. Introduction

The prevalence of mobile and internet-of-things (IoT) devices in recent years has led to massive amount of data being generated every day due to usage of user-end applications. Such data can be potentially used to train state-of-the-art machine learning models to address a wide variety of ML tasks such as keyword spotting [1], image captioning [2], translation [3] and medical applications [4]. Traditional methods of training expensive deep learning models involve the transfer of user data from their low-powered IoT devices to the high powered computation clusters of the third-parties who wish to train the model. However, such data are usually user-owned and may contain private data which they may not be willing to share with third-parties. Additionally, regulations such as HIPAA [5], [6] and GDPR [7], [8] limit the access and transmission of personal data such as healthcare and financial information in consideration of security and privacy. Such large amounts of data are, however, required to train good, generalizable models which might otherwise not be possible using controlled datasets since they tend to be higher in number and more realistic [9]. Therefore, this leads us a demand for using user-generated data for ML training but at the same time ensuring privacy of the users.

Federated Learning (FL) [10], [11] has been proposed as distributed training method for deep learning models without compromising a user's privacy. Unlike traditional distributed training where the data is transferred from the user client devices to the centralized server, the main idea here is to transfer the user-trained models instead. The FL architecture consists of a third-party server which contains the *global* model that needs to be trained, and all the IoT clients on whose data we need to train this *global* model on. This *global* model is initially untrained, and sent to selected IoT clients. The models are trained individually on each device using their local data only and using the computational resources of the IoT hardware. This results in separate local models for each IoT which are then sent to the third-party server. The models have encryption or noise (or both) applied as added security and privacy mechanisms when being sent.

Once the individual models arrive at the server, they are consolidated into a new single *global* model. This step (from sending the *global* model till aggregation) is called a training *round*. Training proceeds across multiple rounds and in each round a subset of IoT clients are randomly selected from all available clients. Eventually, the *global* model converges as the final model. Thus, FL training is considered a privacy preserving method since data stays localized and is not accessible by third-parties. This *cross-device* FL training method is known to be highly effective in training deep learning models.

As discussed above, in conventional ML training the data is typically owned by a single party and maintained in a centralized location. Therefore, both data and computation can be controllably distributed over a cluster of computing nodes. This results in more predictability for resource consumption

and the quality of the model. However, since data in FL is generated and owned by clients and the privacy requirements prevent accessing or moving the personal data during training, control of resources and data is not possible. This leads to a few challenges unique to FL.

One of the most prominent of these problems is called the *data heterogeneity* issue. Since usage behavior and environments differs between users, the data being generated among them can vary widely. Features of the dataset (also called *data quality*) such as class distribution can significantly vary, as well as the number of total datapoints generated (called *data quantity*) for training. This can result in biased local model training and an overall sub-par models after aggregation [12]–[14]. Bias in FL models is generally measured by the *variance of accuracies* of the global model after being evaluated on the data of individual client's test datasets (termed *good-intent fairness* [15]). The quantity of datapoints per device also varies widely, resulting in different amounts of *training cost* incurred by each device. Since we are blind to the data distribution and have no direct control over it, one of the main challenges in FL is to find indirect methods of adjusting the training process in order to mitigate these issues.

Another challenge is that the local training times vary greatly across clients depending on their hardware resources (known as *resource heterogeneity*) which can result in stragglers and thus longer overall training time due to straggling clients. We have little to no direct control over this as well. Furthermore, mobile and IoT devices are not dedicated to the training tasks. Only when clients meet certain criteria for device properties such as battery status, idle time, training time, and network status [16], can they participate in training. This state can change in the middle of training as well, resulting in the phenomenon of *client dropouts*, where clients stop participating in the middle of rounds, causing synchronization delays and biased training [17], [18].

Data heterogeneity, resource heterogeneity, and client dropouts are all important characteristics of FL and can heavily impact model error, fairness, cost, and training time of a model. However, previous works [16], [19]–[22] consider them separately, resulting in sub-optimal models and long training time. To fill this gap, we propose *HDFL*, a holistic approach that considers the impact of data heterogeneity, resource heterogeneity, and client dropout on model error, fairness, cost, and training time when selecting clients to query. We start by performing a first-of-a-kind characterization study on the interactions between the data, resource heterogeneity and dropouts. We first define a set of metrics to quantify the four most important qualities of a FL system as focused on in current literature, namely the final model performance, final model fairness, total training time and total resource consumption. We then control the data heterogeneity, dropouts and resource properties of a standard FL system and observe how they interact among them to influence the metrics.

We observe that there exists trade-offs between the various metrics depending on the system environment. Namely, we observe that 1) attempting to reduce training time by reducing slow client selection results in worse final model performance and model biasness, 2) client dropouts reduce final model performance, waste resources and results in worse training fairness, and 3) reducing training cost also reduces model performance. Works in current literature only tend to focus on one aspect or other such as reducing training time or increasing model accuracy but ignore their tradeoffs such as less performance or more incurred resource costs.

Based on the our observations, we propose *HDFL*, which balances these various metrics to reduce the tradeoffs as much as possible for a given amount of resource heterogeneity, data heterogeneity and dropouts in a certain FL system. By quantifying the impact of these factors on model convergence, resource cost and training time, *HDFL* makes judicious client selection decisions to achieve the most profitable training results. *HDFL* formulates the problem as a multi-objective optimization and considers two key properties when making a scheduling decision: *selection probability* and *selection mutualism*. Both properties are derived by taking into consideration data heterogeneity, resource heterogeneity, and client dropouts. Selection probability describes how often a client should be selected and its quantification is empowered by a training efficiency assessment approach that employs Underestimation Index (UEI) [23] as a unified measure to represent model error, fairness, and cost while preserving the privacy requirements. Selection mutualism captures the mutualism among clients in terms of training time in a specific training round and aims at minimizing the straggler and client dropout effects to improve the overall training time.

We implement *HDFL* on a real distributed cluster where clients and the aggregation server are deployed on their own individual hardware. We evaluate our system using three standard FL benchmarks (FEMNIST, Cifar10, and Shakespeare). We compare *HDFL* with the state-of-the-art large-scale FL systems [16] and the widely used bare-bone FL system [11]. We show that *HDFL* is capable of automatically deriving the best possible tradeoff between the various metrics as compared against the other state-of-the-art solutions which only tend to focus on one aspect or another. We show this by demonstrating that *HDFL* achieves better Pareto frontier across all metrics and requires little to no tuning. Our holistic approach is empirically shown to more balanced, and we outperform both the state-of-the-practice and state-of-the-art systems with up to 4-10% better model accuracy, 33% improved good-intent fairness, 63% lower cost, and 17% faster training time.

