Self-balancing Federated Learning with Global Imbalanced Data in Mobile Systems

Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, Liang Liang

Abstract—Federated learning (FL) is a distributed deep learning method that enables multiple participants, such as mobile and IoT devices, to contribute a neural network while their private training data remains in local devices. This distributed approach is promising in the mobile systems where have a large corpus of decentralized data and require high privacy. However, unlike the common datasets, the data distribution of the mobile systems is imbalanced which will increase the bias of model. In this paper, we demonstrate that the imbalanced distributed training data will cause an accuracy degradation of FL applications. To counter this problem, we build a self-balancing FL framework named Astraea, which alleviates the imbalances by 1) Z-score-based data augmentation, and 2) Mediator based multi-client rescheduling. The proposed framework relieves global imbalance by adaptive data augmentation and downsampling, and for averaging the local imbalance, it creates the mediator to reschedule the training of clients based on Kullback–Leibler divergence (KLD) of their data distribution. Compared with *FedAvg*, the vanilla FL algorithm, Astraea shows +4.39% and +6.51% improvement of top-1 accuracy on the imbalanced EMNIST and imbalanced CINIC-10 datasets, respectively. Meanwhile, the communication traffic of Astraea is reduced by 75% compared to *FedAvg*.

Index Terms—Federated Learning, Distributed Machine Learning, Neural Networks.

1 Introduction

With the enormous success of deep learning (DL) [2], more and more neural network applications enter every aspect of our lives, such as self-driving cars [3], face [4] and speech recognition [5]. The success of deep learning is attributed to its high representational ability of data. Therefore, a larger dataset can always help us build a more complex and accuracte nerual network model. Fortunately, we have the data we need. People generate massive amounts of data, which are scattered across mobile devices. If we can aggregate these distributed data for training, it will significantly boost the quality of deep learning models. However, such a large-scale data aggregation faces some tricky challenges in practice. For example, mobile device users and organizations are unwilling or not allowed to share their data due to security considerations and privacy policies (i.e. GDPR [6]). In addition, the communication and storage required for aggregation are unaffordable. Instead of aggregating data, why not aggregate models? Therefore, many distributed machine learning frameworks (i.e. Parameter Server [7], EASGD [8], Federated Learning [9]) emerge.

Federated Learning (FL) [9], [10], [11], [12] is a promising distributed neural network training approach for deep learning applications such as image classification [13] and natural language process [5]. FL enables mobile devices to collaboratively train a shared neural network model while keeping the training data decentralized. In FL applications, any mobile device can participate in the neural network model training task as a client. Each client independently trains the neural network model based on its local data. A federated learning server then averages the models' updates or parameters from a random subset of FL clients and aggregates them into a new global model.

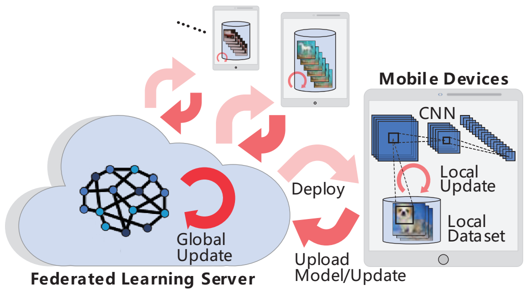


Fig. 1. An example application of federated learning, the server and mobile devices train a CNN image classifier collaboratively.

Fig. 1 depicts a federated learning application in which we aim to train a convolutional neural network (CNN) classifier based on the images from mobile devices. The FL server first deploys a global CNN model to the participating clients. Then each client calculates the gradient update and refreshes their local model in parallel based on their local image dataset. Finally, the FL server integrates the updates from clients and refresh the global model. Repeat the above procedure multiple epoch, the global model can learn the patterns of all participating clients. No private data is transferred throught the model training procedure. In addition, clients can withdraw from this federal learning network at any time. In this way, no private data is transferred throught the model training procedure. FL not only ensures privacy,

- *A preliminary version of this paper was presented at the IEEE 2019 International Conference on Computer Design(ICCD) [1]*
- *Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan are with Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China, and College of Computer Science, Chongqing University, Chongqing 400044, P. R. China.*
- *Liang Liang is with School of Microelectronics and Communication Engineering, Chongqing University, Chongqing 400044, P.R.China*

costs lower latency but also makes the mobile applications adaptive to the changes of its data.

Nevertheless, the main challenge of mobile federated learning is that the distribution of training data on mobile devices is volatile, which results in low prediction accuracy. For example, the class skewness of the image dataset of each client in Fig. 1 may be different, which increases the bias of pattern learned by the FL server. Several efforts have been made to tackle the challenge. McMahan *et al.* propose a communication-efficient FL algorithm *Federated Averaging (FedAvg)* [9], and show that the CNN model trained by *FedAvg* can achieve 99% test accuracy on non-IID MNIST [14] dataset, i.e., any particular user of the local dataset is not representative of the population distribution. Zhao *et al.* [15] point out that the CNN model trained by *FedAvg* on non-IID CIFAR-10 [16] dataset has 37% accuracy loss. Existing studies assume that the expectation of the global data distribution is balanced even though the volume of data on the devices may be disproportionate. In most real scenarios of distributed mobile devices, however, the global data distribution is imbalanced.

In this paper, we consider one more type of imbalanced distribution, named global imbalanced, of distributed training data. In global imbalanced distribution, the collection of distributed data is class imbalanced. We draw a global imbalanced subset from the EMNIST [17] dataset and explore its impact on the accuracy of FL in Section 2.3. The experimental results show that the global imbalanced training data leads to an 8.44% accuracy loss for *FedAvg*.

The accuracy degradation caused by imbalances drives us to design a novel self-balancing federated learning framework, named Astraea. The Astraea framework counterweighs the training of FL with imbalanced datasets by two strategies. First, before training the model, Astraea performs data augmentation [18] to alleviate global imbalance. Second, Astraea proposes to use some mediators to reschedule the training of clients according to the KLD between the mediators and the uniform distribution. By combining the training of skewed clients, the mediators may be able to achieve a new partial equilibrium.

With the above methods, Astraea improves 4.39% top-1 accuracy on the imbalanced EMNIST and 6.51% on the imbalanced CINIC-10 [19] over *FedAvg*. Our rescheduling strategy can significantly reduce the impact of local imbalance and decrease the mean of the KLD between the mediators and the uniform distribution to below 0.2. The proposed framework is also communication-efficient. For example, the experimental results show that Astraea can reduce 75% communication traffic than that of *FedAvg* in achieving 74% accuracy on imbalanced EMNIST.

The main contributions of this paper are summarized as follows.

- We first find out that the global imbalanced training data will degrade the accuracy of CNN models trained by FL.
- We propose a self-balancing federated learning framework, Astraea, along with two strategies to prevent the bias of training caused by imbalanced data distribution.
- We implement and measure the proposed Astraea based on the Tensorflow Federated Framework [20]. The experimental results show that Astraea can efficiently retrieve 52.0% accuracy loss on imbalanced EMNIST and retrieve 46.9% accuracy loss on imbalanced CINIC-10 dataset.

The rest of this paper is organized as follows. Section 2 outlines the background and shows the motivation to overcome the imbalanced FL challenges. Section 3 details the design of Astraea framework. Evaluation results are presented and analyzed in Section 4. Section 6 concludes the paper.

2 Background and Motivation

In this section, we first introduce the background on distributed machine learning, especially federated learning, and briefly introduce imbalanced data learning, then we present the motivation.

2.1 Distributed Machine Learning

Distributed machine learning covers many aspects, including distributed operation of inference and training, distributed storage of dataset and model, ensemble learning from distributed clusters, etc. Many distributed machine learning frameworks are proposed. Dean *et al.* [21] propose DistBelief, which utilizes computing clusters with thousands of machines to train large models. Each computing node stores part of the model, performs gradient calculations and transmits state with other nodes. GraphLab [22] is a distributed machine learning platform for large-scale data-mining systems. Parameter Server [7] framework uses server nodes to maintain globally shared parameters, while both data and workload are distributed over worker nodes. However, the application scenario for the above frameworks is in the cloud High-Performance Cluster (HPC) or the cloud server holding all data, the training data is shared across these distributed nodes. Therefore, the privacy cannot be guaranteed.

To provide privacy support, researchers propose federated learning framework [9], which includes the model aggregation algorithm *FedAvg*. In the FL system, all clients calculate the updates of their weights using synchronous stochastic gradient descent (SGD) in parallel, then a parameter server [7] collects these updates and aggregates them using *FedAvg* algorithm.

