FedCav: Contribution-aware Model Aggregation on Distributed Heterogeneous Data in Federated Learning

Hui Zeng
College of Computer, National University of Defense Technology
Changsha, China
zenghui116@nudt.edu.cn

Tongqing Zhou*
College of Computer, National University of Defense Technology
Changsha, China
zhoutongqing@nudt.edu.cn

Yeting Guo
College of Computer, National University of Defense Technology
Changsha, China
guoyeting13@nudt.edu.cn

Zhiping Cai†
College of Computer, National University of Defense Technology
Changsha, China
zpcai@nudt.edu.cn

Fang Liu
School of Design, Hunan University
Changsha, China
fangl@hnu.edu.cn

ABSTRACT

The emerging federated learning (FL) paradigm allows multiple distributed devices to cooperatively train models in parallel with the raw data retained locally. The local-computed parameters will be transferred to a centralized server for aggregation. However, the vanilla aggregation method ignores the heterogeneity of the distributed data, which may lead to slow convergence and low training efficiency. Yet, existing data scheduling and improved aggregation methods either incur privacy concerns or fail to consider the fine-grained heterogeneity. We propose FedCav, a contribution-aware model aggregation algorithm that differentiates the merit of local updates and explicitly favors the model-informed contributions. The intuition is that the local data showing higher inference loss is likely to facilitate better performance improvement. To this end, we design a novel global loss function with explicit optimization preference on informative local updates, theoretically prove its convex property, and use it to regulate the gradient descent process iteratively. Additionally, we propose to identify abnormal updates with fake loss by auditing historic local training statistics. The results of extensive experiments demonstrate that FedCav needs fewer training rounds (~34%) for convergence and achieves better inference accuracy (~2.4%) than the baselines (i.e., FedAvg and FedProx). We also observe that FedCav can actively mitigate the model replacement attacks with agile recovery capability towards the aggregation.

CCS CONCEPTS

• **Computer systems organization → Distributed systems**.

*Corresponding Author
†Corresponding Author

ICPP '21, August 9–12, 2021, Lemont, IL, USA

ACM ISBN 978-1-4503-9068-2/21/08…$15.00
https://doi.org/10.1145/3472456.3472504

KEYWORDS

federated learning, data heterogeneity, model replacement

ACM Reference Format:
Hui Zeng, Tongqing Zhou, Yeting Guo, Zhiping Cai, and Fang Liu. 2021. FedCav: Contribution-aware Model Aggregation on Distributed Heterogeneous Data in Federated Learning. In *50th International Conference on Parallel Processing (ICPP '21), August 9–12, 2021, Lemont, IL, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3472456.3472504

1 INTRODUCTION

The growing computation power and wealthy data of mobile devices and AIoT have advanced the recent development of Federated Learning (FL) [7, 13] as a typical paradigm of distributed machine learning. FL explores paralleled local training and global aggregation to cooperatively learn a model with private data retained locally. In Fig. 1, we demonstrate a typical FL process based on the widely adopted FedAvg for aggregation. As shown, local training is performed using the latest global model, while a new global model is attained by taking the average of all the locally trained models.

However, the FedAvg is criticized to have degraded performance when given heterogeneous training data in practice [21, 23]. Generally, the data samples distributed on end devices are non-independent or identical (i.e., non-IID) with imbalance classes, as different users have distinct device usage habits [19]. FedAvg favors the training results on 'large' clients (with more data samples), making it biased on the distributed workers and slow to learn the knowledge on the 'small' clients. Taking the case in Fig. 1 as an example, the unique samples with star label in the left client, though can yield a possibly better performance gain, are considered less important during the aggregation. This will accidentally increase the time for the training to reach convergence and also reduce the performance on model accuracy [21]. It is analyzed that the CNN model trained by FedAvg on the non-IID CIFAR-10 dataset get a 37% accuracy decline [23].

Existing work proposes to solve the above statistical heterogeneity problem by either actively scheduling the data distribution [3, 5, 8, 19, 23] or optimizing the aggregation process [11, 17, 18]. On one hand, some try to sharing the local device data [23] or server-side proxy data [8] to different end devices attain a globally more balanced and IID dataset. However, data sharing may incur new

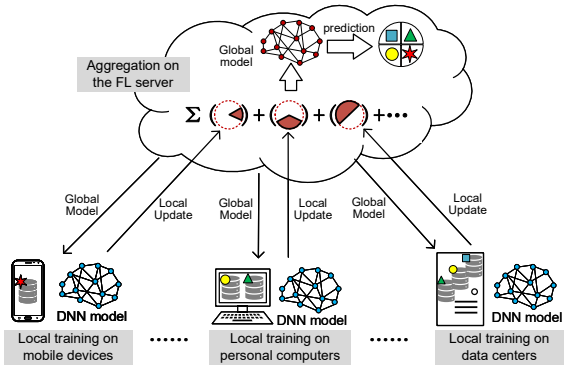


Figure 1: Illustration of the FedAvg aggregation algorithm. Wherein, weighted averaging on local updates are performed with the weights determined according to the amount of corresponding local data.

privacy concerns that go against the intention of FL. Alternatively, much efforts have been devoted to actively select the clients for balanced data distribution. For example, Astraea [5] introduces a virtual mediator to reschedule the clients and rebalance the data. Fed-Focal [17] adopts focal loss to select the best-performing clients for global training. Reinforcement learning is introduced in the design of FAVOR [19] and FedSens [3] to help decide and select the participants in each round of training. However, these approaches may accidentally disclose the local data distribution and usually introduce additional establishment latency for FL. On the other hand, a series of optimization techniques focus on improving the loss function. FedProx [11] adds a proximal term to the objective that helps to improve the stability while there is a large deviation between local updates. FedCurv [18] adds a penalty term to the loss function and compels all local models to converge to a shared optimum. However, these improved aggregation methods fail to consider the imbalanced classes inside each client, as an additional term only smooths the global heterogeneity.

In this paper, we propose a novel model aggregation method for FL, named FedCav, to improve the training performance by endowing Awareness on the Contributions (informativeness) of clients during aVeraging on their local updates. Intuitively, the amount of local data on a device does not reflect its merit and its potential contribution to the global model. In other words, those minority classes play a much more important role beyond their proportion in data [20], e.g., it is more important for G-board to predict SOS precisely than street names. To this end, we introduce the concept of inference loss as the loss of making a prediction on local data with the current global model. The inference loss is designed to be real-time updated and reflecting the quality of the data. Then a global loss function is designed by exponentially adding all the local updates so the inference loss can influence the global aggregation. A local update with larger inference loss, which is considered to be rendered by some 'unseen' samples, will dominate the overall loss value and lead the optimizing direction (i.e., gradients) for the model parameters. Such a contribution-aware optimization process can help the model to agilely learn the difference and to absorb the new knowledge faster, thereby significantly accelerating convergence.