II. Related Works

For this section, we discuss the state of current literature for cross-device federated learning as they focus on final model performance, fairness, resource cost and time reduction.

A. Model Performance in FL.

In addition to current challenges in ML model training, FL faces extra hurdles due to data heterogeneity. As pointed out in [24], the imbalanced distribution of data among the clients means imbalanced local training which eventually results in

worse overall model performance compared to having the same dataset in a traditional distributed setting. There have been much focus on addressing this issue specifically since it is a fundamental yet unique property of FL. The first line of works involves works like FedProx [17], FedCL [25], FedCurv [26] and MOCHA [27] which change the fundamental aggregation formula. The baseline FL [11] uses Federated Averaging, which simply performs weighted averaging of the model parameters based on the number of datapoints the local models were trained on. The newer works change this aggregation algorithm by adding more complex methods such as better local minimizers [12] and better loss functions and optimizers [28]. Another line of works involves changing the selection mechanisms to choose the best subset of clients that yield good model performance [19], [29], [30]. Finally, some other works focus on changing the local training process by tuning the training hyperparameters such as learning rate, local epochs and batch size to ensure that local models do not overfit on biased data [31], [32]. However, none of these works discuss the impact of their techniques on the other aspects of FL such as resource cost, consumption and training time, or consider the impact of dropouts.

B. Fairness in ML.

Fairness of models is an extensively explored concept in traditional ML [33]–[38] and many works have defined their own notions of "fairness". For example, [39] introduces *counterfactual fairness* where a decision is considered fair towards an individual if the decision taken by a model would be the same if that individual belonged to a different sample group. This topic was recently explored in FL by [40]. [41] talks about *classification fairness* which measures how much a model is biased during inference towards or against a particular target class. [34] proposes a criterion for discrimination against sensitive attributes for protected classes in general supervised learning. [42] extensively discusses current fairness issues in ML. While these approaches focus on mitigating bias for unprivileged groups, e.g., race or gender, our fairness definition does not consider such protected attributes.

C. Fairness in Federated Learning.

Good-intent fairness was defined as the variance of client test accuracies of a model in [15]. If a model performs well on one client's dataset and bad on another, it indicates that the model is biased against the features of the worse-performing client and therefore is not fair. In this paper we use this fairness definition. [15] also propose a minimax optimization framework called Agnostic Federated Learning (AFL) to reduce overfitting on local client data by optimizing with learning bounds on the clients with the highest losses. However, AFL does not consider resource usage or the biased participation of clients which are important practical concerns in FL. Works like [43] talks about utilizing resources fairly, but does not take resource heterogeneity or data heterogeneity into account. [44] proposes q-FFL, which is a method to reduce biasness in the global model by making the client accuracies more uniform (i.e., increasing *good-intent fairness*). They do this by assigning more weights to the client updates with higher empirical loss values, thereby ensuring that the worst client updates can still contribute enough to the global model and get a more uniform testing accuracy across clients. For our paper, we use the same definition of fairness and our objective is the same. However, instead of focusing on the aggregation algorithm, we focus on the client dropouts phenomenon of FL (i.e., how to be fair to clients if they do not consistently contribute to the FL training process). This paper works under the same assumptions as [15] in that they assume equal participation of all clients. [45], [46] talks about fairness not in terms of *good-intent fairness* but how much value a client gets from participation. Costs are considered in terms of monetary compensation, which is orthogonal to our paper. We focus on the cost in terms of resource efficiency (total samples used in training) instead.

D. Resource Reduction in Federated Learning.

Most works in FL focus on communication and energy efficiency [47]–[50], but few have explored policy-driven schedulers. [49] theoretically analyzes the trade-off between local update and global parameter aggregation to minimize the loss function under a given resource budget. [47] uses reinforcement learning for optimizing caching, local computation and communication efficiency. [21] selects clients every round such that they can complete training within a given time limit, thereby controlling the amount of resources consumed per round. [51] focuses on reducing model size using compression methods and update frequencies resulting in less resources used overall. [52] proposed a novel aggregation and global model distribution scheme that reduces time to converge and reduces communication cost early in the training process. [53] introduces *FedPAQ* with the aim to reduce communication overhead of too many devices trying to communicate with the central server at the same time. [16] proposes a comprehensive system to enable large-scale distributed FL frameworks. [54] focuses on scaling up wireless communication systems for edge devices.

E. Training Time Reduction in FL

There have also been extensive work towards reducing training time of FL systems. Works such as TiFL [19] and HFL [55] attempt to mitigate stragglers by grouping slower clients together so that no single client slows down the time per round too much. Some frameworks such as [16], [56] directly remove slower clients from the training process entirely, while others [19], [57] reduce their probability of being chosen. Frameworks like [17], [58], [59] perform partial local training such that the overall training time per device remains same across all clients regardless of their hardware and number of datapoints. However, such drastic changes in the overall architectures impacts the training process and thus influences the output of the final model performance, which none of the works here consider. In contrast, our *HDFL* takes a more

holistic approach and tries to reduce time with minimal impact to model performance.

III. Characterization Study

In this section, we systematically characterize multiple variability points of FL, which demonstrates the importance of a holistic approach.

A. Performance Metrics

The goal for the *model owners* in FL is to train a highly accurate and generalizable ML models using other clients' private data that would otherwise be unavailable. On the other hand, the incentive for the *data owners* in FL is to get better services from model owners by contributing their data to training under privacy protection. They usually prefer good user experience (i.e., with as less cost as possible) and fair reward (i.e., the trained model performs well on their data). Thus, we identify four important performance metrics when evaluating FL and define them as below.

1) Model Error: is defined as the test accuracy error on all datasets, i.e., $mean$ error of the global model on each of the client's sampled test data denoted as $1 - \frac{\sum_{i=0}^{n} A^i}{n}$, where A^i is the accuracy of global model on test data of i and n is the total number of clients. One point to note here is that in FL, the test data is derived from the clients too. Generally, it is set to randomly sampled 10% of the total datapoints per client [60]. Therefore, we use the same setup for our case as well. The reason for doing so is that in a real-world system, the trained model must perform well on actual data. Since actual data here is derived from the IoT clients, we test the global model on the client's data as is standard practice. Before training, each client puts aside this test set from its local dataset and is never used for training.

