FedCSS: Joint Client-and-Sample Selection for Hard Sample-Aware Noise-Robust Federated Learning

ANRAN LI, School of Computer Science and Engineering, Nanyang Technological University, Singapore
YUE CAO, School of Computer Science and Engineering, Nanyang Technological University, Singapore
JIABAO GUO, School of Cyber Science and Engineering, Wuhan University, China
HONGYI PENG, School of Computer Science and Engineering, Nanyang Technological University, Alibaba-NTU Singapore Joint Research Institute & Alibaba Group, Singapore
QING GUO, IHPC and CFAR, Agency for Science, Technology and Research, Singapore
HAN YU*, School of Computer Science and Engineering, Nanyang Technological University, Singapore

Federated Learning (FL) enables a large number of data owners (a.k.a. FL clients) to jointly train a machine learning model without disclosing private local data. The importance of local data samples to the FL model vary widely. This is exacerbated by the presence of noisy data, which exhibit large losses similar to important (hard) samples. Currently, there lacks an FL approach that can effectively distinguish hard samples (which are beneficial) from noisy samples (which are harmful). To bridge this gap, we propose the Federated Client and Sample Selection (FedCSS) approach. It is a bilevel optimization approach for FL client-and-sample selection to achieve hard sample-aware noise-robust learning in a privacy preserving manner. It performs meta-learning based online approximation to iteratively update global FL models, select the most positively influential samples and deal with training data noise. Theoretical analysis shows that it is guaranteed to converge in an efficient manner. Experimental comparison against six state-of-the-art baselines on five real-world datasets in the presence of data noise and heterogeneity shows that it achieves up to 26.4% higher test accuracy, while saving communication and computation costs by at least 41.5% and 1.2%, respectively.

CCS Concepts: • **Computing methodologies** → **Distributed algorithms**; • **Information systems** → **Data mining**.

Additional Key Words and Phrases: sample selection, federated learning, bi-level optimization

ACM Reference Format:
Anran Li, Yue Cao, Jiabao Guo, Hongyi Peng, Qing Guo, and Han Yu. 2023. FedCSS: Joint Client-and-Sample Selection for Hard Sample-Aware Noise-Robust Federated Learning. *J. ACM* 1, 3, Article 10 (September 2023), 23 pages. https://doi.org/XXXXXXX.XXXXXXX

*Han Yu (han.yu@ntu.edu.sg) is the corresponding author.

Authors' addresses: Anran Li, anran.li@ntu.edu.sg, School of Computer Science and Engineering, Nanyang Technological University, Singapore; Yue Cao, CAOY0033@e.ntu.edu.sg, School of Computer Science and Engineering, Nanyang Technological University, Singapore; Jiabao Guo, garbo_guo@whu.edu.cn, School of Cyber Science and Engineering, Wuhan University, China; Hongyi Peng, hongyi001@e.ntu.edu.sg, School of Computer Science and Engineering, Nanyang Technological University, Alibaba-NTU Singapore Joint Research Institute & Alibaba Group, Singapore; Qing Guo, tsingqguo@ieee.org, IHPC and CFAR, Agency for Science, Technology and Research, Singapore; Han Yu, han.yu@ntu.edu.sg, School of Computer Science and Engineering, Nanyang Technological University, Singapore.

0004-5411/2023/9-ART10 $15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Federated Learning (FL) is an emerging learning paradigm that performs distributed training of models on data owners, a.k.a. FL clients (*e.g.*, sensors or edge devices) with potentially sensitive local data [1–4]. The global model can be obtained by aggregating a large corpus of data owners' local models without exposing their data to any third party. Recent FL applications include next-word predictors on users' smartphones [5, 6], traffic sign classifiers [7], and clinical decision support model for COVID-19 [8]. However, due to the complexity of the optimization problem, resource overhead incurred by data owners (*e.g.*, local computational and communication cost) have become key issues hindering the training of large-scale FL models. When training FL models, it is observed that not all local samples are equally important [9, 10]. On one hand, knowledge embedded within some samples has been extracted after a number training epochs. These samples can be ignored thereafter without affecting the global model. On the other hand, many participants may possess erroneous data (*e.g.*, data with label noise), which negatively affect global model performance [11, 12]. For example, data collected via crowdsourcing may contain mislabeled samples [13]. Such data should be excluded from training.