With above privacy-preserved training method, a number of mobile deep learning applications based on FL have recently emerged. Hard *et al.* [23] improve the next word predictions of Google keyboard through FL. Bonawitz *et al.* [11] build a large scale FL system in the domain of mobile devices. Hard *et al.* [23] train a recurrent neural network language model for next word predictions through Google keyboard application based on the FL framework. Brisimi *et al.* [24] propose cPDS to solve the problem of predicting hospitalizations due to heart diseases, the privacy of participant's data is preserved with the help of the distributed training method. In addition, Smith *et al.* [25] propose a systems-aware optimization framework MOCHA to address the statistical and systems challenges of FL.

Recent research on federated learning has focused on reducing communication overhead [9], [10], [26], [27] and protecting privacy [28], [29], [30], [31], but only a few studies have noticed the problem of accuracy degradation due to imbalance [15], [32], [33]. However, [9], [15], [32] only discuss the impact of local imbalanced (non-IID) data and assume the global data distribution is balanced, which is rare in the distributed mobile systems.

2.2 Imbalanced Data Learning

Most real-world classification tasks have class imbalance which will increase the bias of machine learning algorithms. Learning with imbalanced distribution is a classic problem in the field of data science [34], and its main solutions are sampling and ensemble learning. Undersampling method samples the dataset to get a balanced subset, which is easy to implement. This method requires a large data set while the local database of the FL client is usually small. Chawla *et al.* propose an over-sampling method SMOTE [35], which can generate minority classes samples to rebalance the dataset. Han *et al.* improve SMOTE by considering the data distribution of minority classes [36]. However, the above method is unsuitable for FL, because the data of clients is distributed and private. Some ensemble methods, such as AdaBoost [37] and Xgboost [38], can learn from misclassification and reduce bias. However, these machine learning algorithms are sensitive to noise and outliers, which are common in the distributed dataset.

Federated learning is designed to be widely deployed on mobile phones and IoT devices, where each device trains a model using its local data. It means that the data distribution of different devices depends on their usages which are likely to be different. For example, cameras deployed in the school capture more human pictures than the cameras deployed in the wild. Furthermore, another kind of imbalance is the class imbalance in the collection of distributed data, such as the word frequency of English literature (following Zipf's law [39]). In order to distinguish between these imbalances in federated learning, we summarize above cases into three categories: 1) Size Imbalance, where the data size on each device (or client) is uneven; 2) Local Imbalance, i.e., independent and non-identically distribution (non-IID), where each device does not follow a common data distribution; 3) Global Imbalance, means that the collection of data in all devices is class imbalanced.

2.3 Motivation

To clarify the impact of imbalanced training data on federated learning, we use the FL framework to train convolutional neural networks (CNNs) based on an imbalanced dataset. However, since there is no large distributed image classification dataset, we build new distributed datasets by resampling EMNIST [17] dataset. EMNIST is an image classification dataset which contains 47 class of handwritten English letters and digits. Although there has a federated version called FEMNIST [40], some unaccounted imbalances contained in the training set and test set.

Imbalance dataset. We build five distributed EMNIST datasets: BAL1, BAL2, INS, LTRF1 and LTRF2, the detail characteristics are shown in TABLE 1. BAL1 and BAL2 are scalar balanced and global class balanced, their only difference is BAL1 is local balanced but BAL2 is random. INS is a scalar imbalanced dataset and the client data size is following the images uploads amount of Instagram users [41]. LTRF1 and LTRF2 further have global imbalance by making the global class distribution following the frequency of the English letters, which is obtained through a corpus of the Simple English Wikipedia (50441 articles in total). In addition, the training data size of LTRF2 is almost twice than of LTRF1. Note that the test set is class balanced and there is no identical sample between the training sets for any clients.

TABLE 1
Settings of Distributed EMNIST Dataset.

	Types of Data Distribution			Sample Size
Notation	Scalar	Global	Local	Train/Test
BAL1	Even	Balanced	Balanced	117500/18800
BAL2	Even	Balanced	Random	117500/18800
INS	Instagram uploads	Balanced	Random	117500/18800
LTRF1	Instagram uploads	Letters frequency	Random	117500/18800
LTRF2	Instagram uploads	Letters frequency	Random	230752/18800

Model architecture. The implemented CNN model has three convolution layers and two dense layers: the first two convolution layers have 12 and 18 channels, 5×5 and 3×3 kernel size (strides is 2), respectively. Above convolution layers followed by a dropout [42] with keep probability 0.5; The third convolution layer has 24 channels, 2×2 kernel size (strides is 1) and following a flattening operation. The last two dense layers are a fully connected layer with 150 units activated by ReLu and a softmax output layer. By the way, the loss function is categorical cross-entropy and the metric is top-1 accuracy. This CNN model has a total of 68,873 parameters and can achieve 87.85% test accuracy after 20 epochs on EMNIST.

FL settings. We adopt the same notation for federated learning settings as [9]: the size of local mini-batch B is 10 and the local epochs E is 5. The total number of clients K is 500 and the fraction C of clients that performs computation on each round is 0.02. For local training, each client updates the weights via Adam [43] optimizer with learning rate $\eta = 0.001$ and no weight decay.

The top-1 test accuracy on five distributed EMNIST datasets is shown in Fig. 2(a). Experimental results show that the global imbalance leads to a significant decrease in accuracy. Qualitatively, for the global imbalanced dataset LTRF1, an 8.44% reduction in accuracy compared to INS was observed (from 83.29% to 74.85%). For LTRF2, a 5.08% reduction in accuracy compared to INS was observed (from 83.29% to 78.21%) although LTRF2 has twice amount of training data than LTRF1. In addition, the random local imbalance does not lead to accuracy degradation, the accuracy on BAL1 and BAL2 is 81.40% and 80.98%. The test accuracy is slightly improved +2.31% in the scalar imbalance case (from 80.98% to 83.29%).

In order to elucidate the influence of the global imbalance, Fig. 2(b) and Fig. 2(c) show the confusion matrixes of BAL1 and LTRF1. The meaning of labels is the same as

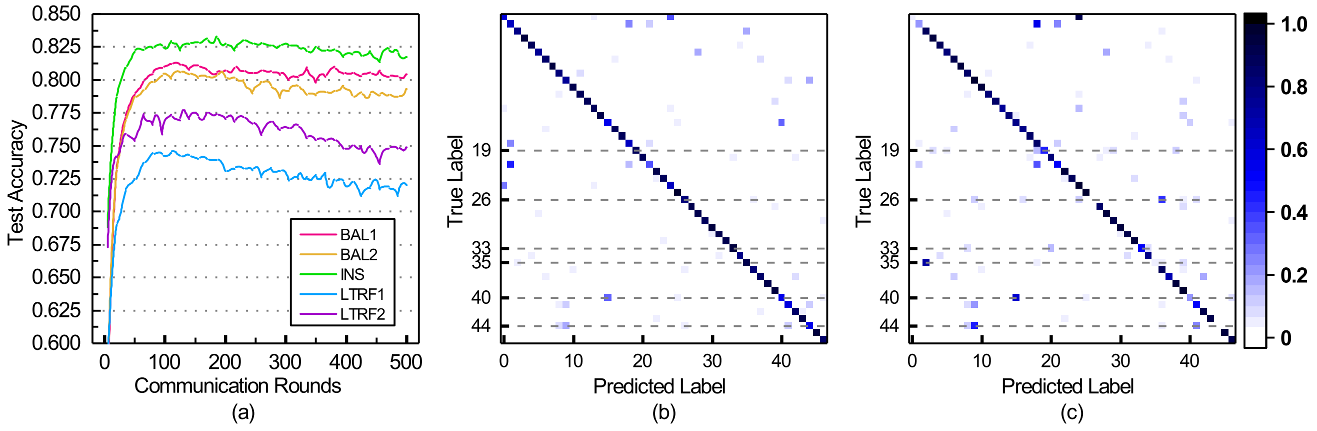


Fig. 2. Accuracy and confusion matrixes evaluated on distributed EMNIST. (a) Test accuracy versus communication rounds on distributed EMNIST; (b) Comparison between the confusion matrixes of CNN models trained on BAL1 dataset and (c) LTRF1 dataset.