2) Training time: is defined as the wall-clock time of training. We choose wall-clock time instead of training rounds, as the round time can differ significantly due to data and resource heterogeneity. In state-of-the-art paper such as [19], it is measured as total time taken beginning from when the first round's clients are selected till the end of the last round. This does not include the time taken for inference during testing. We take the same approach and measure the time in seconds.

3) Cost: is used to quantify the total amount of resources that are used by the IoT devices for local training. The aggregation expense on the server side is negligible compared to the forward and backward propagation cost during local training [16] since the aggregation is a simple average of the model weights. This cost can be in terms of computation power, energy or memory with the units FLOPs, Joules and bytes respectively. It can also be a combination of all of them together. Different costs are used for different papers depending on what their focus is. For our case, instead of a single resource type, we focus on resources in general.

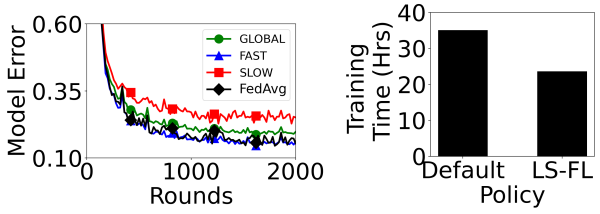
We know from literature that one of the most significant indicators of resources consumed are the number of data samples that have been used for training [17], [42], [48]. This is because given the same model and deployment environment, the number of datapoints is directly proportional to the total amount of computation. As such, the number of datapoints serves as a generalizable and abstract level indicator of the total resources consumed since we can derive the approximate of the actual cost after some profiling. Note that even if a client drops out during training, the used data samples also count into the cost. We use training samples instead of resource hours as the resource in FL is highly heterogenous across clients. It is worth noting that more sophisticated cost metrics can also be used such as carbon footprint, executed floating point operations, which we defer to our future work.

4) Fairness: is defined as *good-intent fairness* [15] that measures the accuracy variance when the global model is evaluated using test datasets of individual clients: $\sqrt{\frac{\sum_{i=0}^{n}(A^i - \bar{A})^2}{n-1}}$, where A^i is the accuracy of global model on test data of i, n is the total clients and $\bar{A}$ is the mean accuracy. This is tied to the way that model performance is measured in FL as discussed in *Model Error*. By testing the global model on a client's test dataset, we effectively measure how well the FL training was. The idea is that if the features of a specific client is learnt well by the global model, the test accuracy should be high on its local test dataset.

Ideally, we want our global model to perform well on every device. However, in practice, we know that the differences in test accuracies among the clients' local test datasets can vary widely. This difference in performance indicates that some clients may not have contributed to the learning process at all even though they have spent resources for local training. This is therefore called a $fairness$ issue since the model is biased towards some clients' features over others, which indicates a non-generalizable model and is therefore undesirable. We choose this good-intent fairness definition since it is the standard method of reflecting the bias issues among clients in FL [15]. The lower the $Fairness$ value, the more fair the model is.

These metrics quantify the different performance aspects of FL systems. To understand how these metrics influence each other, next we perform a set of characterization studies.

B. Tradeoff Between Fairness and Training Time



(a) Model Error over Rounds (b) Training Time

Fig. 1: Tradeoff between fairness and training time. (a) shows the test error comparison across rounds. LS-FL results in higher model error for the slower clients (SLOW), leading to worse fairness than FedAvg. (b) shows the training time comparison, where LS-FL outperforms FedAvg.

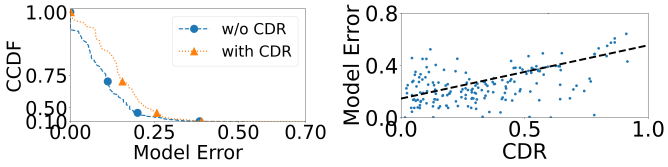
One of the focuses of state-of-the-art large-scale FL systems such as [16] is on improving the training time. Due to the highly heterogeneous nature of clients, the *training latency* (defined as client's local training time) varies greatly across clients. Given the training time of each round is bounded by the slowest client (i.e., straggler), the straggler effects significantly impact the overall training time. To address this, the widely popular [16] (which we name *LS-FL* for convenience) suggests selecting 33% more clients but only use the weights from the first 75% for training the global model and discard the weights of the slowest 25%. Similar methods are applied in [21], [61], albeit with variations to the default approach. This approach helps reduce the straggler impact, but it results in biased training as slower clients are have less opportunities to contribute in training in addition to wasting their client-side resources.

To demonstrate the effects of this policy empirically, we set up a real distributed FL system. We conduct our experiments using the FEMNIST image classification dataset, model, and training hyperparameters from the LEAF [60] framework. We use a total of 53,000 images for training across 200 clients and select 10 clients uniform randomly per round during the training. The dataset of FEMNIST comes pre-sharded such that it follows a real world data heterogeneity. Specifically, each client is assigned only the images hand-written by the same person, thus making each client have unique calligraphic features in their local datasets. We manually assign a training latency per client using a normal distribution with a mean of 5 seconds and a deviation of 1.5 seconds. We train this setup until convergence and plot 4 different test accuracies over rounds in Figure 1a that demonstrates the impact of dropouts. For testing, the global model is sent to all participating clients. Each client has its own reserved test dataset made by sampling 10% of its total available data on which the global model is tested on, and the results reported back to the server.

Figure 1a demonstrates the tradeoff between fairness and training time (experimental setup is detailed in Evaluation Section). The *GLOBAL*, *FAST*, *SLOW* represents the mean error of all clients, the fastest 75% clients, and the slowest 25% clients, respectively, when *LS-FL* is used. *FedAvg* represents the mean error of all clients when state-of-the-practice system *Federated Averaging* is used, where no client update is dropped. We observe about 15% difference in test error between the fastest 75% and slowest 25% clients in *LS-FL*, indicating a significant difference in model accuracy between faster and slower clients, leading to poor *fairness*. We also observe an overall increased error for *GLOBAL* when compared to *FedAvg* by around 4% at convergence as well. However, Figure 1b shows a significant improvement in training time when using *LS-FL* compared to *FedAvg*, clearly revealing the tradeoff between fairness and training time.