Further, we notice that our preference on high inference loss may be misused by adversaries to perform model replacement attacks easier [1]. To mitigate such threats, we design a detection mechanism by exploiting the differential inference loss in two neighbor training rounds. The update that incurs significant inference performance degradation on most clients is supposed to be abnormal.

The main contributions of this paper are summarized as follows:

(1) We observe that the data statistical heterogeneous will degrade the performance of DNN models learned through FL, we point out that the degradation increases with the variance of class size and give some analysis for our observation (See § 3)
(2) We propose FedCav for training FL, based on differentiating and quantifying the contributions of local updates during aggregation. A novel loss function is designed with rigorous proof on its convex property and the dedicated aggregation function is theoretically deduced (See § 4).
(3) We identify the possible aggravation of model replacement threats when implementing our aggregation algorithm, and bring forth an online anomaly detection mechanism by jointly considering the performance statistics (See § 4.4).
(4) We implement and measure the proposed FedCav on three datasets. Experimental results show that FedCav can converge faster and achieve better performance than the baselines (i.e., FedAvg, FedProx). Meanwhile, it can recover from abnormal updates or attacks agilely (See § 5).

2 RELATED WORK

Data heterogeneity in FL. Federated Learning (FL) allows edge devices to collaboratively learn a shared global machine learning model while keeping all the private training data on the devices [12]. Data heterogeneity is one of the most important problems in FL. Although some recent works have explored some approaches that aim to tackle this problem. We conclude these methods in two parts. One is reducing the difference between clients' local gradients due to the heterogeneity of data. Another is focused on selecting participants. **Reducing the difference of local gradients.** Based on FedAvg, a small subset of training data [23] or server-side proxy data [8] was shared between the clients. However, these methods may be unrealistic: in addition to imposing burdens on network bandwidth, sending local data to the server violates the key privacy assumption of FL. Some try to add a term to make the gradients closer. FedProx adds a proximal term to the local objective, Canh T et al. proposed FedProxVR [4] considering the local accuracy threshold. FedCurv [18] adds a penalty term to the loss function and compels all local models to converge to a shared optimum. However, the additional term increases the computation burden of local devices, and ignore the impact of class imbalance. **Selecting participants.** Some researchers try to let quality clients participate in training and improve the performance of the global model. FAIR [22] adopts the loss reduction during the learning process to quantify the individual learning quality, and leverage the historical quality records to infer the current learning quality and maximize the collective learning quality of all the participants. Reinforcement

learning is used to directly select the participants [19] or help the clients make decisions whether to join a training round [3]. However, selection will make the model more biased to these clients, quality clients are the minority and can not be online for the whole time. And the reinforcement learning model still needs time and resources to train.

Threats against FL. Moreover, recent research also shows the FL vulnerability when facing malicious attacks. Adversarial attacks the model attempt to modify the behavior of the model in some undesirable way [9]. There are generally two levels of scope. **Target attacks**, or backdoor attacks. Bagdasaryan et al. [1] first propose this threat model, which aims to alter the model's behavior on a minority of examples while maintaining good accuracy on other examples by model replacement. **Untarget attacks** or model downgrade attacks, which aim to reduce the model's accuracy, or 'fully break' the global model. Byzantine threat model [2] assumes an adversary controls some clients, these clients send arbitrary but not true local updates to the server and lead the model to divergence.

3 OBSERVATIONS AND PROBLEM STATEMENT

In this section, we conclude and analyze data statistical heterogeneous and investigate the influence of data heterogeneous on model performance in FL. We observe that with the increase of data deviation, the performance deteriorates constantly.

3.1 Preliminaries on data heterogeneity

Some have pointed out that the impact of data heterogeneous on FL comes from the deviation of local model [9]. Based on this, we analyze the data statistical heterogeneous from two aspects. From a global perspective, non-IID is the main cause of local model deviation. On the client's side, the class imbalance leads to a biased local model is another important reason for the model deviation.

Global non-IID. In static view, there are native discrepancies between each client, the discrepancies of environment and function of client lead to the discrepancies of local data. In the dynamic view, the data distribution changes with the clients dynamically participating the training process at any time.

Local class imbalance. In static view, the data is a native class imbalance, the devices of each client can only collect the class imbalance data. In the dynamic view, the data can be generated and collected in real-time, which leads to the class distribution changes and make the local data class imbalance.

3.2 Observations of FL with heterogeneous data

We build multiple data heterogeneous networks and evaluate the performance of FedAvg under these datasets. The experiment settings are shown as follows in detail.

FL settings. We follow the experimental settings in existing FL works [13]. Specifically, the total number of clients is set to be 100 with 30% clients randomly participate in each round. For local training of each client, the size of mini-batch, training epoch is set to be 10 and 5, respectively, and denote the learning rate as 0.001.

Model architecture. The clients collaboratively train a CNN model. We use the model LeNet-5 [10] which is the most classic DNN model for image classification. The model can achieve 99.683% classification accuracy on MNIST after 20 epochs.

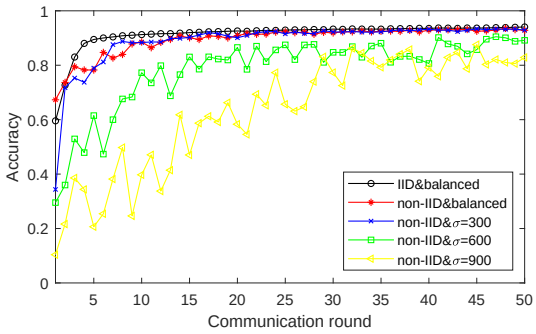


Figure 2: Classification accuracy over communication rounds of FedAvg on 5 different distributed datasets.

Data distribution. According to our analysis on data heterogeneous, we set three different types of data distribution. We consider that IID is a strong constraint, and it must be locally balanced if the global is IID. We are aware that the size of different class is usually different in practice. Considering this fact, we set σ, the statistical variance of different class sizes, to measure the imbalance level of class distribution. We use the MNIST and disperse the instances to clients according to σ. The details are shown in Table 1.

Table 1: Different types of data distribution.

Notation	Description
IID&balanced	Each client is randomly assigned a uniform distribution over all classes, and the scalar of each class is balanced.
non-IID&balanced	Each client is randomly assigned two partitions from two classes, and the scalar of each class is balanced.
non-IID&imbalanced	Each client is randomly assigned two partitions from two classes, and the variance of each class scalar is σ.