EMNIST, labels 0 to 9 correspond to digitals, labels 10 to 46 corresponding English letters (15 letters are merged according to [17]). As shown in the confusion matrix of BAL1, most images are classified correctly, which is represented in the confusion matrix as most of the blue squares spread over the diagonal. However, for the confusion matrix of LTRF1, 6 classes of images (which correspond to the 6 letters with the lowest frequency in English writings) are not well classified as shown by the gray lines. Due to the global imbalance, the training procedures of CNNs are more biased towards classifying the majority classes samples.

In summary, the global imbalance will cause an accuracy loss of the model trained through FL. The main challenge of the mobile FL applications is to train neural networks in the various distributed data distribution. However, uploading or sharing users' local data is not optional because it exposes user data to privacy risks. To address the challenge, we put forward a self-balancing federated learning framework named Astraea, which improving classification accuracy by z-score-based data augmentation and mediator based multi-client rescheduling.

3 Design of Astraea

As aforementioned, the accuracy of federated learning on the distributed imbalanced dataset is lower than that on the balanced dataset. To explain the decrease in accuracy, we mathematically prove that the imbalance of distributed training data can lead to a decrease in the accuracy of FL applications. Based on this conclusion, we design the Astraea framework, the goal of which is to relieve the global imbalance and local imbalance of training data and recover the accuracy.

3.1 Mathematical Demonstration

We define the problem of federated learning training on an imbalanced dataset that leads to accuracy degradation. To formulaize the accuracy degradation in federated learning, we use traditional SGD-based deep learning [2] as the ideal case and derive the update formula of optimal weights. For SGD-based deep learning, the optimization objective is:

$$\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\hat{p}_{data}} L[f(\boldsymbol{x};\boldsymbol{w}),\boldsymbol{y})], \tag{1}$$

where L and $\hat{p}_{data}$ are the loss function and the distribution of training data, respectively. Since the goal is to minimize the test loss, we assume $\hat{p}_{data} = p_{test}$, where p_{test} means the distribution of test set that is balanced for image classification tasks. Both SGD-based deep learning and federated learning use the same test set. We assume that the initial weights for SGD-based deep learning and federated learning are the same:

$$w_0^{(k)} = w_0 = w_0^*. \tag{2}$$

The optimal weights of SGD-based deep learning is updated by:

$$\begin{gathered}\boldsymbol{w}_{t+1}^* = \boldsymbol{w}_t^* - \eta\nabla_{\boldsymbol{w}_t^*}\sum_{i=1}^{n} L(f(x^{(i)};\boldsymbol{w}_t^*),y^{(i)}),\\ (x^{(i)},y^{(i)}) \sim p_{test}.\end{gathered} \tag{3}$$

Because $\boldsymbol{w}^*$ is the weights that achieves the best accuracy on the test set, so it is the optimal weights of federated learning too. For federated learning, the optimization objective is:

$$\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\hat{p}_{data}^{(k)}} L[f(\boldsymbol{x};\boldsymbol{w}^{(k)}),\boldsymbol{y})], k=1,2,...,K. \tag{4}$$

where $\hat{p}_{data}^{(k)}$ means the training data distribution of any client k. Given any client k, the corresponding training data distribution $\hat{p}_{data}^{(k)}$ is imbalanced for the considered federated learning. $\boldsymbol{w}^{(k)}$ is the weights of client k. The weights of each client k that is optimized by gradient descent with learning rate η is updated by:

$$\begin{gathered}\boldsymbol{w}_{t+1}^{(k)} = \boldsymbol{w}_t^{(k)} - \eta\nabla_{\boldsymbol{w}_t^{(k)}}\frac{1}{n_k}\sum_{i=1}^{n_k} L(f(x^{(i)};\boldsymbol{w}_t^{(k)}),y^{(i)}),\\ (x^{(i)},y^{(i)}) \sim \hat{p}_{data}^{(k)}.\end{gathered} \tag{5}$$

The weights of federated learning server are calculated by the *FedAvg* [9] algorithm:

$$\begin{aligned} \boldsymbol{w}_{t+1}^{(Avg)} &= \sum_{k=1}^{K} \frac{n_k}{n} \boldsymbol{w}_{t+1}^{(k)}, \\ &= \sum_{k=1}^{K} \frac{n_k}{n} \boldsymbol{w}_{t}^{(k)} - \frac{\eta}{n} \nabla_{\boldsymbol{w}_t^{(k)}} \sum_{i=1}^{n_k} L(f(x^{(i)}; \boldsymbol{w}_t^{(k)}), y^{(i)}), \\ &\quad (x^{(i)}, y^{(i)}) \sim \hat{p}_{data}^{(k)}. \end{aligned} \tag{6}$$

Since $\hat{p}_{data}^{(k)} \neq p_{test}$, we have $\boldsymbol{w}_{t+1}^{(Avg)} \neq \boldsymbol{w}_{t+1}^{*}$, which means that federated learning cannot achieve optimal weights when training data distribution is imbalanced.

Next, we prove that federated learning can restore the accuracy of models if condition $\hat{p}_{data}^{(k)} = p_{test}$ is satisfied by mathematical induction.

Proposition: $\boldsymbol{w}_t^{(Avg)} = \boldsymbol{w}_t^{*}$ is true for t is any non-negative integer and $\hat{p}_{data}^{(k)} = p_{test}$.

Proof: *Basis case*: Statement is true for $t = 0$:

$$\boldsymbol{w}_0^{(Avg)} = \sum_{k=1}^{K} \frac{n_k}{n} \boldsymbol{w}_0 = \boldsymbol{w}_0^{*}. \tag{7}$$

Inductive assumption: Assume $\boldsymbol{w}_t^{(Avg)} = \boldsymbol{w}_t^{*}$ is true for $t = \mu$ and $\hat{p}_{data}^{(k)} = p_{test}$, $\mu \in Z^{+}$.

Then, for $t = \mu + 1$:

$$\begin{aligned} \boldsymbol{w}_{\mu+1}^{(Avg)} &= \sum_{k=1}^{K} \frac{n_k}{n} \boldsymbol{w}_{\mu}^{*} - \frac{\eta}{n} \nabla_{\boldsymbol{w}_{\mu}^{*}} \sum_{i=1}^{n} L(f(x^{(i)}; \boldsymbol{w}_{\mu}^{*}), y^{(i)}), \\ &= \boldsymbol{w}_{\mu}^{*} - \eta \nabla_{\boldsymbol{w}_{\mu}^{*}} L(f(x^{(i)}; \boldsymbol{w}_{\mu}^{*}), y^{(i)}) = \boldsymbol{w}_{\mu+1}^{*}, \\ &\quad \hat{p}_{data}^{(k)} = p_{test}. \end{aligned} \tag{8}$$

Therefore, by induction, the statement is proved. ■

According to the above conclusion, the difference between the distributions of the training set and test set accounts for the accuracy degradation of FL. Therefore, to achieve a new partial equilibrium, we propose the Astraea framework to augment minority classes and create mediators to combine the skewed distribution of multiple clients. The details of the proposed framework are shown in the following section.

3.2 Astraea Framework

To solve the problem of accuracy degradation, the training data of each client should be rebalanced. An instinct method is to redistributing the clients' local data until the distribution is uniform. However, sharing data raises a privacy issue and causes a high communication overhead. Another way to rebalance training is to update the global model sequentially. Each client calculates updates based on the latest global model and applies its updates to the global model sequentially. It means that the communication overhead and time consumption of the method is K times that of the federated learning (FL). Combining the above two ideas, we propose Astraea, which adds mediators between the FL server and clients to rebalance training.

The overview of the proposed Astraea framework is shown in Fig. 3. Astraea consists of three parts: FL server, mediator, and clients. The FL server is responsible for maintaining a global model $\boldsymbol{w}_0$, deploying the model to mediators, and synchronously aggregating the updates $\Delta\boldsymbol{w}_4$, $\Delta\boldsymbol{w}_7$, $\Delta\boldsymbol{w}_8$ (as shown in Fig. 3) from them using the federated averaging algorithm. The clients can be mobile phones or IoT devices that maintain a local training dataset. The four shapes in the figure represent four classes of data. The clients can be divided into three categories according to the characteristics of their data distribution:

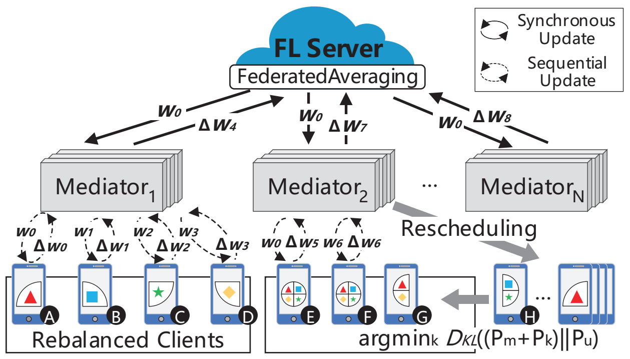


Fig. 3. Astraea Framework Overview.