C. Client Dropout Impacts Fairness and Model Error

Next, we study the impact of the client dropout phenomenon on the overall training process. As pointed out in [17], one major issue of training on mobile/IoT devices is the availability



(a) Error Distribution (b) Error vs. CDR

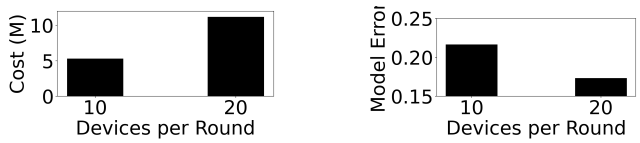
Fig. 2: (a) CCDF of clients' test error evaluated with local datasets. With client dropout, error is noticeably worse. (b) Clients' test error vs. CDR, which are positively corrected.

as clients can dropout even in the middle of training. This can be due to a variety of reasons such as network interruptions, hardware malfunctions, not meeting training criteria, etc. This *client dropout* phenomenon is non-deterministic since we are unaware of the client-end usage behavior, environment and system properties. Therefore, we model this dropout property as a probability of a client to stop training if selected in a round. To study its effect, we randomly assign a probability named ***Client Dropout Ratio (CDR)*** to every client with an exponential distribution with a scale of 0.4, resulting a skewed *CDR* distribution across clients. When a client participates in training, its probability of dropping out equals to its *CDR*. We run the same experimental setup as in the previous section but now use LEAF FEMNIST with the baseline policy instead of [16] since we are not considering policy-driven dropouts like in *LS-LF* here for now. We perform more detailed experiments in the evaluation section.

We use *FedAvg* to run two sets of results: with client dropout (denoted as *with CDR*) vs. no client dropout (denoted as *w/o CDR*) shown in Fig.2a. The CCDF of the error distribution of *with CDR* has a significantly longer tail than *w/o CDR*. The clients on the tail are those with higher *CDR* (> 0.7) and they tend to perform much worse than other clients. This indicates that dropping out clients from the training process results in the global model from being unable to train well on them, which leads to performing very badly on their test datasets. We also observe that the mean model error in the case of *with CDR* is also significantly higher than *w/o CDR*, indicating that the loss of training data due to client dropouts adversely effects the model's performance. Figure 2b shows the is a strong positive correlation between the *CDR* and the resulting global model's error on that client's test dataset. This trend exists since the clients with lower *CDR* tend to participate more in the training process and thus achieve lower test error and vice versa. These results together clearly show that client dropouts can result in higher model error and unfairness, an observation that has widely been ignored by the current state-of-practice and state-of-art.

D. Tradeoff Between Cost and Model Error

One simple way to increase the overall participation is to increase the number of clients selected per round. Works such as [55], [62], [63] attempt to mitigate the performance degradation due to dropouts by increasing the total partici-



(a) Cost (b) Model Error

Fig. 3: (a) Cost (total number of datapoints trained in 2000 rounds in Millions) vs. number of clients selected in each round.(b) Mean model error for different numbers of clients selected per round.

pation of all the devices. We conduct the same experiment as in Figure 2a with client dropout, but increase the total number of clients selected per round from 10 to 20 as a simple measure of increasing the overall participation. We compare *Cost* (measured by the total number of datapoints trained) and *Model Error* in Figure 3a, showing the cost comparison among 10 and 20 client devices selected per round for training over 2,000 rounds. As expected, increasing clients selected per round yields higher cost and imposes a significant heavier burden on clients as more resources are consumed. Meanwhile, it also effectively reduces the model error (see Figure 3b) since more clients per round result in more successful participation frequency for dropout-prone clients. From this experiment, we conclude that increasing participation can benefit model accuracy, but at the cost of a higher burden on clients, thus simply increasing clients selected per round is not a good solution.

IV. Methodology

In this section, we first formulate the problem and then use our observations from the characterization study to develop the *HDFL* framework.

A. Problem Definition

- ***Point 1- Changed overall organization by splitting up the paragraphs into explicit sub-sections. Also added text for better readability.*** Our goal is to design an effective client selection scheduler that optimizes the performance metrics in FL. The scheduling parameter is defined as the selection probability of a client in each training round. Given there are four performance metrics (model error, fairness, cost, and training time) to consider, we formulate the problem as multi-objective optimization. Assume we train a global model G on a set of clients $D = [d_1, d_2, d_3, ...d_n, ...d_N]$ according to a client selection scheduler S defined as the selection probability of each client in training round i: $S_i = [s_1^i, s_2^i, s_3^i, ...s_n^i, ...s_N^i]$. Let the evaluation error of G on the data of individual client in D as $A = [a_1, a_2, a_3, ...a_n, ...a_N]$. The goal is to optimize the model's mean test error defined as $a(S) = 1 - mean(A)$, good-intent fairness defined as $f(S) = var(A)$, total training cost $c(S)$ defined as the total number of data points processed (including dropped out data points), and the training time $t(S)$:

$$minimize\ (a(S), f(S), c(S), t(S)). \tag{1}$$

B. Proposed HDFL Method

In this section, we describe our proposed HDFL framework in details. We first introduce our method of using Underestimation Index *(UEI)* to assign values to parties. We then show how to use UEI to skew the selection probability towards a more fair training, and finally discuss how intelligently excluding certain parties from training would reduce the overall training efficiency.

1) Underestimation Index: Simultaneously optimizing model error, fairness, cost, and training time in FL is challenging due to their complex interactions. To solve this multi-objective problem, we need to first define a measurable metric that can represent and unify the optimization metrics and has the following properties: 1) preserve privacy requirements; 2) can be modeled with scheduling probability. The Underestimation Index (UEI) proposed in [23] has potential to meet the above requirements. UEI is a metric for measuring the distance between a model's prediction results and the actual labels, which is a good indicator of how well a model has learned the features of a dataset, and defined as:

$$UEI_n \ = \ \frac{1}{\sqrt{2}} || \sqrt{P_n^{pr}} - \sqrt{P_n^{act}} ||_2, \tag{2}$$

where n is the client index number, P_n^{act} is the class distribution of the training dataset, and P_n^{pr} is the predicted class distribution of global model. UEI values range from 0.0 to 1.0, where higher UEI means more bias against the training dataset.