The performance is shown in Fig. 2. We can observe that class imbalance results in a significant accuracy decrease. Specifically, with the *balanced* data, FedAvg can get convergence after 5 communication rounds, while it needs approximately 20 or 35 rounds when given *imbalanced* data. Moreover, the accuracy of the model decreases and more unstable with the increase of σ.

3.3 Problem statement

In summary, the data with imbalanced class distribution will cause an accuracy decrease and slow convergent in FedAvg. The underlying reason is that FedAvg weights the updates of each client according to the data size, which is not appropriate in the real world. In our consideration, the divergence of class distribution renders the model unable to obtain the optimal solution through simple averaging operations. Besides, weighting the updates by data size causes the global model to favor the 'large' class, which means that the model can fit the majority class faster and better but

the others need more iterations to get acknowledged. This would significantly limit the practicality of FL.

4 DESIGN OF FEDCAV

As aforementioned, the main challenge of the FedAvg application is that it implicitly treats the merit of clients as the size of data it holds. In view of this, we design FedCav. On the basis of the original FL mechanism, we focus on improvement on model aggregation. Unlike FedAvg, our aggregation favors the model-informed contribution rather than merely data size.

4.1 Overview of the framework

In this subsection, we first give an overview of FedCav. As shown in Fig. 3, it mainly includes **initialization** and **training**. In **initialization**, the FL server first initializes the weights (model parameters) and the optimizer of the neural network model. And it also caches the initialized model as a backup. Then the server connects to the network and gets ready for FL training iteration. The **training** part is a multi-round interaction between clients and the server. Each iteration of training includes three phases: **compute inference loss**, **local training & detection** and **global model update**.

Compute inference loss (phase ①). Each client downloads the global model from the server and then compute the inference loss. The **inference loss** is the value of the global model loss function(e.g., cross-entropy loss function) on the local data, more details will be shown in § 4.2. The inference loss is transferred to the server with the local updates. Considering the dynamic change of local user's data, the inference loss is updated based on the current state.

Local training (clients in phase ②). Each client trains the downloaded global model with its local data and returns the updated model to the FL server when finishing training.

Detection (server in phase ②). The detection mechanism is designed to prevent misusing a high inference loss for an attack. The server compares the differential inference loss in two neighbor training rounds to get detection results, which we will show in § 4.4. If most clients report convergent inference loss than before, the detection result will be set as normal, then the server caches the inference loss for next round detection, prepares to receive the local update; Otherwise, the server abandons this iteration and sends a signal to all clients to reject all local updates.

Global model update (phase ③) depends on the detection result. If the detection result is normal, the server caches the current global model before aggregation, then waits for the clients to report updates. As updates are received, the server aggregates them by processing contribution-aware model aggregation with received inference loss. Otherwise, the server reverses the global model to the cached model and starts a new iteration.

As shown in Fig. 3, an attacker initiates a model replacement attack at round $t-1$, the global model was destroyed after the aggregation in round $t-1$, at the next round t, we detect out that the last update is abnormal, and directly reverse the global model to the cached one. Surely an destroyed model can gradually recover from training, it induces much more communication overheads, reverse to the cached model can greatly reduce these communication overheads. The method abandons the abnormal updates in the previous round, eliminates the long-term impact of the destroyed model or abnormal model.

Table 2: Definitions used in the formulation

Notations	Description
$x_{i,j}$	The jth sample in client i.
$y_{i,j}$	The label of $x_{i,j}$.
n	Total number of all clients in the network.
w	Model parameters.
ℓ	Loss function.
η	Local learning rate.
d_i	Local data of client i.
S_t	The set of participants at communication round t.
D	Total samples size of n clients.
D_{S_t}	The samples size of S_t, $D_{S_t} = \|S_t\|$.
$F(w)$	Global optimization problem(global loss function).
$f_i(w)$	Local loss function value of model w on client i.
w_t	Global model parameters at communication round t.
w_t^i	Local model parameters on client i at communication round t.

4.2 Learning problem with FedCav

In this subsection, we formulate the process of FedCav. Some notations are listed in Table 2. For convenience, we explain some basic operations: $\|\cdot\|$ is L_2 norm, $|\cdot|$ is the size of the set, $\triangleq$ is denoted to 'is defined to be equal to'.

Assume that a distributed network contains n clients, each client has its local data d_i (where $i = 1, \ldots, n$), the total samples size of the network is $D = \sum_i^n |d_i|$, the global loss function is $F(w)$, the loss on the collection of data samples at client i is

$$f_i(w) \triangleq \ell(w, d_i) = \sum_j^{|d_i|} \ell(w, x_{i,j}, y_{i,j}) \tag{1}$$

For most machine learning models, the learning problem is to minimize $F(w)$, where the $F(w)$ donates to the loss sum of all clients, that is $F(w) = \sum_i^n \frac{|d_i|}{|D|} f_i(w)$, we need to find

$$w_{opt} = \arg\min F(w) \tag{2}$$

Due to the complexity of training machine learning models, it's almost impossible to find a closed-form solution. Thus, gradient descent is often adopted to alleviate this limitation.

4.2.1 Distributed Gradient Descent. Distributed gradient descent is widely used in FL. In a distributed network, each client i with its local data d_i has the capacity to compute a local model. At communication round t, where $t = 0, 1, 2 \ldots, n$, we assume the global model is w_t, the local model of client i is w_t^i, each client locally takes one step of gradient descent on the local loss function (defined on its local dataset), which is the **local update**. After one or multiple local updates, a **global aggregation** is performed in the server to update the local parameters at each client to the weighted average of all clients' parameters, then the server deploys the global model to all clients. We define that each **iteration** or **communication round** includes a local update step and a global aggregation step.