- Uniform clients, which have enough balanced training data and are ready to run FL applications (e.g. client E and F in Fig. 3).
- Slight clients, which have relatively small amounts of data and are hard to participate in the training process.
- Skewed clients, which have enough training data but prefer to hold certain classes of data which leads to local imbalanced (i.e. client A-D, G, H).

In short, the slight clients and skewed clients introduce scalar imbalance and local imbalance respectively.

Mediators have two functions. One is to reschedule the training processing of the three kinds of clients. For example, as shown in Fig. 3, client G has data with label 0 and label 1, meanwhile, client H has data with label 2 and label 3. Then, the mediator can combine the training of G and H to archive a partial equilibrium.

Second, mediators also need to make the distribution of the collection of data close to the uniform. To measure the extent of partial equilibrium, we using Kullback–Leibler divergence between $P_m + P_k$ and P_u, where P_m, P_k, P_u means the probability distributions of mediator, rescheduling client, and uniform distribution, respectively. In practice, we use the empirical distribution of m and k to approximate its probability distribution P_m and P_k. Besides, by combining multi-client training, a mediator can expand the size of the training set and learn more patterns than a separate client. Note that the mediators are virtual components, it can be deployed directly on the FL server or the mobile edge computing (MEC) server to reduce communication overhead.

Algorithm 1 shows the training procedure of Astraea. First, the FL server initializes the global model to start the training. Then, the FL server starts a new communication round r and sends the global model to the mediators. Next,

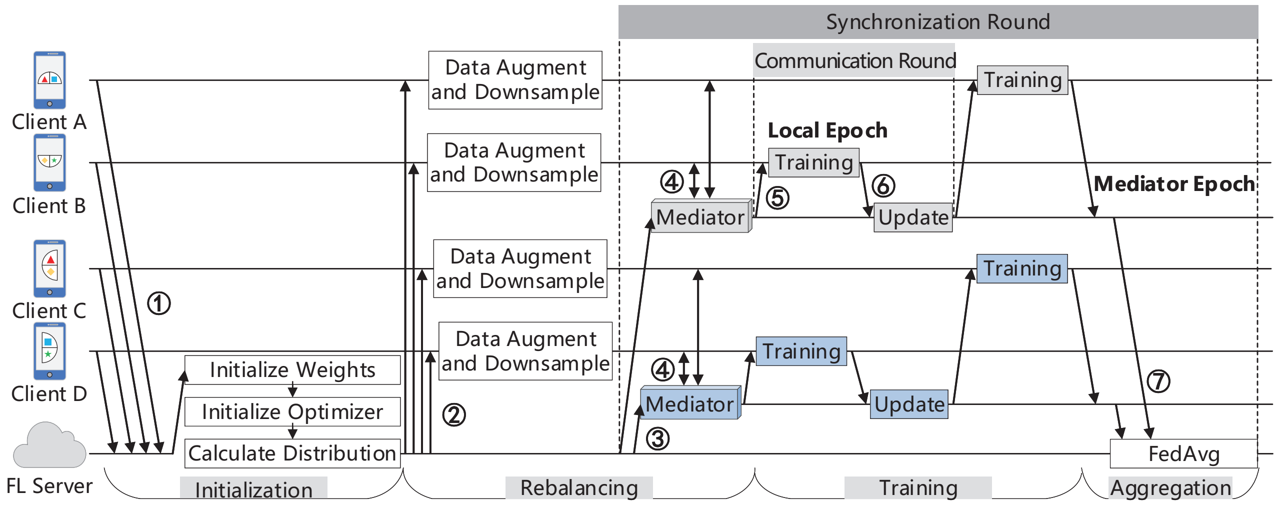


Fig. 4. Astraea Workflow. Z-score-based data augmentation and downsampling(②). Mediator based rescheduling(③④).

Algorithm 1 Astraea distributed neural network training.

```
1:  procedure FL SERVER TRAINING
2:      Initialize $\boldsymbol{w}_0$, $\boldsymbol{w}_1 \leftarrow \boldsymbol{w}_0$.
3:      for each synchronization round $r = 1, 2, ..., R$ do
4:          for each mediator $m$ in $1, 2, ..., M$ parallelly do
5:              $\Delta \boldsymbol{w}_{r+1}^m \leftarrow$MediatorUpdate($m$, $\boldsymbol{w}_r$).
6:          $\boldsymbol{w}_{r+1} \leftarrow \boldsymbol{w}_r - \sum_{m=1}^{M} \frac{n_m}{n} \Delta \boldsymbol{w}_{r+1}^m$. // FedAvg.

7:  function MEDIATORUPDATE($m$, $\boldsymbol{w}$)
8:      $\boldsymbol{w}^* \leftarrow \boldsymbol{w}$.
9:      for each mediator epoch $e_m = 1, 2, ..., E_m$ do
10:         for each clients $i$ in mediator $1, 2, ..., M$ do
11:             // Sequential SGD.
12:             for each local epoch $e = 1, 2, ..., E$ do
13:                 $\boldsymbol{w}_e \leftarrow \boldsymbol{w} - \eta \nabla \ell(\boldsymbol{w};\ \mathbb{X}^{(i)},\ \mathbb{Y}^{(i)})$.
14:                 $\boldsymbol{w} \leftarrow \boldsymbol{w}_e$.
15:     $\Delta \boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{w}^*$.
16:     return $\Delta \boldsymbol{w}$
```

each mediator m coordinates the assigned clients for training and calculates the updates of weights $\Delta \boldsymbol{w}_{r+1}^m$ in parallel. Finally, the FL server collects the updates of all mediators, aggregates these updates with the weight of n_m/n (n_m is the cumulative train size for the clients assigned to mediator m, $n = \sum n_m$). Then, the server updates the global model to $\boldsymbol{w}_{r+1}$ and ends this round. $\boldsymbol{w}_{r+1}$ is the start model for the next (i.e. n+1) communication round.

3.3 Astraea Workflow

The workflow of Astraea includes initialization, rebalancing, training, and aggregation, as shown in Fig. 4.

Initialization. In the initialization phase, the FL server first waits for the mobile devices to join the FL model training task. The devices participate in the training by sending their local data distribution information to the server (①). After determining the devices (clients) to be involved in the training, the FL server initializes the weights and the optimizer of the neural network model and collects the local data distribution of participants.

Rebalancing. We rebalance the training of FL in two steps. First, we perform the z-score-based data augmentation and downsampling to relieve the global imbalanced of training data. However, the augmentation strategy have the potential to increase the local imbalance, which we will demonstrate in Section 4.2. The local imbalance (non-IID) will cause weight divergence and accuracy loss [15]. Therefore, we propose the mediator which asynchronouly receives and applies the updates from clients to averaging the local imbalance. By rescheduling clients' training, the mediator can get a more balanced model. The combination of augmentation and rescheduling strategies can achieve more accuracy improvement than using only one of them, which we will show in Section 4.2. We detail these two strategies below.

The basic idea of our augmentation strategy is to augment the minority classes samples and downsample the majority classes samples based on the global data distribution. In order to divide the majority and minority classes, we borrow the idea of the z-score-based outlier detection algorithm [44] and treat the majority and minority classes as outliers.

We show the details in Algorithm 2. The τ_d and τ_a are the downsampling threshold and augmentation threshold, respectively. Like outlier detection, we treat a class as majority class (denote as $\mathbb{Y}_{down}$) or minority class (denote as $\mathbb{Y}_{aug}$) if its z-score of class size is greater than τ_d or less than τ_a. The R_{ad} is the ratio we use to control how many augmentations are generated or how many samples are retained, and its formula is shown in Algorithm 2 line 9 and line 12. The principle of the above formula is we will generate more augmentations or drop more samples if the z-score is lesser than τ_a or larger than τ_d. Meanwhile, the expected size of class y after augment is in $(C_y,\ \sigma\tau_a + \mu)$, and the expected class size after downsample is in $(\sigma\tau_d + \mu,\ C_y)$.