A client with a high UEI value indicates that the features in the data of this client are not well captured in the global model, thus the client is "disenfranchised" so far and more training involvement of this client is need to improve fairness. Additionally, reducing UEI across all clients means the features of global data has been well captured and thus improve model error. The participation of clients with low UEI benefits less the training progress, thus such participation may reduce resource efficiency and incur high cost. For clients with the same UEI, their resource efficiency can be different, meaning that in order to reduce UEI by the same amount the number of local datapoints may be different. To reflect the resource efficiency difference, we introduce a cost normalized UEI:

$$CUEI_n \ = \ \frac{UEI_n}{c_n}. \tag{3}$$

Next, we introduce how to quantify *selection probability* and *selection mutualism* and combine them in *HDFL* to solve the multi-objective optimization problem defined in Eq. 1.

2) Selection Probability: Due to the client dropout effects in FL, the eventual participation rate of a client, termed PR_n, depends on both the selection probability of a client S_n and its Client Dropout Ratio CDR_n:

$$PR_n = S_n \times (1 - CDR_n). \tag{4}$$

To design a client selection scheduler that minimizes model error, fairness, and cost, the selection probability shall be set so that $CUEI$ is minimized. In other words, client with higher

TABLE I: Training Setup.

Dataset	Model	Train/Test split	Total Clients/ Selected Per Round	Learning Rate /Batch Size
FEMNIST	2 conv 2 dense	53,839/5,383	179/10	0.004/10
CIFAR10	4 conv 2 dense	50,000/10,000	100/10	0.0005/32
Shakespeare	256 cell lstm 1 dense	115,135/11,513	30/3	0.0003/2

$CUEI$ needs higher selection probability. In addition, clients with high Client Dropout Ratio also need to be compensated with higher selection probability so that their eventual participation rate can be consistent with their selection probability. Therefore, we first define the participation rate of client n as a function of $CUEI$ and then add the Client Dropout Ratio to compute selection probability. For the function, we choose a standard exponential function as it produces a proper skew from $CUEI$ to participation rate:

$$PR_n^i = f(CUEI_n^i) = \sigma * \frac{1}{e^{-CUEI_n^i}}, \tag{5}$$

where i is the round index and n is the client index. σ is a normalization term that converts the $CUEI$ based metric into a probability based metric. By adding the Client Dropout Ratio, we have the selection probability of a client n at round i as:

$$S_n^i = \begin{cases} \frac{PR_n^i}{1-CDR_n^i} & if \;\; CDR_n^i < 1.0 \\ PR_n^i & if \;\; CDR_n^i = 1.0. \end{cases} \tag{6}$$

Because the client selection probability sums to 1 ($\sum_{n=1}^{N} S_n^i = 1$). We can compute σ as:

$$\sigma = \begin{cases} \frac{1}{\sum_{n=1}^{N} \frac{1}{e^{-CUEI_n^i \times (1-CDR_n^i)}}} & if \;\; CDR_n^i < 1.0 \\ \frac{1}{\sum_{n=1}^{N} \frac{1}{e^{-CUEI_n^i}}} & if \;\; CDR_n^i = 1.0. \end{cases} \tag{7}$$

3) Selection Mutualism: To optimize the training time, our main idea is to minimize the straggler and client dropout effects. Here we propose the idea of ***selection mutualism***, which captures the mutualism among the training time of clients in a specific training round. Let the response latency of a client c_i selected in round r be L_i, and the latency of a global training round as -

$$L_r = Max\Big(L_1, L_2, L_3, L_4 ... L_{|C|}\Big) \tag{8}$$

where C is the total number of clients selected in a round and L_r is the training latency of round r. Here, we can see the latency of a global training round is bounded by the maximum training latency of all clients, i.e., the slowest client. Thus, our idea is to mitigate this issue by selecting clients with similar training latencies in a round.

Specifically, clients with similar round training latency are given higher probability to be selected in the same round to reduce the straggler effects and the average Client Dropout Ratio of all clients in a round needs to be smaller than a user defined threshold. The total time per round is determined by the slowest client selected. Randomly choosing clients may group slow and fast clients together, thereby cause the training time of that round bottlenecked by the slow clients. On the other hand, if client selection is restricted to only include devices with similar training time, it can greatly mitigate the straggler issue. The proposed *selection mutualism* approach is more generalizable compared to more strict tier and group-based systems such as [19] since it removes the requirement for fixed tiers and adds support to mitigate client dropout effects to optimize training time.

Algorithm 1 HDFL Algorithm. w_i: the global model for round i, D: list of all participating clients, R: total of training rounds, I: metric update frequency, UEI, c, CDR, L: list of UEI, C, CDR and $training\ time$ metrics for each client, CDR_{max}: minimum average CDR in a round.

Aggregator: initialize weight w_0.
for each round $i = 1$ **to** R **do**
  if $i\%I == 0$ **then**
    $SendGlobalModel(w_i, D)$
    $UEI, c, CDR, L = GetClientMetrics(D)$
  end if
  S = (Calculate using Eq. 6 and 7 with UEI, c, CDR)
  d = (randomly select one client from all clients using S)
  S' = (Calculate using Eq. 8 and 9 S, L)
  s = (randomly select n clients using S' such that CDR_{max} is met)
  for each client c in $s + d$ **do**
    Client: W_c = $LocalSGD(w_i, \delta_i)$
  end for
  w_{i+1} = $\frac{1}{n_c} \sum_{c \in s+d} W_c$
end for