To achieve fast convergence and obtain good FL model accuracy, in this work, we focus on identifying the most positively influential clients and the most influential samples within these clients so that the usage of limited computation and communication resources can be optimized. There are a number of existing data selection methods for achieving high-performance model under centralized learning settings [10, 14–16] and FL settings [9, 17, 18]. In centralized learning, a series of works use importance sampling [10, 14, 16] or hard negative mining [15, 19] to accelerate training and improve the performance of neural networks. However, these methods either ignore the presence of noisy data, or select samples with high training losses as they can result in large changes in model parameters without distinguishing whether they are hard samples (which are beneficial) or noisy data (which are harmful). Other works address the issues of training set noises with data resampling, or more generally by assigning a weight to each example and minimizing a weighted training loss. The example weights are typically calculated based on the training loss (*i.e.*, AdaBoost [20]). Again, they cannot distinguish hard samples from noisy samples. There are few works involving hard sample identification and reducing noisy data influence [21, 22]. These works integrate an auxiliary teacher deep network to supervise the training of the student network by focusing on probably correct samples. However, the auxiliary network suffers from cumulative errors from sample selection bias [23] and often incur substantial computation overhead, especially when facing large volumes of training data.

For data selection in FL settings, existing works leverage client selection [17, 24] or sample-level data selection [9, 25] either through Shapley values [26, 27] or through proposing various statistical selection metrics, *e.g.*, differences of model weight updates [28, 29], training losses [24] or gradients [9]. These methods have the following limitations. Firstly, Shapley value-based methods are prohibitively expensive to calculate, which makes it difficult to scale up. Secondly, statistical metric-based methods either do not take into account the existence of label noise, or simply use hand-designed loss or gradient thresholds to filter out noisy clients or samples. In practice, prior knowledge of the thresholds for distinguishing noisy samples from non-noisy samples for different datasets training different models is not available, rendering these methods inapplicable. Besides, these methods cannot be used to distinguish positively influential clients and local samples from noisy ones. This is because to mitigate noisy labels, samples with small training losses are preferred as they are more likely to be clean data. When attempting to identify positively influential samples, those with high training losses are preferred as they induce large changes in model parameters, thereby accelerating model convergence. These approaches are not well suited to

federated learning in which noisy data are common and identifying influential samples to achieve fast model convergence is highly desirable.

To address these limitations, we propose a novel client-sample bilevel selection scheme in this paper - the federated client and sample selection (FedCSS) approach. It accelerates the training of FL models by focusing the limited computational and communication resources on the most positively influential samples, while dealing with label noise. In order to learn general forms of training set noise and positively influential samples, FedCSS leverages a small clean validation to guide FL training, which is not uncommon in practical FL scenarios[1] and is adopted in existing works [31, 32]. It jointly solves the sample selection problem and global FL model construction problem as a bilevel optimization problem: the inner objective aims to construct the optimal global FL model, while the outer objective aims to learn the optimal selection probabilities for data samples. Our key contributions are summarized as follows.

(1) **Privacy-Preserving Convergence Guarantees.** FedCSS ensures that the local models converge to the globally optimal inner model in a privacy-preserving manner. The convergence of the inner optimization is further leveraged to ensure the outer optimization of accurate sample selection probability computation.
(2) **Computation & Communication Efficiency.** FedCSS achieves highly efficient sample selection probability estimation by bypassing expensive global Hessian computations through the inner loop learning process. It is based on approximating differentiation through online first-order approximation over few communication rounds.
(3) **Interpretable Selection.** The rules governing the FedCSS decision-making with regard to updating client and sample selection probabilities are transparent. The selection probabilities for the positively influential data owners and samples are boosted, while those for noisy ones are suppressed. They can be used to support the emergence of sustainable FL ecosystems through additional mechanisms such as reputation and incentives.

We evaluated FedCSS across various FL tasks with real-world workloads. Compared to the eight state-of-the-art approaches, FedCSS can efficiently handle various data distribution across thousands of data owners. It improves test accuracy by 1.2%-26.4%, while incurring 41.5%-93.3% lower computation cost and 1.2%-79.6% lower communication cost, thereby achieving the best efficiency and accuracy in the presence of data noise.