To simplify this algorithm, we set τ_a to the negative reciprocal of τ_d as shown in Algorithm 2 line 2. Hence, the formula of augmentation ratio and downsample ratio can be simplified into the same formula. We apply this trick in the implementation. Note that, the recommended value of τ_d is 3.0 or 3.5, which is the same as the z-score-based outlier detection algorithm [44]. In addition, since the recommended τ_d is based on standard deviation, it is also robust and can adapt to different data distributions. We show the advantages of this design in Section 4.2.

After completing the calculation of $\mathbb{Y}_{aug}$, $\mathbb{Y}_{down}$, R_{ad}, the server sends these parameters to the clients (②). Then, all clients perform data augment and downsample in parallel as shown in Algorithm 2 line 18 and line 20. For the *Downsample* function, it returns the original sample if it is dropped, otherwise, it returns empty. For the *Augment* function, it takes one sample and generates augmentations, including random shift, random rotation, random shear, and random zoom. We subtract the second parameter of *Augment* function by one, which means that the original sample is not counted. The goal of data augmentation is to mitigate global imbalance rather than eliminate it while a large τ_d will generate too many similar samples, which makes the model training more prone to overfitting.

Algorithm 2 Z-score-based data augmentataion and downsampling for rebalancing distributed datasets.

```
1:  FL Server:
2:  Initialize: $\tau_d$, $\tau_a \leftarrow -(1/\tau_d)$, $R_{ad} \leftarrow [0] * N$.
3:  Calculate the data size of each class $C \leftarrow [C_1, ..., C_N]$.
4:  Calculate the mean $\mu$ and the standard deviation $\sigma$ of $C$.
5:  Calculate the z-score $Z \leftarrow (C - \mu)/\sigma$.
6:  for each class $y$ in $1, ..., N$ do
7:      if $Z_y < \tau_a$ then
8:          Augmentaion set $\mathbb{Y}_{aug} \cup y$.
9:          $R_{ad}[y] \leftarrow (\sigma\sqrt{|Z_y/\tau_a|} + \mu)/C_y$.
10:     if $Z_y > \tau_d$ then
11:         Downsampling set $\mathbb{Y}_{down} \cup y$.
12:         $R_{ad}[y] \leftarrow (\sigma\sqrt{Z_y * \tau_d} + \mu)/C_y$.
13: Send $\mathbb{Y}_{aug}$, $\mathbb{Y}_{down}$, $R_{ad}$ to all clients.

14: Clients:
15: for each client $1, ..., k$ in $K$ parallelly do
16:     for each sample $(x, y)$ in $k$ dataset $(\mathbb{X}^{(k)}, \mathbb{Y}^{(k)})$ do
17:         if label $y$ in downsampling set $\mathbb{Y}_{down}$ then
18:             $(\mathbb{X}^{(k)}, \mathbb{X}^{(k)}) \leftarrow (\mathbb{X}^{(k)}, \mathbb{X}^{(k)})-$
                        Downsample($(x, y)$, $R_{ad}[y]$).
19:         if label $y$ in augmentaion set $\mathbb{Y}_{aug}$ then
20:             $(\mathbb{X}^{(k)}_{aug}, \mathbb{Y}^{(k)}_{aug}) \leftarrow (\mathbb{X}^{(k)}_{aug}, \mathbb{Y}^{(k)}_{aug}) \cup$
                        Augment($(x, y)$, $R_{ad}[y] - 1$).
21:     $(\mathbb{X}^{(k)}, \mathbb{Y}^{(k)}) \leftarrow (\mathbb{X}^{(k)}, \mathbb{Y}^{(k)}) \cup (\mathbb{X}^{(k)}_{aug}, \mathbb{Y}^{(k)}_{aug})$.
22:     ShuffleDataset($\mathbb{X}^{(k)}$, $\mathbb{Y}^{(k)}$).
```

Once all the clients have completed augmentation and downsampling, the FL server creates mediators (③) to rescheduling clients (④) in order to achieve partial equilibrium. In order to get more balanced training, we can increase the collaborating clients of each mediator. However, this will also induce higher communication overhead. Hence, we require that each mediator can only coordinate training for γ clients. We will evaluate the communication overhead of mediators in Section 4.3.

The policy of rescheduling is shown in Algorithm 3. We design a greedy strategy to assign clients to the mediators. A mediator traverses the data distribution of all the unassigned clients and selects the clients whose data distributions can make the mediator's data distribution to be closest to the uniform distribution. As shown in line 6 of Algorithm 3, we minimize the KLD between mediator's data distribution P_m and uniform distribution P_u. The FL server will create a new mediator when a mediator reaches the max assigned clients limitation and repeat the above process until all clients training are rescheduled.

Algorithm 3 Mediator based multi-client rescheduling. D_{KL} is Kullback-Leibler divergence.

```
1:  procedure RESCHEDULING
2:      Initialize: $\mathbb{S}_{mediator} \leftarrow \emptyset$, $\mathbb{S}_{client} \leftarrow 1, ..., K$.
3:      repeat
4:          Create mediator $m$.
5:          for $|\mathbb{S}_{client}| > 0$ and $|m| < \gamma$ do
6:              $k \leftarrow \arg\min_i D_{KL}((P_m + P_i)||P_u)$, $i \in \mathbb{S}_{client}$
7:              Mediator $m$ add client $k$.
8:              $\mathbb{S}_{client} \leftarrow \mathbb{S}_{client} - k$.
9:          $\mathbb{S}_{mediator} \leftarrow \mathbb{S}_{mediator} \cup m$.
10:     until $\mathbb{S}_{client}$ is $\emptyset$
11:     return $\mathbb{S}_{mediator}$
```

Training. At the beginning of each communication round, each mediator sends the model to the subordinate clients (⑤). Each client trains the model with the mini-batch SGD for E local epochs and returns the updated model to the corresponding mediator. The local epoch E affects only the time spent on training per client and does not increase additional communication overhead.

Then, the mediator receives the updated model ⑥ and sends it to the next waiting training client. We call it a *mediator epoch* that all clients have completed a round of training. Astraea loops this process E_m times. Then, all the mediators send the updates of models to the FL server (⑦). There is a trade-off between communication overhead and model accuracy for the mediator epochs E_m times for updating the model. We will discuss this trade-off in Section 4.3.

Aggregation. First, the FL server aggregates all the updates using the *FedAvg* algorithm as shown in Equation 6. Then, the FL server sends the updated model to the mediators and starts the next synchronization round. The main difference between Astraea and the standard FL algorithms in the model integration phase is that Astraea can achieve partial equilibrium. As a result, the integrated model in Astraea is more balanced than that in the standard federated learning algorithms.

4 EVALUATION

4.1 Experimental Setup

We implement the proposed Astraea by modifying the TensorFlow Federated Framework (TFF) [20] and evaluate it through the single-machine simulation runtime provided by TFF. The notations used in the experiments are listed in the TABLE 2.

TABLE 2
Notations and definitions.

Notation	Definition
K	Total number of clients.
B	Local mini-batch size.
N	Number of classes.
c	Number of online clients per synchronization round (per communication round if no mediator).
τ_d	Downsampling threshold.
γ	Maximum number of clients assigned to a mediator.
E	Local epochs. Each client updates weights E times on local data in a communication round.
E_m	Mediator epochs. All clients in a mediator are updated sequentially of E_m times in a synchronization round.

Datasets and models. We adopt two widely used datasets and the corresponding models in the evaluation: 1) Imbalanced EMNIST and its corresponding model. Same as LTRF2 dataset and the CNN model mentioned in Section 2.3, K and B are set to 500 and 10, respectively. 2) Imbalanced CINIC-10 [19] and the CIFAR-10 [16] model described in Keras documentation, where K and B are set to 100 and 50, respectively. We get the imbalanced CINIC-10 by re-sampling CINIC-10 and make the empirical distribution of labels set $\mathbb{Y}$ following a half-normal distribution, which is another kind of global imbalance compared to the imbalanced EMNIST dataset. Furthermore, to prevent leakage, there is no overlap between the training sets for any client.