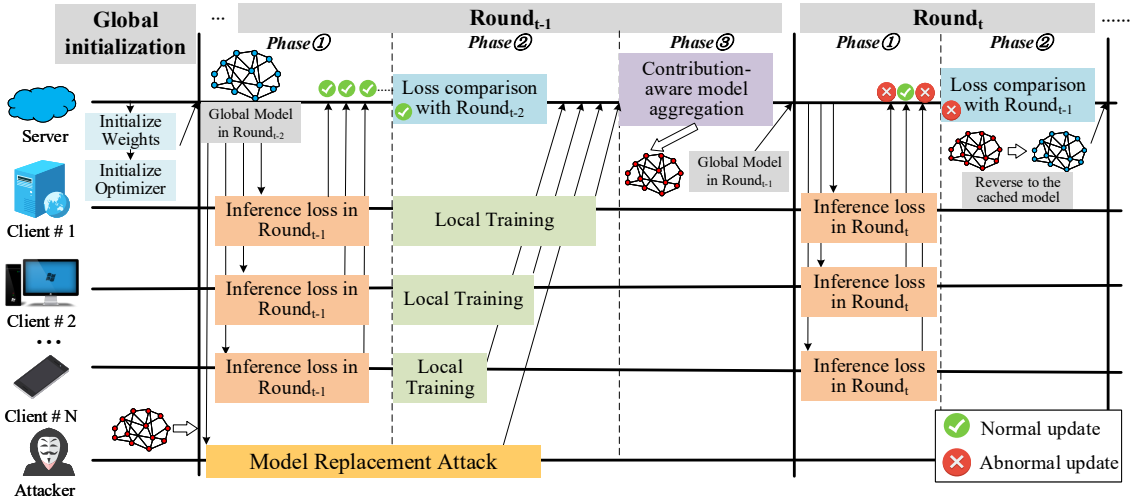


Figure 3: The workflow of FedCav. Note that clients in the two rounds are different.

The local update in each communication round is based on the previous deployed model. For client i, the local update is as follows:

$$w_{t+1}^i = w_t - \eta \partial f_i(w_t) \tag{3}$$

where η is the local model learning rate. According to the global loss function of FL, $F(w) = \sum_i^n \frac{|d_i|}{|D|} f_i(w)$, we can easily get the partial derivative of $F(w)$, which is

$$\partial F(w) = \sum_i^n \frac{|d_i|}{|D|} \partial f_i(w) \tag{4}$$

for global gradient descent, we have

$$w_{t+1} = w_t - \eta \partial F(w_t) \tag{5}$$

so we can get $w_{t+1} = w_t + \sum_i^n \frac{|d_i|}{|D|} \left(w_t - w_{t+1}^i\right)$. And it is the formula of FedAvg if all clients participate. Unfortunately, all clients participation requirement is unrealistic. In real-world applications, only partial clients outputs can be collected by the central server. And the aggregation step performs [12]

$$w_{t+1} = w_t + \frac{|D|}{|D_{S_t}|} \sum_i^n \frac{|d_i|}{|D|} \left(w_t - w_{t+1}^i\right) \tag{6}$$

which can be simplified as $w_{t+1} = w_t + \sum_i^n \frac{|d_i|}{|D_{S_t}|} \left(w_t - w_{t+1}^i\right)$.

4.2.2 Global loss function of FedCav. The purpose of the training process is to minimize $F(w)$. Since it describes the difference of model prediction and true label, the smaller value of $F(w)$, the better the model trained. Rather than use the global loss function which averages all the client's loss, we design a new loss function.

From the observation, the model needs more communication rounds to get convergence while the data distribution is heterogeneous. The reason is the defective gradient descent step. In FedAvg, imbalanced class distribution causes the bias of local loss, some are extremely larger than others, some approximate to zero, decreasing these extremely losses by using average gradient descent step needs more iterations. To address this problem, we consider that the gradient descent step should be different between different clients, a large step is needed for a large local loss. The linear average weakens the influence of each client, so we use the exponential function to scale up, a logarithm to limit the interval of the exponential sum. The learning formula we define as follows:

$$F(w) = \ln\left(\sum_i^n e^{f_i(w)}\right) \tag{7}$$

4.2.3 The optimization problem. For the distributed gradient-descent based learning process, the question narrows down to determining the optimal values of $f_i(w)$. Considering that biased local losses will be scaled up by exponential operations, the optimization process favors a local updates with less and unbiased loss values. We give our definition of FedCav optimization problem.

Definition 1 (Optimization problem of FedCav). *For a distributed network of FL, the optimization problem of FedCav is to find a optimal w_{opt} to minimize the $F(w)$ we define.*

$$w_{opt} = \arg\min F(w) = \arg\min \ln\left(\sum_i^n e^{f_i(w)}\right) \tag{8}$$

To solve this optimization problem, We use basic gradient descent. If we want to get a certain optimal solution, it requires the $F(w)$ we define is convex and can be optimized.

Theorem 1 (Strictly convex function). *For a function $f : R^z \to R$ is a convex function, it requires **1)** $dom(f) \subset R^z$ is a convex set. **2)** $\forall t \in (0,1), \forall x_1, x_2 \in dom(f), x_1 < x_2$, it satisfies $f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$, f is strictly convex function.*

Since $F(w)$ is related to local loss function value $f_i(w)$, $f_i(w)$ is also the optimization problem of local model of client i, which means $f_i(w)$ should get the minimum value in local training, then we can assume it is convex certainly. Theorem 2 shows relation between $F(w)$ and $f_i(w)$.

Theorem 2. *If for all client i, $f_i(w) \geq 0$ and $f_i(w)$ is convex, $F(w)$ is convex.*

Proof. Please refer to Appendix.[1] for the proof. □

However, the requirement for all $f_i(w)$ to be convex is not always stand. For example, when DNN is used, $f_i(w)$ is not convex. We can only get a solution that makes $F(w)$ as minimum as possible if we can find a w that makes each $f_i(w)$ as small as possible.The global update can be written as follows according to (3), (5) and (8)

$$w_{t+1} = \sum_{i}^{n} softmax[f_i(w_t)] w_{t+1}^i \tag{9}$$

The only difference of (9) from FedAvg is the weight factor. Roughly, this shall only impact the convergence speed. The softmax function is used to scale up the gradients from the clients whose local data have not been fit well, and accelerate the convergence on this part of the data.

Note that the softmax function involves the calculation of exponential function, so there is an overflow problem. To solve this, we make the local loss function subtract the maximum value to suppress the overflow. Besides, the softmax function scales up the difference between local loss, if the difference is extreme, the model training process will be jiggling. Inspired by [14], we control the weight of each client, so we choose one simplest method, clipping the local loss $f_j(w)$ with its mean, that is, $\min(f_j(w), mean(f_j(w)))$.

4.3 The aggregation algorithm

In this subsection, we present the algorithm for FedCav. We use the theoretical results above to guide the design of the algorithm.

Algorithm 1: FedCav mdoel aggregation algorithm

Input : Number of clients n, sample ratio q.
Output : The expected optimization result of global model w_{opt}.