HDFL employs the above methods to make optimal client selection scheduling decisions. As the round training time is bonded by the slowest client (straggler), the key idea to minimize the straggler effect is to adjust the selection probability so that clients with similar training latency can be selected in the same round. Specifically, in a training round, after selecting the first client, we use its training latency as the standard of this round, denoted as L. We adjust clients' selection probability based on the training latency difference between theirs and L. We formulate the mutualism adjusted selection probability as:

$$S_n'^i = f(S_n^i, L_n, L) = \theta * S_n^i * e^{|L_n - L|}, \tag{9}$$

where L_n is the training latency of client n. We use expo-

nential function as an example to reflect the training latency difference's impact on selection probability and such function can be changed to adjust the impact. θ is a normalization coefficient such that $\sum_{i=1}^{N} S_n^{\prime i} = 1$ and can be computed as

$$\theta = \frac{1}{\sum_{n=1}^{N} S_n^i * e^{|L_n - L|}}. \quad (10)$$

To minimize the client dropout effects, the average Client Dropout Ratio of selected clients needs to be below a threshold CDR_{max} to avoid the situation where enough clients are not present to meet the requirement of minimum participants ([64], [65]). The total training time $t(S)$ can be computed as:

$$t(S) = \sum_{i=1}^{I} LS^i, \quad (11)$$

where i is the training round index and I is the total number of rounds. LS^i is the training latency of the slowest client selected in round i, which is impacted by the mutualism based selection probability adjustment above. The full algorithm of *HDFL* is present in Algorithm 1. Here, for each round we first get the values for the variables UEI, c, CDR, and L if it is within profiling interval I. Using these values, we sample clients s such that its participation rate is kept up to a level given a certain CDR using the above equations. The next steps proceed as in standard FL.

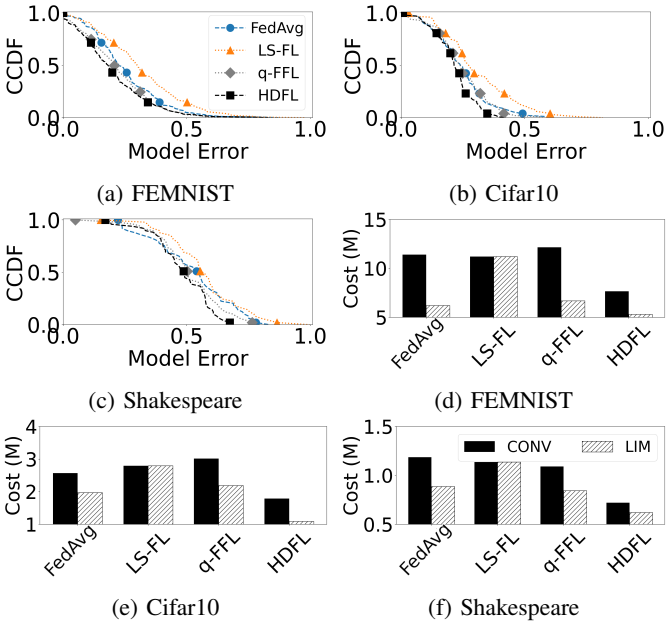
V. Evaluation

We evaluate *HDFL* against the following methods in four performance metrics: model error, fairness, cost, and training time against the following popular FL benchmarks -

- The state-of-the-practice FL: ***Federated Averaging (FedAvg)***. This is the standard cross-device FL implementation in Tensorflow as proposed in [11]. For our experiments, we keep our environment properties (for example, a client's *CDR*) the same for all frameworks regardless of the policies they implement. In this case, it is the baseline FL with no extra mechanisms to handle training time, cost and dropouts.
- The state-of-the-art resource heterogeneity-aware FL which focuses on reducing training time and enable scalability of number of clients [16] (we name it ***LS-FL*** for convenience). This framework does not consider dropouts, resource cost, model fairness and performance.
- The state-of-the-art data heterogeneity-aware FL focuses on fairness solely called ***q-FFL*** [44], and contains no techniques to handle the other metrics.
- The state-of-the-art in data heterogeneity-aware FL – ***FedProx*** [12] which also focuses solely on convergence rate and model performance.

We choose *FedAvg*, *LS-FL*, *q-FFL*, and *FedProx* as baselines because they are open-sourced and we expect similar performance from frameworks such as [19], [53] (not open-sourced).

Benchmarks and Data Heterogeneity. We now evaluate *HDFL* on a real distributed cluster against three popular FL benchmarks. We use the standard **Cifar10**, and we generate its data distribution heterogeneity across clients using the class-wise distribution. We split up the full dataset evenly into 100 parties, where each party has datapoints from at most 5 different classes. This setup is commonly used in state-of-the-art works such as [19], [29]. We also use **FEMNIST** and **Shakespeare** from the federated learning framework *LEAF* [60], which provides a realistic data heterogeneous distribution between devices and has been considered as the new standard for recent state-of-the-art FL works. We use their data distribution provided by default (further details given in Table I and Section III-B).



(a) FEMNIST (b) Cifar10 (c) Shakespeare (d) FEMNIST (e) Cifar10 (f) Shakespeare

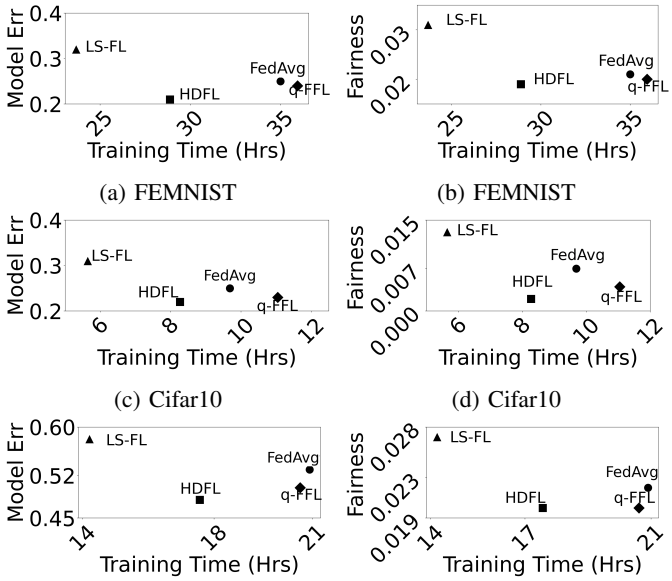
Fig. 4: (a-c) Comparison of CCDF of clients' test error. Demonstrates that *HDFL*can consistently have a lower distribution variance of accuracies (fairness) between devices compared to the *LS-FL*, *q-FFL* and *FedAvg*. (d-f) Cost comparison at the convergence time (CONV) and with constrained training time (LIM) to 24, 5, and 14 hours for FEMNIST, Cifar10, and Shakespeare datasets respectively. The time constraints are set based on the fastest framework to converge. M stands for millions of datapoints.