2 RELATED WORK

2.1 Sample Selection in Centralized Learning

Sample selection for centralized learning can be divided into two categories. The first one focuses on dynamically selecting hard samples that are important to the model to form a training batch in order to accelerate the training of neural networks and improve their performance. Curriculum learning (CL) [33–35] based methods aim to train a network with easier samples in early training stages and gradually exposing model to more difficult training samples. However, applying them in FL is challenging as CL requires a reference model to determine the hardness of samples [36], which is not readily available in FL scenarios. Importance sampling methods [10, 14, 16] have been proposed to prioritize samples that can result in the biggest changes in the parameters which reduce the variance of the gradient estimates. These methods employ either the gradient norm or the loss to compute each the importance of each sample. However, these methods either overlook the existence of data noise, or select samples with high training losses without distinguishing if they are hard samples (which are beneficial) or noisy data (which are harmful). The second category

[1]For instance, Google enlists its employees to type with Gboard to create the root dataset for its federated next-word prediction approach [30]

mitigates data noise via data resampling (*e.g.*, choosing the correct proportion of samples to train the network) or generally by assigning a weight to each example and minimizing a weighted training loss) [37–40]. They also cannot distinguish hard samples from noisy ones.

There are few works that can handle hard samples and noise data simultaneously [21, 22]. They generally leverage an auxiliary teacher deep network to supervise the training of a student network by focusing on probably correct samples. However, they are not applicable to FL due to the following issues. Firstly, the auxiliary network suffers from cumulative errors due to sample selection bias [23]. Secondly, they require direct access to all training samples which violates the privacy preservation requirements of FL. In addition, training the large extra auxiliary network incurs substantial computation and communication overhead, which may not be affordable for resource constrained FL devices.

2.2 Client and Sample Selection in FL

Existing approaches in FL settings either focus on client selection [17, 24, 26, 27, 29] or sample-level data selection [9, 25, 32] with the aim of accelerating model convergence.

2.2.1 FL Client Selection. Existing FL client selection works can be divided into two branches. One branch utilizes the client contribution evaluation results [26, 27], *e.g.*, calculated through Shapley values, to select clients and optimize the global model aggregation. However, since Shapley value-based methods account for complex dependencies, they are prohibitively expensive to calculate. The other branch proposes various statistical selection metrics, *e.g.*, differences of model updates [28, 29, 41], local losses [24] or utilities [17] calculated based on local losses, and measure the metric values of clients to make selections. However, these methods treat all samples from a selected client as equally important, which can lead to a waste of communication and computational resources when updating local models on unimportant samples, resulting in suboptimal model performance.

2.2.2 FL Sample-level Selection. One category [32, 42] proposes to distributedly select relevant data before FL training based on a benchmark model, without considering dynamic data importance during training. The other category dynamically selects samples to compose local batch during training. The work [9] prioritizes local samples with high importance using gradient norm upper bound to achieve fast convergence speed. Another work [25] proposes a deadline control strategy for FL that predicts the deadline for each round with different client training data. These methods, however, either ignore noise data, or simply use pre-defined thresholds to filter out noisy clients or samples. However, in practical FL systems, there is no prior knowledge about the thresholds for different datasets training on different models, which makes them inapplicable.

Although these methods separately solve the noisy data problem and positively influential sample selection problem, they cannot distinguish positively influential clients and samples from noisy ones. To mitigate noisy labels, samples with small training losses are preferred as they are more likely to be clean data. To identify positively influential samples, those with high losses are preferred as they can induce large changes in model parameters, thereby accelerating model convergence. This important research gap inspires us to design a hard sample-aware noise-robust FL approach to efficiently and privately train high-performance models in the presence of training set noise.

3 PRELIMINARIES

3.1 Basics of Federated Learning

Let there be K FL clients $\{1, 2, \cdots, K\}$ each $k \in [K]$ holding a local dataset $D_k = \{z_{k,1}, z_{k,2}, \cdots, z_{k,n_k}\}$, consisting of n_k data samples, where $z_{k,i} = (x_{k,i}, y_{k,i}) \in \mathcal{X} \times \mathcal{Y}$. $N = \sum_{k=1}^{K} n_k$ is the total number of

all clients' samples. The goal of a standard federated optimization problem is to find:

$$\theta^* = \arg\min_{\theta} \sum_{k=1}^{K} \frac{n_k}{N} L_k(\theta), \text{where } L_k(\theta) = \frac{1}{n_k} \sum_{i=1}^{n_k} l(z_{k,i}; \theta), \tag{1}$$

where $l(z_{k,i}; \cdot)$, $L_k(\cdot)$ represent loss functions of a sample $z_{k,i}$, and of a client k on its local model.