1 initialize global model w_0;
2 **for** *each communication round t = 1, 2, . . .* **do**
3 $P_t \leftarrow$ client sets with random sampling ratio q;
4 **for** *each client* $i \in P_t$ *in parallel* **do**
5 $w_{t+1}^i, f_i(w_t) \leftarrow$ `LocalUpdate`(w_t) ;
6 Clip the $f_i(w_t)$;
7 $f_j(w_t) = min\{f_i(w_t), mean(f(w_t))\}$;
8 ; $w_{t+1} \leftarrow \sum_i^{m_t} softmax\left[f_i(w_t)\right] w_{t+1}^i$;
9 $w_{opt} = w_{t+1}$;
10 **return** w_{opt}

As mentioned before, the local updates run on edge clients, and the global aggregation is performed through the assistance of an aggregator, where the aggregator is a logical component, usually deployed in the cloud server. The complete procedure of FedCav is shown in Algorithm 1 and local update in Algorithm 2.

In Algorithm 1, the global server initializes the aggregator and global model, then distributes the global model to all clients. Limited by resource, only part of clients participate in the training, we use q as sample ratio(line 3). For each client, the local update steps based on downloaded global model(line 5). When the server gets the latest response from all clients, the aggregator on the server updates the global model, averaging the local model parameters according to our deduction(line 8). Meanwhile, a clip is needed to prevent an extreme inference loss(line 7).

[1]Appendix is available at https://github.com/zenghui9977/FedCav_appendix/blob/main/appendix.pdf

Algorithm 2: Function **LocalUpdate**

Input : Global model parameter w_t, local epochs E, local mini-batch size B.
Output : Local updated model w_{t+1}^i, inference loss $f_i(w_t)$.

1 Compute the loss based on local data d_i;
2 $f_i(w_t, d_i) \leftarrow \ell\,(w_t, d_i)$;
3 initialize local model $w_t^i \leftarrow w_t$;
4 $\mathcal{B} \leftarrow$ (split d_i into batches of size B);
5 **for** *each local epoch* $e = 1, 2, \ldots, E$ **do**
6 **for** *batch* $b \in \mathcal{B}$ **do**
7 $w_{t+1}^i \leftarrow w_t^i - \eta \partial f_i(w_t^i, b)$;
8 **return** $w_{t+1}^i, f_i(w_t)$

Algorithm 2 shows the pseudo code of local training. Before local updates getting start, each client computes the inference loss $f_i(w_t)$ based on local data (line 2), the inference loss indicates the difference between the prediction of global model and the expectation of local data. Then assign the local model as the downloaded global model, usually SGD or others(line 7). The parameter η is the gradient-descent step size in each iteration.

4.4 Mitigating model replacement attack

Some clients have high inference loss on the current model, indicating that they still have some information to learn, which is believed to improve the performance of the global model. Our approach attempts to understand the contributions of these clients. However, in the real-world network, there exist some distrustful clients who lie about inference loss values and performance information to disrupt the model. Thus it needs to distinguish trusty clients with qualified data from these malicious clients. To solve it, we design a framework to mitigate the effects of malicious attacks.

As a most efficient attack approach, model replacement is common in FL [1]. The main idea of this attack is to utilize the feature when the model gets converged, scale up the attack model to replace the global model. In this method, attackers ambitiously attempt to substitute the new global model w_{t+1} with a malicious model M:

$$M = w_{t+1} = w_t - \sum_{i}^{n} \gamma_i (w_{t+1}^i - w_t) \tag{10}$$

γ_i is the weight in aggregation.

Because of the non-IID training data, each local model may be far from the current global model. while the global model get convergent, these deviations start to cancel out, and the local updates are approximate zero, $\sum_i^n (w_{t+1}^i - w_t) \approx 0$, therefore, the attacker

Table 3: Variables used in the experiment

Variable	Definition
σ	The statistic variance between each class size. The value set to 300, 600 and 900.
α	Fraction of fresh class data that is recently collected by clients and these data have not appeared in previous training process. The value set to 0.1, 0.3 and 0.5.

m can upload as follows to substitute:

$$\begin{aligned} w_{t+1}^{m} &= \frac{1}{\gamma_m}(M - w_t) - \frac{1}{\gamma_m}\sum_{i}^{n}\gamma_i(w_{t+1}^{i} - w_t) \\ &\approx \frac{1}{\gamma_m}(M - w_t) \end{aligned} \tag{11}$$

For FedAvg, γ_i can be $\frac{1}{n}$, $\frac{\|d_i\|}{\|D\|}$ or $\frac{\|d_i\|}{\|Ds_t\|}$, an attacker who does not know γ_i can approximate it by iteratively increasing it every round; for FedCav, γ_i is $softmax(f_i(w_t))$, attackers just need to scale up the local loss, make other γ as small as possible, γ_m approximate to 1. Even if local loss is clipped, attackers can also iteratively increase to get an approximate value.

We design a simple-yet-effective approach to confront this attack. The prime purpose of the model replacement attack is to substitute the global model or just make it break down by participating in distributed training. An obvious feature is that the performance of the model decreases greatly after the model replacement attack, but FedCav is designed to improve the model performance by adding contributions in aggregation. We utilize this feature to detect.

Concretely, the detection design is based on historic statistic comparison and majority voting. For historic statistic comparison, each client compares the inference loss with the maximum of inference loss in the last round, if it decreases greatly, the client should note that there might exist an abnormality. For majority voting, the final detection result consists of the decisions made by more than half of the clients.

In this approach, the server caches the training result of the last communication round, including global model parameter w_{t-1} and local loss $f(w_{t-1})$. At communication round t, n clients upload the results, the server collect w_{t+1}^{i} and $f_i(w)$, $i = 1, 2 \ldots, n$, we define

$$I(condition) = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

We can detect whether there exists attacks in last round by

$$Dr = I\left\{\sum_{i}^{n} I\left[f_i(w_t) > max(f(w_{t-1}))\right] \geq \frac{n}{2}\right\} \tag{13}$$

if $Dr = 1$, we conclude that most clients vote there is an abnormality in the latest model. When an abnormality was detected at w_t, we cancel the aggregation and set the global model as w_{t-1}, we call this process as **reverse**.

5 EXPERIMENTS

5.1 Setup

We implement the FedCav algorithm by Pytorch. For ease of description, the notations used in the experiments are listed in Table 3.

5.1.1 Datasets and Implementation Details. We adopt three widely used datasets: MNIST, FMNIST and CIFAR-10. MNIST is a popular gray-scale image dataset of handwritten digits 0-9, the size of each image is 28×28. FMNIST has the same format as MNIST, but includes images of 10 different fashion items . CIFAR-10 consists of 10 classes of 32×32 images with three RGB channels. There are 50,000 training images and 10,000 test images. We use LeNet-5 for MNIST, 9-layers CNN for FMNIST and Resnet18 for CIFAR-10.