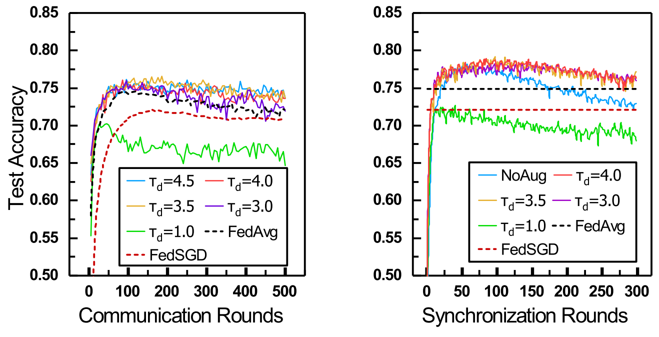
Baseline. We choose the vanilla federated learning algorithm *FedAvg* as the baseline [9], which has been applied to Google keyboard for improving query suggestions [45]. For the baseline evaluated on imbalanced EMNIST, the training configurations are $B = 10,\ E = 5,\ c = 20$. For CINIC-10, $B = 50,\ E = 5,\ c = 20$. Above configurations are proposed in [9] and proved to be efficient.

4.2 Effect of Accuracy

We use the top-1 test accuracy as metrics to evaluate the CNN models. We do not use the recall rate or F1 score because our test set is balanced which means all classes of data have the same cost of misclassification. Furthermore, we consider the maximum test accuracy during training procedure as the final accuracy of the model, which will ignore the effects of overfitting. However, we also show the accuracy curves to indicate whether overfitting occurs.

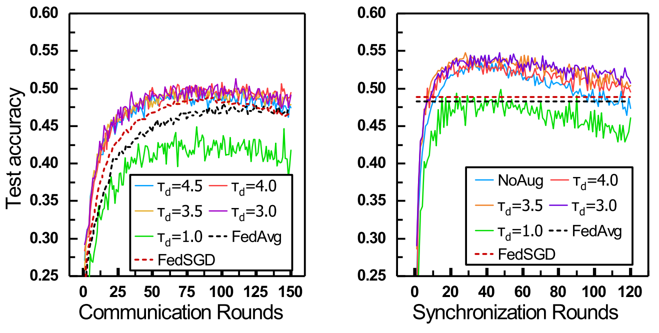
Augmentation vs. mediator: Fig. 5 shows the accuracy improvement on imbalanced EMNIST, including the improved accuracy achieved by the z-score-based data augmentation strategy and the improved accuracy achieved by combining augmentation and rescheduling. We also implement *FedSGD* and compare it with our proposed methond. Fig. 5(a) shows that our augmentation strategy can improve accuracy +1.68% for $\tau_d = 3.5$ except when $\tau_d = 1.0$, a significant decrease in accuracy occurs. The reason for the decline is most training samples are dropped and the number of augmentations is insufficient, resulting in the CNN not learning enough patterns to classify samples.

For our rescheduling strategy, the accuracy of the model is further improved +4.39% (from 74.85% to 79.24%) when $\tau_d = 3.5$ as shown in Fig. 5(b). To explore the accuracy improvement of rescheduling in detail, we measure the



(a) Astraea (Data augmentation and downsampling only) (b) Astraea (Data augmentation, downsampling and rescheduling)

Fig. 5. Classfication accuracy evaluated on imbalanced EMNIST ($N = 47$), two comparative experiments involved: (a) Only the z-score-based data augmentation and downsampling strategy applied, $C = 20,\ E = 1$; (b) Combining the augmentation strategy and the rescheduling strategy, $C = 50,\ \gamma = 10,\ E = 1,\ E_m = 2$.



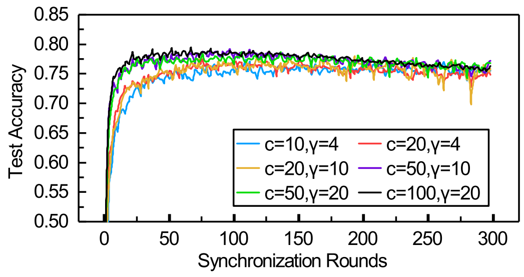
(a) Astraea (Data augmentation and downsampling only) (b) Astraea (Data augmentation, downsampling and rescheduling)

Fig. 6. Classfication accuracy evaluated on imbalanced CINIC-10 ($N = 10$), two comparative experiments involved: (a) Only the z-score-based data augmentation strategy applied, $C = 20,\ E = 1$; (b) Combining the augmentation strategy and the rescheduling strategy, $C = 50,\ \gamma = 10,\ E = 1,\ E_m = 2$.

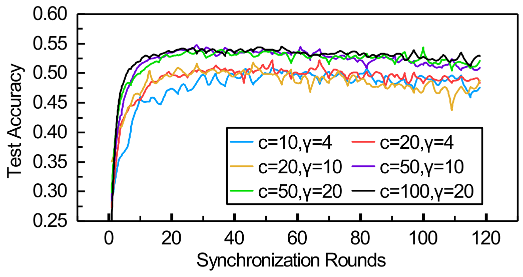
accuracy of the model that data augment is disabled. The results are denoted as *NoAug* in Fig. 5(b). Although +3.6% accuracy improvement can be achieved without using the augmentation strategy, experimental results show that this will make the model more prone to overfitting. The curve of *NoAug* indicates that the accuracy gradually decreases after 100 synchronization rounds (from 76.76% to 73.72%) and is below baseline after 200 rounds.

The accuracy evaluated on imbalanced CINIC-10 is shown in Fig. 6. The data augmentation strategy can improve +3.01% top-1 accuracy when $\tau_d = 3.0$. The accuracy of the model is significantly improved (+6.51% when $\tau_d = 3.5$) after applying the proposed rescheduling strategy. Similar to the *NoAug* in Fig. 5(b), which evaluated on imbalanced EMNIST, Fig. 6(b) shows that the curve of *NoAug* is gradually reduced after 50 synchronization rounds. It means that the model would suffer from overfitting if augmentation is not applied.

The main goal of our rescheduling strategy is to achieve partial equilibrium, which cannot mitigate global imbalance. Thus, combining the two strategies is important and can achieve maximum improvement of accuracy. The choice of τ_d is public, we can adjust its values according to

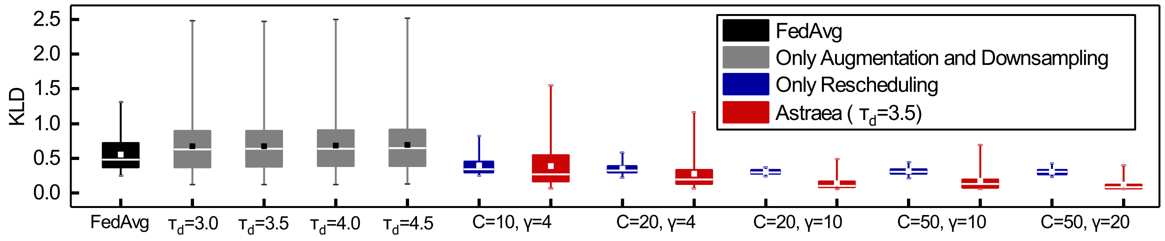


(a) Test accuracy evaluated on imbalanced EMNIST, train with $E = 1$, $E_m = 2$, $\tau_d = 3.5$.

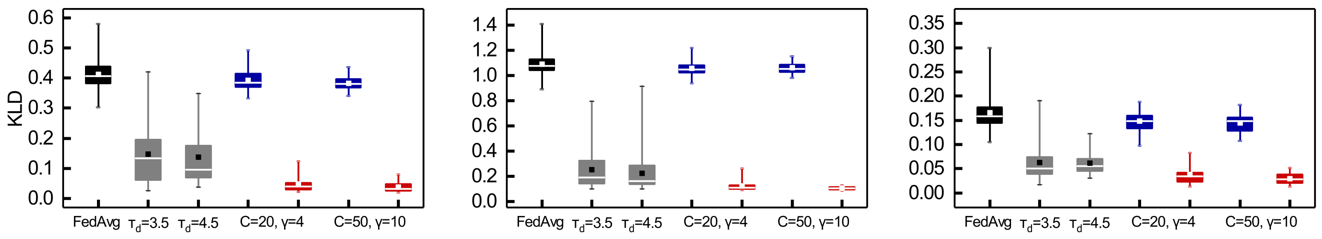


(b) Test accuracy evaluated on imbalanced CINIC-10, train with $E = 1$, $E_m = 2$, $\tau_d = 3.5$.

Fig. 7. Test accuracy of Astraea on imbalanced datasets, train with different numbers of participating clients per round c and different maximum assigned clients limitations of mediator γ.