This optimization problem can be solved via iterative stochastic optimization, *e.g.*, stochastic gradient decent (SGD). In the t-th iteration, given the fraction ϕ of FL clients to be selected, the FL server S randomly selects a subset U_t of $u = \max\{\lfloor \phi \cdot K \rfloor, 1\}$ clients and distributes the parameters of the current model θ_t to them. Each selected client $k \in U_t$ trains a local model $\theta_{t+1}^k = \theta_t - \eta_k \nabla L_k(\theta_t)$ with learning rate η_k, and sends θ_{t+1}^k to the FL server. The server aggregates the received updates from the selected clients and updates the global FL model as $\theta_{t+1} = \frac{1}{u} \sum_{k \in U_t} \theta_{t+1}^k$. This process is repeated until the global FL model converges (*i.e.*, a convergence criterion is met). Then, the global FL model θ^* is obtained.

3.2 Problem Formulation

There are two types of entities involved in a typical FL scenario: an FL server S, and K distributed clients $\{1, 2, \cdots, K\}$. Each client k possesses a local dataset D_k. Under the coordination of the server, all clients collaboratively train a global model $\theta^* \in \mathbb{R}^d$ by sharing their local models updated by their private datasets. Our goal is to construct high-performance FL models in the presence of noisy training data (*e.g.*, data noises, or data heterogeneity) which can lead to slow convergence speed and reduced prediction accuracy.

Specifically, we aim to enable FL participants to collaboratively select the most positively influential samples for model training, while solving training set noise problems by eliminating negatively influential samples. The goal of sample selection in FL is to dynamically select a subset of clients and their individual samples, and construct a global model θ^* based on the selected clients and samples. The selection of samples and clients, and choice of global parameters can be evaluated by minimizing the following risk:

$$\theta^*(P, p) = \arg\min_{\theta} \mathbb{E}_{k \sim P_k} \mathbb{E}_{z_{k,i} \sim p_{k,i}} [l(z_{k,i}; \theta)], \tag{2}$$

where $0 \leq P_k, p_{k,i} \leq 1, k \in [K]$ are the probabilities that client k and the i-th sample of client k are selected. We assume that all FL participants including the FL server are semi-honest (*i.e.*, they follow the FL protocol and the selection protocol faithfully, but may be curious about others' local private data).

We aim to design an FL model training framework to support collaborative client-sample selection and model updating to achieve the following design goals:

- **Effective Selection:** the framework shall accurately select important clients and their important individual local data samples which have large positive effects on the global model performance, and exclude negatively influential clients and samples from training. Based on the selection results, the framework shall improve model performance in terms of convergence speed and inference accuracy.
- **Privacy Preservation:** the framework shall preserve each data owner's data privacy. That is, the local training data or even data distribution shall not be exposed to any party other than their rightful owners during the selection and updating process.
- **Low Overhead:** considering the resource constraints of edge and mobile devices, the selection and updating process shall not incur high computation and communication cost.

4 THE PROPOSED APPROACH

In this section, we first present the detailed formulation of joint client-and-sample selection via federated bilevel learning (Section 4.1). Then, we show how to optimize the two nested optimization loops through federated meta-weight learning based on online approximation (Section 4.2). Finally, we describe the implementations of client and sample selection probability estimation and the procedures of influential client and sample selection and model updating (Section 4.3).

4.1 Sample Selection via FL Bilevel Learning

In the presence of training set noise, to achieve accurate client selection and sample selection while simultaneously training the global FL model, we need to dynamically quantify the influences of clients and samples on the global FL model during training, and increase the sampling probabilities for highly influential clients and their highly influential samples. Thus, we formulate the objective (Eq. (2)) into a bilevel optimization problem, with learning the optimal selection probabilities P^*, p^* as the outer problem and the optimal global FL model θ^* as the inner problem. Specifically, assume that there is a small clean validation set $D_v = \{z_i^v = (x_i^v, y_i^v) \in \mathcal{X} \times \mathcal{Y}\}, i \in [M]$, and $M \ll N$, stored at the FL server. Then, the objective in Eq. (2) can be re-expressed as:

$$\theta^*(P, p) = \arg\min_{\theta} \sum_{k=1}^{K} \frac{n_k}{N} P_k L_k(\theta), \text{where } L_k(\theta) = \frac{1}{n_k} \sum_{i=1}^{n_k} p_{k,i} l(z_{k,i}; \theta), \tag{3}$$

with P and p being unknown at beginning. The optimal selection of P and p are calculated based on the performance of the model on the validation set with the objective:

$$P^*, p^* = \arg\min_{P,p} \frac{1}{M} \sum_{i=1}^{M} l(z_i^v; \theta^*(P, p)), \tag{4}$$

where $l(z_i^v; \cdot)$ is the loss function of the validation sample z_i^v. This problem formulation is different from the existing work [43] since the goal of our work is to identify hard samples from noisy samples and construct a high-performance global model simultaneously, while [43] does not address the problem of sample selection and identifying noisy data. Besides, [43] assumes that each client has its own individual outer and inner functions and has its local clean validation datasets to guide local model update, while in our formulation, local clients only have model update functions and they have no access to the validation set.

However, calculating the optimal P^*, p^* and θ^* requires two nested optimizations, and each can be prohibitively resource intensive, especially for large-scale FL systems and resource-constrained FL clients. The motivation of our approach is to adapt online probabilities P, p through a single optimization loop. That is, for each training iteration, we inspect the descent direction of some clients and their training examples locally on the training loss surface and reweight them according to their similarity to the descent direction of the validation loss surface. Thus, each iteration involves computing the gradients $\nabla_P l(z_i^v; \theta(P, p)), \nabla_p l(z_i^v; \theta(P, p)), i \in [M]$. Taking the calculation of the gradient $\nabla_p l(z_i^v; \theta(P, p))$ of the validation loss w.r.t. the probability p as an example:

$$\nabla_p l(z_i^v; \theta(P, p)) = \nabla_\theta l(z_i^v; \theta(P, p))^\top \nabla_p \theta(P, p), \tag{5}$$

where $\nabla_p \theta(P, p)$ is a $d \times N$-size Jacobian matrix with $\theta \in \mathbb{R}^d$, $p \in \mathbb{R}^N$. The second term $\nabla_p \theta(P, p)$ in Eq. 5 can be derived through the implicit differentiation method [44], as illustrated below,

$$\nabla_p \theta(P, p) = -P_k \nabla_\theta l(z_{k,i}; \theta) \cdot \left(\frac{1}{K} \sum_{i=1}^{K} H_k \right)^{-1}, \tag{6}$$

where $H_k = \nabla^2_\theta \frac{1}{n_k} \sum_{i=1}^{n_k} l(z_{k,i}; \theta)$ is the Hessian matrix of client $k \in [K]$. The gradient $\nabla_P l(z_i^v; \theta(P, p))$, $i \in [M]$ can be calculated in a similar way.

Main Challenges. The online approximation method for bilevel FL optimization (Eq. (5)) enables us to learn the selection probabilities of all clients and their local samples so as to construct the optimal global model. However, evaluating Eq. 6 requires forming and inverting the Hessian of the loss function, which requires $O(Nd^2 + d^3)$ computational operations [31, 45] with N denoting the total number of training samples and $\theta \in \mathbb{R}^d$. Meanwhile, it also requires all clients to upload $H_k, k \in [K]$ to the FL server, which incurs $O(Kd^2)$ communication costs. Considering the large N, d typically involved in training deep FL models, the calculation of Eq. (5) incurs unacceptable computation and communication overhead. One similar work [43] in the bilevel optimization literature use implicit Hessian-vector products (HVPs) to approximate $(\frac{1}{K} \sum_{i=1}^{K} H_k)^{-1} \nabla_\theta l_i^v(\theta^*(P, p))$. However, with both outer functions and inner functions to learn in [43], clients would cost prohibitively high computation cost, *e.g.*, $O(Kd)$ operations, and more communication cost, *e.g.*, $O(Nd)$ parameters per training round as it requires transferring matrix-vector products for all of the training data at each FL training round. Besides, assuming that each client has its own clean validation set makes [43] unsuitable for practical FL applications.

Key Ideas. We design our system to significantly reduce the selection overhead with the following two approaches.