5.1.2 Baselines. We choose the following algorithms as baselines:

(1) Centralized gradient descent [16]. The entire training dataset is stored on one client and the model is trained directly using a standard (centralized) gradient descent procedure.
(2) FedAvg [13]. A widely used algorithm proposed by Google.
(3) FedProx [11]. A distributed optimization framework that tackles the statistical heterogeneity inherent in FL, in our experiment the heterogeneity is the local data.

5.1.3 Data distribution on different clients. We distribute these training samples among 100 clients. For simulating non-IID data distribution, we refer to the method in [6], each client only has two different labels of samples. We omit the word non-IID as all the experiments in this part are performed in the non-IID setting. Also, we consider that the size of each class should be imbalanced. Specifically, we use the factor σ introduced in § section 3 to control the size difference between two labels in a client. In this setting, the size of each class is different and the distribution of each class over the clients is also different.

5.1.4 Training and Control Parameters. In all our experiments, we set the total number of clients $n = 100$. The configurations of local training are local batch size $B = 10$, local epoch $E = 5$, local learning rate $\eta = 0.01$, the sample ratio $q = 0.3$, following the settings in [13]. The same environment of each algorithm is required for fairness.

5.2 Evaluation Results

5.2.1 Classification accuracy with different σ. In this part, we use the top-1 test accuracy as metrics to evaluate the FL model. We don't use recall, precision, and F1 because the test dataset is balanced which means all class have the same cost of misclassification. We first train for a short period, for the reason that pre-training solves the initialization problem and facilitates a fair comparison between multiple algorithms. FedCav achieves an average 2.4% accuracy improvement on the three datasets, details are shown in Table 4.

We set σ to 300, 600, and 900, which represents three typical class imbalance level for FL tasks. We conclude that FedCav can get higher accuracy in those three datasets. With a larger σ, the data in the environment is more imbalanced, and the accuracy decreases obviously. As shown, FedCav can generally overcome the influence and provide relatively better accuracy after the model gets convergence. Especially, in case with $\sigma = 300$, the FedProx get a slightly higher accuracy on MNIST, the reason is that the difference of each class is small, and the performance of FedAvg and FedProx are still compatible.

5.2.2 Classification accuracy with dynamic environment. In our analysis, the dynamic changes of user's data can still have some influence over the class imbalance. We refer to the class which is

Table 4: Average classification accuracy under different levels of data heterogeneity (varying σ) on three datasets. Here we list the accuracy performance of different methods after the learning process gets convergence.

	σ = 300			σ = 600			σ = 900		
	FedAvg	FedProx	FedCav	FedAvg	FedProx	FedCav	FedAvg	FedProx	FedCav
MNIST	0.9333	**0.9391**	0.9365	0.9175	**0.9200**	**0.9200**	0.8467	0.8498	**0.8623**
FMNIST	0.8447	0.8459	**0.8621**	0.8111	0.8236	**0.8349**	0.7397	0.7716	**0.7913**
CIFAR-10	0.4612	0.4644	**0.4686**	0.4239	0.4254	**0.4287**	0.4003	0.424	**0.4387**

collected recently and has never appeared before as **fresh class**. Since some fresh class is surely different with previously existing classes. We set an experiment to simulate the dynamic user's data. In this experiment, we set α as the proportion of fresh class in all data, for example, $\alpha = 0.1$ means 10% of class labels are collected recently and never appear in the previous FL training process. To simulate this, we first separate the $\alpha \times 100\%$ labels as fresh classes, the rest is distributed in the same way with the previous experiment. To highlight the performance of FedCav in the dynamic environment, we pre-train the global model in the common class, then use the three different aggregation algorithms to fit the fresh data.

The results are shown in Fig. 4. We set $\alpha = 0.1$, 0.3 and 0.5, which represents three different situation. $\alpha = 0.1$ denotes only a small part of the class is fresh, and $\alpha = 0.5$ represents half of the class is collected recently. We don't set the $\alpha > 0.5$, since the great change on data will cause the convergence more difficult, in order to get a more stable global model, we set the $\alpha < 0.5$. We conclude that FedCav can get better performance when some fresh data is collected. In those three datasets, the curve of FedCav is better than FedProx and FedAvg in most communication rounds. Since the MNIST is simple, the FedCav just needs few communication rounds to get convergent. For FMNIST, the curve is more approximate to the centralized CNN. For the CIFAR-10, the images with three-channel are more complex than MNIST and FMNIST, in the FL framework its classification accuracy decrease obviously compared to centralized CNN, even though our solution can get some improvements. Moreover, with the increments of α, the classific ation accuracy difference between FedCav and other methods becomes more significant. We analyze that FedProx was worse because the regularization added in our data heterogeneity settings still didn't solve the problem of too much variation across clients and it's not suitable to deal with the dynamic challenge. Meanwhile, the results show that our approach is better to fit the fresh data dynamically, and achieves 34% fewer training rounds average on the three datasets, and the reason that the global model can get convergent quickly is making the fresh data take more contribution for accuracy improvement.