(a) KLD evaluated on imbalanced EMNIST



(b) KLD evaluated on imbalanced CINIC-10 with various global imbalances (Left: Half-normal; Middle: Zipf($s = 2$) [39]; Right: Linear)

Fig. 8. Kullback–Leibler divergence between P_u and P_m (or P_k). The horizontal axis represents different configurations of mediator (or client), the white line indicates the median and the square indicates the mean.

the task. However, as shown in the Fig. 5 and Fig. 6, too small τ_d is inappropriate. Hence, the recommended range for τ_d is 3.0 to 3.5, which is also the recommended setting of the z-score-based outlier detection algorithm.

c vs. γ: c is the number of online clients per round, which determines the scale of training in each synchronization round. γ is the max assigned clients limitation of the mediator, which determines the scope of partial equilibrium. We explore the impact of c and γ on the training procedure of Astraea.

The experimental results on imbalanced EMNIST are shown in Fig. 7(a). In the first 100 rounds, the training of model converges faster with the increase of c. However, after 150 rounds, the accuracy is slightly reduced, especially for the models trained with a large c. For example, the accuracy is reduced from 77.83% to 75.33% when $c = 100$ and $\gamma = 20$. It means that the CNN models are over-training and suffered from overfitting. To remedy the loss of accuracy caused by overfitting, we can use the regularization strategy called early stopping [46], in which optimization is halted based on the performance on a validation set, during training. Furthermore, experimental results show that a larger γ does not help to improve the accuracy of the model.

To further explore the impact of mediator and augmentation configurations on the equilibrium degree, we show the KLD of all clients or mediators in Fig. 8. The KLD of each client of traditional FL denotes as *FedAvg* is calculated by $D_{KL}(P_k||P_u)$, which means the equilibrium degree of FL client without augmentation and rescheduling. The P_k is the empirical distribution of labels set of client k and the P_u is the probability distribution of uniform. The KLD of *Aug* is the equilibrium degree of Astraea framework without rescheduling, which is calculated by $D_{KL}(P_{k'}||P_u)$. The $P_{k'}$ is the empirical distribution of labels set of client k', whose training data has been augmented. The KLD of each mediator of Astraea framework, which denotes as multiple sets of C and γ, is calculated by $D_{KL}(P_m||P_u)$. The P_m is the empirical distribution of labels set of mediator m, which has been assigned.

The KLD results are shown in Fig. 8(a) and Fig. 8(b). In Fig. 8(a), all distributions are left-skewed and the mean of KLD of *FedAvg* is the highest (0.550), indicating that the distribution of *FedAvg* is most imbalanced. Our augmentation

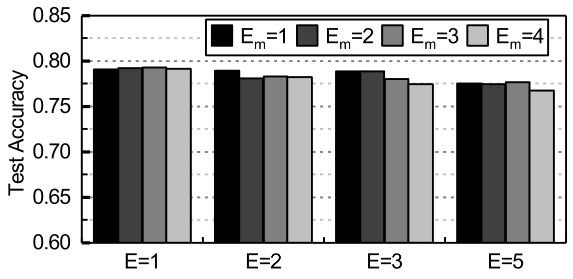


Fig. 9. Maximum test accuracy evaluated on imbalanced EMNIST. Trained with different local epochs E and mediator epochs E_m, 300 rounds, $C = 50$, $\gamma = 10$, $\tau_d = 3.5$.

strategy, as mentioned before, may increase local imbalance and introduce some new outliers. As shown in the gray bar in Fig. 8(a), the augmentation strategy generally causes the range of KLD to increase and become more left-skewed. On the other hand, the KLD measured on various distributions as shown in Fig. 8(b) shows that our augmentation strategy can also reduce local imbalances, but new outliers are introduced.

The results of applying rescheduling only are shown in the blue bar in Fig. 8. Because some highly skewed clients can arichieve partial equilibrium throught complementary rescheduling, the outliers are significantly reduced and the mean of KLD is sightly lower.

As shown in the red bar in Fig. 8(a), combining two strategies can significantly rebalance data distribution (the mean decreased from 0.550 to 0.174 when $C = 50,\ \gamma = 10$) with the shrink of interquartile range and the increase of c. Besides, the results show that a larger c or γ can reduce the range of KLD. This suggests that mediators can achieve better partial equilibrium when more clients participate in training or more clients are assigned to the mediators. In summary, the accuracy improvement of Astraea increases as the scale of the training expands. The results in Fig. 8(b) show similar conclusions, Astraea significantly reduces the KLD, compared to *FedAvg* or the single strategy case.

Local epochs vs. mediator epochs. Here we explore the impact of local epochs E and mediator epochs E_m on training. The E represents the number of epochs for local gradients update in a communication round and the E_m is the number of epochs for mediator weights update in a synchronization round. The maximum test accuracy evaluated on imbalanced EMNIST is shown in Fig. 9.

Fig. 9 shows that increasing local epochs does not bring significant improvement in accuracy. A larger local epochs can even cause a drop in accuracy. In our experiments, the accuracy of the CNN model drops 1.83% on average if the local epochs E changes from 5 to 1. For mediator epochs, we observe that training with $E_m = 2$ only slightly improve accuracy (+0.14%) compared with $E_m = 1$ when the local epoch E is 1. Although experimental results show that increasing E_m does not significantly help improve the accuracy of the model, we can reduce communication overhead by increasing E_m to accelerate the convergence of the model. Thus, there is a tradeoff between accuracy and communication overhead. We will demonstrate it in the following section.

4.3 Overhead

We discuss three kinds of overheads of Astraea framework: Time, storage and communication. We ignore the computational overhead of Astraea for the additional calculations, such as augmentation and rescheduling, require few computational resources and can be calculated on the FL server. We use the imbalanced EMNIST dataset in this section.

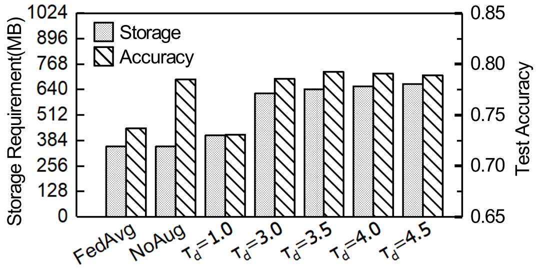
Time overhead. There are three major tasks that require additional time in Astraea: Data augmentation, rescheduling, and extra training epochs of the mediators. As shown in Algorithm 2, the time spent in augmentation is $\max_{k\in K}\sum_{i=1}^{N}(R_{ad}[i]-1)C_i^{(k)}t_a$, where t_a is the time required to augment an image. We assume the values of t_a and τ_d are fixed, then the time complexity is $O(N\mu^2)$. Since data augmentation is only performed once at the initialization phase, the time consumption is negligible to the whole training procedure. The process of rescheduling is shown in Algorithm 3, where we use a greedy strategy to search the clients for rescheduling. The time complexity of the searching process is $O(c^2)$. If the data distribution of clients is static, Astraea only performs rescheduling once. In contrast, if the data distribution of the client is dynamically and rapidly changing, Astraea needs to reschedule in each synchronization round. The main time overhead of Astraea framework is the model training. In FL, the time spent on each communication round is $E \times T$, T is the training time of a local epoch. In Astraea, the time spent on each synchronization round is $E_m\gamma E \times T$.

Storage overhead. The proposed Astraea requires the clients to provide additional storage space to store the augmented samples. We show the trade-off between storage and accuracy in Fig. 10. The experimental results shown in Fig. 10(a) show that Astraea can improve 4.77% accuracy on imbalanced EMNIST without additional storage requirements. It further improves 0.78% accuracy with 81.8% additional storage space ($\tau_d = 3.5$). However, we emphasize that the data augmentation strategy is necessary, and the model in *NoAug* is prone to overfitting as shown in Fig. 5(b). Although it seems that the storage overhead is large, it is acceptable after allocating the overhead to each client. In our experiments, the total additional storage space for data augmentation is 289 MB, i.e., about 592 KB per client. The required storage space is increased with the increase of τ_d. $\tau_d = 1$ shows a unsatisfactory result, where τ_d is set too small.

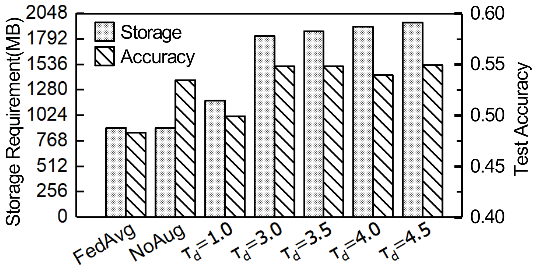
Fig. 10(b) shows the storage overhead evaluated on CINIC-10. Astraea improves 5.15% accuracy without additional storage requirements. It further improves 1.34% accuracy with 103.8% additional storage space ($\tau_d = 3.0$), particularly, average 9.27 MB per client.