Testbed and Resource Heterogeneity. We deploy the aggregation server exclusively on a 32-CPU node and every client on separate 2-CPU nodes. We launch the clients on separate individual hardware (details in Table I). It is worth noting that our testbed is among the most practical and largest scale in FL research (to the best of knowledge, only [16], [19] used similar testbed). We randomly assign training latencies per client (via a sleep function) using a Gaussian distribution sampling with a mean of 5 seconds and a standard deviation of 1.5 seconds following [19] to generate a set of clients with variable training latencies to reflect resource heterogeneity. We assign Client Dropout Ratios to each client using an exponential distribution mean of 0.4, which provides enough high and low dropout ratio clients to have a noticeable impact

on training.

A. Model Error and Fairness Analysis

We first evaluate how *HDFL* performs in terms of model error and fairness metrics without other constraints. Figure 4(a-c) shows the *CCDF* of the test errors of clients at convergence across the different datasets and compared among *HDFL* and the state-of-the-arts. The error distribution variance (i.e. the spread of the accuracies) represents the fairness, with lower variance being better. The error distribution mean represents the drop in accuracy of the global model tested on client test sets. Lower values indicate better model performance. *LS-FL* is the worst-performing system due to its larger tail, as well as distinctly higher median and mean values compared to the others due to not purposefully discarding clients and having no mechanism to handle dropouts. *q-FFL* employs fairness optimization, thus it performs better than *FedAvg*. *HDFL* performs the best in both mean and variance in distribution. This can be attributed to its novel policy as it takes into account the clients' dropout probability and *CUEI* when making participation decisions. Specifically, *HDFL* promotes the participation of high dropout ratio clients as well as clients with data on which the model is under-fitting (clients with high *UEI*) to achieve the best model performance and fairness.



(a) FEMNIST (b) FEMNIST

(c) Cifar10 (d) Cifar10

(e) Shakespeare (f) Shakespeare

Fig. 5: Comparison of the training time against model error and fairness (lower is better).

B. Cost Analysis

Figure 4(d-f) shows the cost comparison measured as the total number of datapoints used at convergence or within a time constraint. *HDFL* has lower cost than others across all cases. Using $CUEI$ for client selection enables *HDFL* to be cost-aware, and as a result, *HDFL* tends to prioritize the selection of lower cost clients. *FedAvg*, *q-FFL*, and *LS-FL* are cost oblivious and thus incurs higher cost.

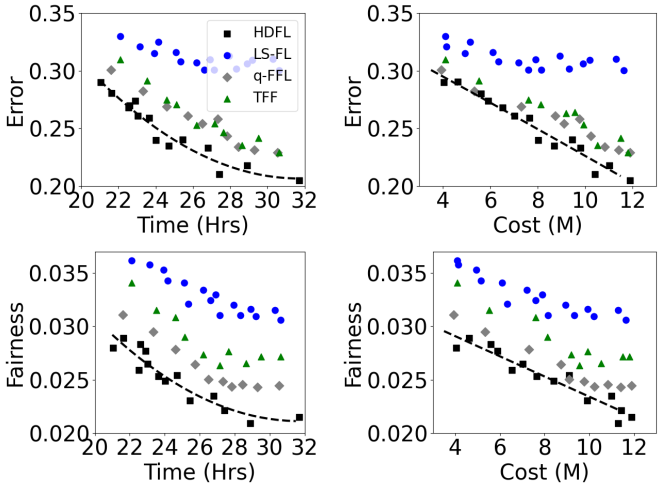


Fig. 6: Pareto optimality comparison results between *HDFL* and others in fairness and model error against training time and cost (datapoints trained). *HDFL* demonstrates the best tradeoffs. M stands for millions of datapoints.

C. Training Time Analysis

Figure 5 shows the comparison of training time at convergence against model error and fairness. *HDFL* consistently achieves better model error and fairness with less training time compared to *FedAvg* and *q-FFL* in all benchmarks. This is because *HDFL* employs the *selection mutualism* to achieve more consistent training time within each round to reduce the straggler effect while *FedAvg* and *q-FFL* has no mechanism to handle stragglers. *LS-FL* has the shortest training time in all cases, but at the cost of significantly compromised model error and fairness. This is expected as *LS-FL* biasedly drops the slowest 25% of clients, which benefits the training time while hurting fairness and accuracy.

D. Pareto Optimality Analysis

Finally, we analyze all the performance metrics together in a full end-to-end manner by demonstrating the Pareto frontier. Figure 6 shows the model error and fairness against cost and training time for the FEMNIST dataset. Across each of the metric combinations, *HDFL* outperforms other frameworks by achieving a better Pareto frontier. *LS-FL* consistently performs the worse with model error due to dropping out clients intentionally, which is also detrimental to fairness. While *FedAvg* and *q-FFL* perform better in model error and fairness, they fail to handle client dropout and result in poor training time and cost.

VI. Conclusion

In this paper, we conduct a first-of-a-kind study on the tradeoffs between the various metrics of importance in FL literature. We demonstrate that current popular methods which aim at optimizing one of these metrics tend to ignore others, resulting in compromises. The observations from the study indicate the need for a framework that can balance

between the various metrics by reducing tradeoffs while optimizing one metric or the other. To this end, we propose *HDFL*, a novel holistic approach-based system that takes into consideration resource heterogeneity, data heterogeneity, and client dropouts to optimize the set of performance metrics simultaneously. We prototype *HDFL* in real distributed FL system and evaluate it using the latest FL benchmarks. The evaluation results show *HDFL* outperforms the state-of-the-art approaches and achieves much better Pareto frontier in every multi-performance metrics optimization scenarios.

VII. Acknowledgements

We are grateful to our anonymous reviewers for their valuable comments and suggestions that significantly improved the paper. This work is sponsored by National Science Foundation CAREER-2048044.

References

[1] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6341–6345.

[2] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," *ACM Computing Surveys (CsUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[3] L. Wu, Y. Chen, H. Ji, and B. Liu, "Deep learning on graphs for natural language processing," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2651–2653.

[4] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, and R. Socher, "Deep learning-enabled medical computer vision," *NPJ digital medicine*, vol. 4, no. 1, pp. 1–9, 2021.

[5] J. K. O'herrin, N. Fost, and K. A. Kudsk, "Health insurance portability accountability act (hipaa) regulations: effect on medical record research," *Annals of surgery*, vol. 239, no. 6, p. 772, 2004.