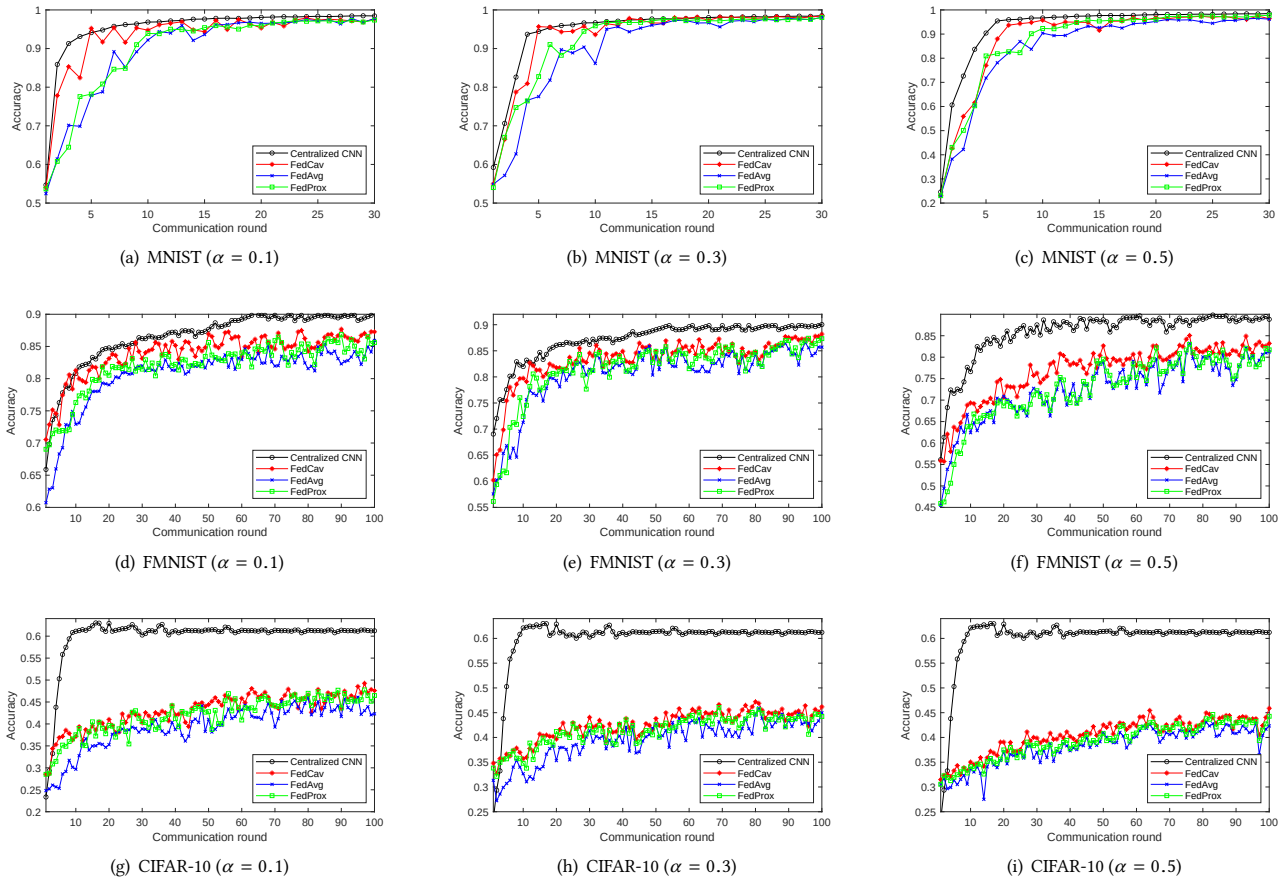
5.2.3 The Impact of clipped local inference loss. Here we explore the impact of clipping in local inference loss. The result is shown in Fig. 5. We can see these curves without clip all are unstable and great up-and-down oscillation, a great drop occurs in the 17^{th} round in Fig. 6(a), 18^{th} and 34^{th} round in Fig. 6(b), more than three drops in Fig. 6(c). It indicates that some clients with a large inference loss make the global model overfitting and finally contribute negative effects on the global model. That implies that it is necessary to clip the inference loss to reduce the jiggling in training.

5.2.4 Performance on abnormal detection. Here we first get the performance when there is a model replacement attack. We simulate an adversary participates in the global training in a fixed round, and the adversary performs a one-time-on-one-round attack, for it's a reasonable and common choice for an adversary. As a random attack, it's simple and of low cost, and multiple attacks may increase the possibility of being exposed. We assume that the adversary trains a malicious model by using all labels flipped data, which results in the adversary's model prediction is totally different from the client's data. We compare FedCav with FedAvg, the result is shown in Fig. 6. We can see that the attack performed at the second round, the accuracy drops to near zero, that's, the model is destroyed. When facing an attack, FedCav can gradually recover from it. However, compared with the training curves in Fig. 4, the model recovery path is tortuous, we consider that might be the confrontation between the normal label and flipped label, for the model prediction which is more similar to flipped label is not consistent with the training data. Notably, the ability to gradually recover from an attack depends on the extent of the attack, a strong attack can still destroy the model trained by FedCav. The experiments only show that FedCav slightly outperforms FedAvg in some attacks.

In our framework, we design a detection mechanism to face this attack. We set three different attacks to measure our proposed detection method. The first one is adversary using the 20% label poisoned model to replace. The adversary flipped 20% labels and train a local poisoned model, then substitute the global model using the model replacement attack. The rest is to flip 50% and 80%. As shown in Fig. 7, there is an attack in the 4th round, and we can detect it in the 5th round, and reverse the global model, which greatly reduces the recovery time compared with Fig. 6. Meanwhile, our detection proposal is also effective in multi-attacks on multi-rounds, for the detection is independent in such attacks. For subsequent attacks in two rounds, the 2^{nd} one won't work, because the global model is reversed and no uploads are allowed when the 1^{st} attack is detected in the later round. For intermittent multiple attacks, they are independent for we reverse the global model to cached one, so they can still be effectively detected.

6 DISCUSSION

Quality of client local data. The premise of our weight-based on loss is that the loss value can reflect the lack of prediction ability of the model on some data. If the loss value can not be well reflected, this method may not have a good effect. If the local data is worthless and greatly different from our expectation, this kind of client will cause a great interference to the global model by using FedCav. If the local data carries much valuable information, our method can accelerate the global model training process. However, it is a big

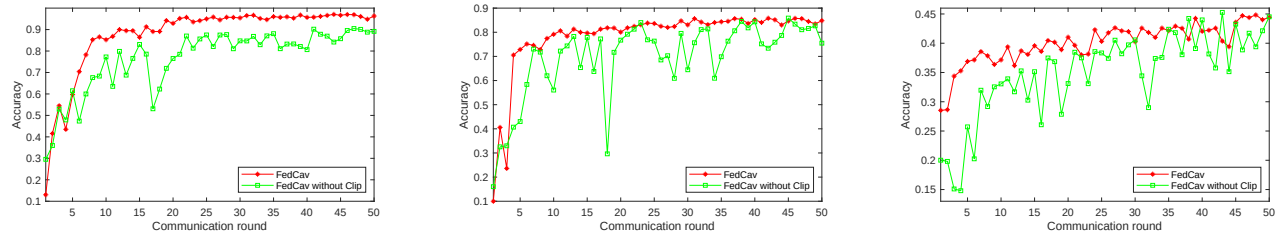


(a) MNIST ($\alpha = 0.1$) (b) MNIST ($\alpha = 0.3$) (c) MNIST ($\alpha = 0.5$)

(d) FMNIST ($\alpha = 0.1$) (e) FMNIST ($\alpha = 0.3$) (f) FMNIST ($\alpha = 0.5$)

(g) CIFAR-10 ($\alpha = 0.1$) (h) CIFAR-10 ($\alpha = 0.3$) (i) CIFAR-10 ($\alpha = 0.5$)

Figure 4: Classification accuracy with dynamic data distribution adjustment controlled by factor α on three dataset. Results on different datasets are painted, wherein FedCav shows a generally stable and superior performance.