(1) Replacing 2nd-Order Calculations with 1st-Order Gradient-based Approximation. FedCSS leverages meta-learning based online approximation for FL bilevel optimization to iteratively update selection probabilities and the global FL model. In this way, the optimization can be efficiently conducted in conjunction with gradient approximation, thus avoiding expensive calculations of 2nd-order Hessian matrices.

(2) Reducing Unnecessary Influence Computation via Hierarchical Analysis. To calculate the selection probabilities of samples, each client needs to upload the gradients of all its samples to the server, which incurs $O(Nd)$ communication overhead. To improve efficiency and preserve privacy, we design a hierarchical selection approach. It first identifies positively influential clients. Then, it only requires these clients to update sample influences with the meta network, and select positively influential local samples to update the local models. In this way, FedCSS only needs to transfer the gradient of the validation data, the meta model, and client influence scores. The communication overhead is reduced to $O(d + d' + 1)$, where $d' \ll d$ is the size of the meta model.

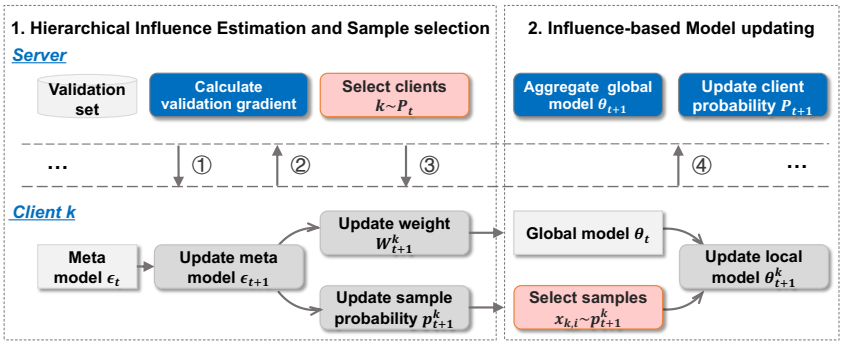
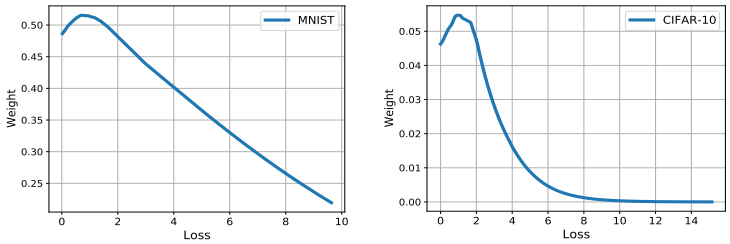


Fig. 1. System overview of FedCSS. ① meta model ϵ_t, global model θ_t and average gradient of validation data, ② meta model ϵ_t, ③ meta model ϵ_{t+1} and global model θ_t, ④ client weight W^k_{t+1} and local model θ^k_{t+1}.

The overall architecture of FedCSS is shown in Fig. 1. It consists of two main steps.



(a) MNIST (b) CIFAR-10

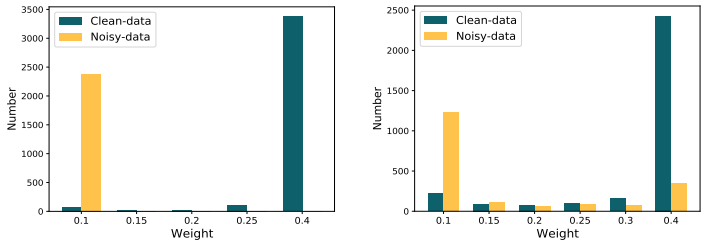
Fig. 2. Federated meta network functions learned by FedCSS on datasets with 40% mislabeling samples.

(1) **Hierarchical Influence Estimation and Sample Selection.** In each training iteration t, given the selection probabilities of FL clients, the FL server first selects a batch of clients to update their sample influences and participate in FL training. Each selected client updates the influence scores of its local training samples with the meta model based on the similarities between the descent direction of sample's loss surface and the validation loss surface. These influence scores are summed up and the average is taken as the influence score of the FL client. The clients then select their most positively influential local data samples to be used for FL model training.
(2) **Influence-based Model Updating.** With the influence estimation and client-and-sample selection results, each selected client updates its local model and sends the updated meta model, client weight (*i.e.*, client influence score) and updated local model to the FL server. The server then aggregates these local models to obtain the global FL model and updates client selection probabilities.