[6] A. Act, "Health insurance portability and accountability act of 1996," *Public law*, vol. 104, p. 191, 1996.

[7] G. D. P. Regulation, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46," *Official Journal of the European Union (OJ)*, vol. 59, no. 1-88, p. 294, 2016.

[8] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, p. 3152676, 2017.

[9] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, pp. 1–21, 2021.

[10] 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.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," pp. 1273–1282, 2017.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020. [Online]. Available: https://proceedings.mlsys.org/book/316.pdf

[13] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, 2019.

[14] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[15] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4615–4625.

[16] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 374–388.

[17] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[18] A. Imteaj, K. Mamun Ahmed, U. Thakker, S. Wang, J. Li, and M. H. Amini, "Federated learning for resource-constrained iot devices: Panoramas and state of the art," *Federated and Transfer Learning*, pp. 7–27, 2023.

[19] 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," *arXiv preprint arXiv:2001.09249*, 2020.

[20] L. Cai, D. Lin, J. Zhang, and S. Yu, "Dynamic sample selection for federated learning with heterogeneous data in fog computing," in *2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, June 7-11, 2020*. IEEE, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICC40277.2020.9148586

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

[22] G. K. Gudur, B. S. Balaji, and S. K. Perepu, "Resource-constrained federated learning with heterogeneous labels and models," *3rd International Workshop on Artificial Intelligence of Things (AIoT'20), KDD 2020*, 2020.

[23] T. Kamishima, S. Akaho, and J. Sakuma, "Fairness-aware learning through regularization approach," in *2011 IEEE 11th International Conference on Data Mining Workshops*. IEEE, 2011, pp. 643–650.

[24] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," 2020. [Online]. Available: https://openreview.net/forum?id=HJxNAnVtDS

[25] M. Wang, J. Guo, and W. Jia, "Fedcl: Federated multi-phase curriculum learning to synchronously correlate user heterogeneity," *arXiv preprint arXiv:2211.07248*, 2022.

[26] N. Shoham, T. Avidor, A. Keren, N. Israel, D. Benditkis, L. Mor-Yosef, and I. Zeitak, "Overcoming forgetting in federated learning on non-iid data," *arXiv preprint arXiv:1910.07796*, 2019.

[27] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[28] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *International Conference on Learning Representations*, 2020.

[29] D. K. Dennis, T. Li, and V. Smith, "Heterogeneity for the win: One-shot federated clustering," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2611–2620.

[30] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–9.

[31] G. Cheng, K. Chadha, and J. Duchi, "Fine-tuning is fine in federated learning," *arXiv preprint arXiv:2108.07313*, 2021.

[32] H. Mostafa, "Robust federated learning through representation matching and adaptive hyper-parameters," *arXiv preprint arXiv:1912.13075*, 2019.

[33] C. Cortes, M. Mohri, and A. M. Medina, "Adaptation algorithm and theory based on generalized discrepancy," in *21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2015*. Association for Computing Machinery, 2015, pp. 169–178.

[34] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Advances in neural information processing systems*, 2016, pp. 3315–3323.

[35] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on fairness, accountability and transparency*, 2018, pp. 77–91.

[36] I. Y. Chen, P. Szolovits, and M. Ghassemi, "Can ai help reduce disparities in general medical and mental health care?" *AMA journal of ethics*, vol. 21, no. 2, pp. 167–179, 2019.

[37] P. Saleiro, K. T. Rodolfa, and R. Ghani, "Dealing with bias and fairness in data science systems: A practical hands-on tutorial," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and*

Data Mining, Virtual Event, CA, USA, August 23-27, 2020, R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, Eds. ACM, 2020, pp. 3513–3514. [Online]. Available: https://dl.acm.org/doi/10.1145/3394486.3406708

[38] C. DiCiccio, S. Vasudevan, K. Basu, K. Kenthapadi, and D. Agarwal, "Evaluating fairness using permutation tests," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, Eds. ACM, 2020, pp. 1467–1477. [Online]. Available: https://dl.acm.org/doi/10.1145/3394486.3403199

[39] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," in *Advances in neural information processing systems*, 2017, pp. 4066–4076.

[40] A. Abay, Y. Zhou, N. Baracaldo, S. Rajamoni, E. Chuba, and H. Ludwig, "Mitigating bias in federated learning," *arXiv preprint arXiv:2012.02447*, 2020.

[41] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012, pp. 214–226.

[42] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *arXiv preprint arXiv:1908.09635*, 2019.

[43] R. Balakrishnan, M. Akdeniz, S. Dhakal, and N. Himayat, "Resource management and fairness for federated learning over wireless edge networks," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020, pp. 1–5.

[44] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *International Conference on Learning Representations*, 2019.

[45] H. Yu, Z. Liu, Y. Liu, T. Chen, M. Cong, X. Weng, D. Niyato, and Q. Yang, "A fairness-aware incentive scheme for federated learning," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 393–399.

[46] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 88–93, 2020.

[47] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, and S. Srikanteswara, "Energy demand prediction with federated learning for electric vehicle networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[48] Y. Sun, S. Zhou, and D. Gündüz, "Energy-aware analog aggregation for federated learning with redundant data," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[49] 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.

[50] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, 2020.

[51] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[52] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, and A. Lawlor, "Fedfast: Going beyond average for faster training of federated recommender systems," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1234–1242.

[53] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 2021–2031.

[54] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 41–47, 2020. [Online]. Available: https://doi.org/10.1109/MCOM.001.2000410

[55] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8866–8870.

[56] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 15–24.

[57] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: a method of vertical asynchronous federated learning," *arXiv preprint arXiv:2007.06081*, 2020.

[58] K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, J. Rush, and S. Prakash, "Federated reconstruction: Partially local federated learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 220–11 232, 2021.

[59] T.-J. Yang, D. Guliani, F. Beaufays, and G. Motta, "Partial variable training for efficient on-device federated learning," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4348–4352.

[60] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[61] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[62] J. Goetz, K. Malik, D. Bui, S. Moon, H. Liu, and A. Kumar, "Active federated learning," *arXiv preprint arXiv:1909.12641*, 2019.

[63] S. AbdulRahman, H. Tout, A. Mourad, and C. Talhi, "Fedmccs: Multicriteria client selection model for optimal iot federated learning," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4723–4735, 2020.

[64] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[65] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1175–1191.