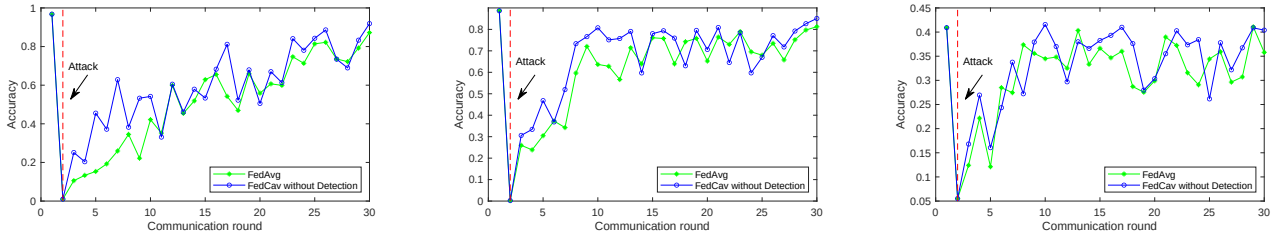


(a) Clip versus non-Clip on MNIST. (b) Clip versus non-Clip on FMNIST. (c) Clip versus non-Clip on CIFAR-10.

Figure 5: Training process with four different algorithms on three datasets. Comparing the difference of whether it is necessary to use the Clip strategy. The curve shows that FedCav without Clip occurs great up-and-down oscillation.

challenge to measure the quality of local data, for that we can not directly access to client's local data in FL design. In our experiments, we just assume that all data is valuable.

Authenticity of updates. In FedCav, the loss and local model parameters are uploaded by the client, the server receives these updates and aggregates. In practice, the authenticity of these updates is important. Fabricating data and uploading false data may bring unexpected threats to the global model. We can use TEE to solve this problem, such as SGX and [15]. It ensures the authenticity and integrity of local calculation results.



(a) Model replacement attack on MNIST. (b) Model replacement attack on FMNIST. (c) Model replacement attack on CIFAR-10.

Figure 6: Part of training process of FedCav without detection and FedAvg after the model replacement attack on three datasets.

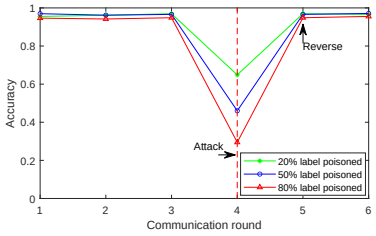


Figure 7: Performance recovery under different strengths of attacks.

Overhead of FedCav. In our analysis, FedCav is still practical while some extra computation and communication overhead is required. For communication overhead, FedCav only needs one extra float (**inference loss**) for each client. The computation overhead is the inference latency at the beginning of each training round. For instance, the latency is 0.0857s in MNIST, which is acceptable compared with the training time $0.1620 \times E$ s (E is the local epoch).

7 CONCLUSION

FedAvg algorithm is widely used in FL, but there are still some problems in real-world applications. In this work, we verify the performance of FedAvg in non-IID & class imbalanced network. We conclude that the distribution of classes greatly influences the global model's performance. Based on this observation, we proposed our method FedCav. Our method is tested on MNIST, FMNIST ,and CIFAR-10 datasets, and the results show that our method has fewer communication rounds and higher accuracy when fresh data come into the distributed network. And we design a detection mechanism for the vulnerability in FedCav, the experiment shows that it can effectively detect attacks.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2020YFC2003400), the National Natural Science Foundation of China (62072465), and the NUDT Research Grants (No. ZK19-38).

REFERENCES

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How To Backdoor Federated Learning. In *Proc. of AISTATS (2020)*, Vol. 108. 2938–2948.
[2] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Proc. of NIPS (2017)*. 118–128.
[3] Zhang Daniel (Yue), Kou Ziyi, and Wang Dong. 2021. FedSens: A Federated Learning Approach for Smart Health Sensing with Class Imbalance in Resource Constrained Edge Computing. In *Proc. of INFOCOM (2021)*.
[4] Canh T Dinh, Nguyen H Tran, et al. 2020. Federated learning with proximal stochastic variance reduced gradient algorithms. In *Proc. of ICPP(2020)*. 1–11.
[5] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang. 2019. Astraea: Self-Balancing Federated Learning for Improving Classification Accuracy of Mobile Deep Learning Applications. In *In Proc. of ICCD (2019)*. 246–254.
[6] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).
[7] Yeting Guo, Fang Liu, Zhiping Cai, Li Chen, and Nong Xiao. 2020. FEEL: A Federated Edge Learning System for Efficient and Privacy-Preserving Mobile Healthcare. In *Proc. of ICPP (2020)*. Article 9, 11 pages.
[8] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. 2018. LoAdaBoost: Loss-Based AdaBoost Federated Machine Learning on medical Data. *CoRR* abs/1811.12629 (2018). arXiv:1811.12629
[9] Edited Kairouz and H. McMahan. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning* 14 (01 2021).
[10] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. 1998. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* (1998), 2278 – 2324.
[11] Tian Li, Anit Kumar Sahu, Manzil Zaheer, et al. 2018. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127* (2018).
[12] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. (2019). arXiv:1907.02189
[13] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of AISTATS (2017)*, Vol. 54. 1273–1282.
[14] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *Proc. of ICLR (2018)*.
[15] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, et al. 2016. Oblivious multi-party machine learning on trusted processors. In *25th USENIX Security Symposium*. 619–636.
[16] S. A. Rahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani. 2020. A Survey on Federated Learning: The Journey from Centralized to Distributed On-Site Learning and Beyond. *IEEE IoTJ* (2020), 1–1.
[17] Dipankar Sarkar, Ankur Narang, and Sumit Rai. 2020. Fed-Focal Loss for imbalanced data classification in Federated Learning. (2020). arXiv:2011.06283
[18] Neta Shoham, Tomer Avidor, Aviv Keren, et al. 2019. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796* (2019).
[19] H. Wang, Z. Kaplan, D. Niu, and B. Li. 2020. Optimizing Federated Learning on Non-IID Data with Reinforcement Learning. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*. 1698–1707.
[20] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. 2020. Towards Class Imbalance in Federated Learning. *arXiv preprint arXiv:2008.06217* (2020).
[21] Peng Xiao, Samuel Cheng, Vladimir Stankovic, and Dejan Vukobratovic. 2020. Averaging Is Probably Not the Optimum Way of Aggregating Parameters in Federated Learning. *Entropy* 22 (2020), 314.
[22] Deng Yongheng, Lyu Feng, et al. 2021. FAIR: Quality-Aware Federated Learning with Precise User Incentive and Model Aggregation. In *Proc. of INFOCOM (2021)*.
[23] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, et al. 2018. Federated Learning with Non-IID Data. *arXiv e-prints* (2018). arXiv:1806.00582