Communication overhead. Due to the training of the clients in each mediator is squential, each synchronization round in Astraea costs more traffic than each communication round in FL. The traffic of each communication round can be calculated by $2c|\boldsymbol{w}|$, where $|\boldsymbol{w}|$ is the size of all parameters. Hence, the traffic of each synchronization round is $2|\boldsymbol{w}|(\lceil c/\gamma\rceil + c)$.

Experimental results in TABLE 3 show that Astraea is actually more communication-efficient than FL. It is because Astraea can improve model convergence speed. As a result,



(a) Storage overhead on imbalanced EMNIST, Astraea train with $E = 1,\ E_m = 2,\ C = 50,\ \gamma = 10$



(b) Storage overhead on imbalanced CINIC-10, Astraea train with $E = 1,\ E_m = 2,\ C = 50,\ \gamma = 10$

Fig. 10. Storage overhead versus model accuracy evaluated on imbalanced datasets.

TABLE 3
Communication consumption to reach a target accuracy for Astraea (with different value of mediator epochs E_m note as Med1-8, $E = 1$), versus FedAvg (Baseline), the local epochs E does not affect communication overhead.

Imbalanced EMNIST, Target Top-1 Accuracy: 74%						
Notaion	c	γ	τ_d	E	E_m	Cost(MB)
FedAvg(baseline)	20	✗	✗	5	✗	861
Med1	50	10	3.5	1	1	473 (0.55×)
Med2	50	10	3.5	1	2	663 (0.77×)
Med3	50	10	3.5	1	3	623 (0.72×)
Med4	50	10	3.5	1	4	706 (0.82×)

Imbalanced CINIC-10, Target Top-1 Accuracy: 48%						
Notaion	c	γ	τ_d	E	E_m	Cost(MB)
FedAvg(baseline)	20	✗	✗	1	✗	19431
Med5	20	4	3.5	1	1	7936 (0.41×)
Med6	20	4	3.5	1	2	5627 (0.29×)
Med7	20	4	3.5	1	3	5002 (0.26×)
Med8	20	4	3.5	1	4	4906 (0.25×)

our framework requires fewer communication costs that FL in achieving a required accuracy. For imbalanced EMNIST, the communication consumption of training a CNN using FL to reach 74% top-1 accuracy is 861 MB whereas Astraea uses merely 473 MB (noted as Med1 in TABLE 3). That is, Astraea achieves a 45.0% reduction in communication cost.

For imbalanced CINIC-10, our framework can reduce the communication consumption to reach 48% accuracy by up to 75% when $E_m = 4$.

5 Related Work

Previous work on the imbalanced data learning in FL focused on the non-IID data. The work proposed FL framework evalutas *FedAvg* on a non-IID partition of EMNIST dataset and shows that *FedAvg* is roboust to non-IID data. However, Zhao *et al.* [15] show that the accuracy of *FedAvg* reduces 55% for CNNs trained on highly skewed non-IID data. Sattler *et al.* [32] propose a communication protocol STC to compress upstream and downstream communications of FL, which make the FL framework more robust to non-IID data. Our work focuses on the global imbalance challenge in FL. In order to distinguish from the previous works about non-IID data, we adopt the random distribution as the local data distribution.

There are some work related to Astraea. Mohri *et al.* [33] recently proposed a new optimization algorithm named Agnostic Federated Learning (AFL), which is designed to minimize the training loss calculated based on any possible weighted combination of clients' domain. AFL shows better accuracy than standard federated learning trained in a single domain. However, this method only evaluated on few domains, exactly 2 and 3 domains. On the other hand, since the baseline of AFL is fitted in a single domain, the AFL is some kind of generalization. The motivation of our work is similar to AFL, but we mainly focused on the global imbalance challenge. Furthermore, we evaluated our framework on a large scale, i.e. 500 clients.

Combining federated learning and meta-learning [47] provides a new way to solve the imbalanced FL problem. Chen *et al.* [48] propose a federated meta-learning framework named FedMeta, which treats the training of clients as learning tasks in meta-learning. However, FedMeta is supposed to provide customized models to improve the performance of a recommendation system. Hence, the discussion on the imbalance challenge in this work is not comprehensive. In addition, there is no conflict between our work and FedMeta, and the self-balancing framework we proposed preserves the possibility of implementing the meta-learning strategy.

Recently, Luo *et al.* [49] propose a real-world image datasets for federated learning, where the training data is natural non-IID and global imbalancd.

6 Conclusion

Federated learning is a promising distributed machine learning framework with the advantage of privacy-preserving. However, FL does not handle imbalanced datasets well. In this work, we have explored the impact of imbalanced training data on the FL and 8.44% accuracy loss on imbalanced EMNIST caused by global imbalance be observed. As a solution, we propose a self-balancing FL framework Astraea which rebalances the training thought 1) Performing z-score-based data augmentation and down-sampling; 2) Rescheduling clients by mediators in order to achieve a partial equilibrium. Experimental results show that the top-1 accuracy improvement of Astraea is +4.39% (retrieve 52.0% loss) on imbalanced EMNIST and +6.51% (retrieve 46.9% loss) on imbalanced CINIC-10 (vs. *FedAvg*). Finally, we measure the overheads of Astraea and show its communication is effective.

References

[1] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *Proceed-*

ings of the 37th International Conference on Computer Design (ICCD). IEEE, 2019, pp. 246–254.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[3] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5419–5427.

[4] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4690–4699.

[5] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[6] P. Voigt and A. Von dem Bussche, "The EU general data protection regulation (GDPR): A practical guide," *Springer International Publishing*, 2017.

[7] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014, pp. 583–598.

[8] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging sgd," in *Proceedings of the 28th International Conference on Neural Information Processing Systems (NeurIPS)*, 2015, pp. 685–693.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.

[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] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," in *Proceedings of the 2nd SysML Conference*, 2019.

[12] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[14] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[15] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[16] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[17] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: Extending mnist to handwritten letters," in *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[18] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" in *Proceedings of the 2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–6.

[19] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "Cinic-10 is not imagenet or cifar-10," *arXiv preprint arXiv:1810.03505*, 2018.

[20] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[21] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker *et al.*, "Large scale distributed deep networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1223–1231.

[22] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, 2012.

[23] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[24] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics*, vol. 112, pp. 59–67, 2018.

[25] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 4427–4437.

[26] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency v2v communications," in *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[27] D. Liu, C. Yang, S. Li, X. Chen, J. Ren, R. Liu, M. Duan, Y. Tan, and L. Liang, "Fitcnn: A cloud-assisted and low-cost framework for updating cnns on iot devices," *Future Generation Computer Systems*, vol. 91, pp. 277–289, 2019.

[28] 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 (CCS)*. ACM, 2017, pp. 1175–1191.

[29] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS)*, 2018, pp. 7564–7575.

[30] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proceedings of the 2019 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 2512–2520.

[31] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 15, pp. 911–926, 2020.

[32] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, pp. 1–14, 2019.

[33] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 4615–4625.

[34] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 21, no. 9, pp. 1263–1284, 2009.

[35] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research (JAIR)*, vol. 16, no. 1, pp. 321–357, 2002.

[36] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," in *Proceedings of the International conference on intelligent computing (ICIC)*. Springer, 2005, pp. 878–887.

[37] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for AdaBoost," *Machine Learning*, vol. 42, no. 3, pp. 287–320, 2001.

[38] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2016, pp. 785–794.

[39] L. A. Adamic and B. A. Huberman, "Zipf's law and the internet." *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

[40] 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.

[41] A. Bodaghi and S. Goliaei, "Dynamics of instagram users," Jul. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.823283

[42] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, vol. 28, no. 3, 2013, pp. 1058–1066.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[44] B. Iglewicz and D. C. Hoaglin, *How to detect and handle outliers*. Asq Press, 1993, vol. 16.

[45] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[46] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[47] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*. JMLR, 2017, pp. 1126–1135.

[48] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *arXiv preprint arXiv:1802.07876*, 2018.

[49] J. Luo, X. Wu, Y. Luo, A. Huang, Y. Huang, Y. Liu, and Q. Yang, "Real-world image datasets for federated learning," *arXiv preprint arXiv:1910.11089*, 2019.