4.2 Meta-weight Learning for Bilevel FL Optimization

As we aim to obtain the optimal client and sample selection probabilities P^* and p^* that minimize the objective function in Eq. (3), we instead consider what the influence of each client and each training sample is on the validation set performance at training iteration t (*e.g.*, validation accuracy or validation loss). Then, the probabilities that clients and samples are selected are determined by their corresponding influence scores (*i.e.*, the probability shall be proportional to the influence). Further, inspired from existing works [31, 46, 47], which posit that the influence of a set is the sum of influences of its constituent samples, we define the influence score of client k as "the average of the influence scores of its local samples which are selected for FL training". In addition, since sample influence values are calculated based on the global model, which is obtained by aggregating local models, the relationships between clients and samples are implicitly embedded in the sample influence values and client influence values. These inspire us to design a hierarchical influence analysis method that identifies influential clients (i.e., influential subsets) first to save a significant portion of the cost incurred during the sample-level influence analysis stage.

To estimate the influence or the weight of each sample, we first denote the weight of sample $z_{k,i}$ as $w_{k,i}$, and the weight of client k is denoted as $W_k = \frac{1}{n_k}\sum_{i=1}^{n_k} w_{k,i}$ [47]. We then formulate the weights $w = \{w_1, \cdots, w_k\}$, where $w_k = \{w_{k,1}, \cdots, w_{k,n_k}\}$ as a meta network $w = f(l;\epsilon)$ (*e.g.*, Multi-layer Perceptron (MLP) networks with only one hidden layer), which takes the losses of samples as the inputs and outputs the corresponding weights w, where ϵ are parameters contained in them (see Fig. 2). To guarantee the output weights are within $[0, 1]$, the output passes through a Sigmoid activation function. At this time, the selection probabilities $P_k, p_{k,i}$ are equal to the weights $W_k, w_{k,i}$, respectively. Then, we focus on automatically learning the parameter ϵ through



(a) MNIST (b) CIFAR-10

Fig. 3. Weight distributions of FedCSS on randomly sampled batches from datasets with 40% mislabeling samples.

meta-learning:

$$\epsilon^* = \arg\min_{\epsilon} \frac{1}{M} \sum_{i=1}^{M} l(z_i^v; \theta^*(\epsilon)). \tag{7}$$

Updating FL Models. We employ SGD to optimize Eq. (7). Specifically, in the t-th training iteration, given the fraction ϕ of FL clients to be selected, and the weight distribution $W = \{W_t^1, \cdots, W_t^K\}$ of clients being selected, the FL server selects a subset U_t of $u \leftarrow \max\{\phi \cdot K, 1\}$ clients, and distributes the parameters of the current FL model θ_t to them. Each selected client $k \in U_t$ then selects a mini-batch of size $b_k \ll n_k$ according to the sample weight distribution $w_t^k = \{w_t^{k,1}, \cdots, w_t^{k,n_k}\}$, where $w_t^{k,i} = f(l(z_{k,i}; \theta_t), \epsilon_t), i \in [n_k]$, in each local epoch. Its local parameters θ_{t+1}^k are updated on the mini-batch which equals to the updates as follows:

$$\theta_{t+1}^k(\epsilon) = \theta_t(\epsilon) - \eta_k \nabla_\theta L_k(\theta)\Big|_{\theta_t}, \text{where } L_k(\theta) = \frac{1}{n_k} \sum_{i=1}^{n_k} f(l(z_{k,i}; \theta_t), \epsilon_t) l(z_{k,i}; \theta), \tag{8}$$

while the global FL model parameters are updated as:

$$\theta_{t+1}(\epsilon) = \frac{1}{u} \sum_{k \in U_t} \theta_{t+1}^k(\epsilon). \tag{9}$$

Then, the updated $\theta_{t+1}(\epsilon)$ is employed to update the parameter ϵ of the meta network by taking multiple gradient decent steps on a mini-batch of b validation samples w.r.t. ϵ_t, as:

$$\epsilon_{t+1} = \epsilon_t - \eta \frac{1}{b} \sum_{i=1}^{b} \nabla_\epsilon l(z_i^v; \theta_{t+1}(\epsilon))\Big|_{\epsilon_t}, \tag{10}$$

which, in turn, is used to update the sample and client weights and selection probabilities. η is the descent step size on ϵ.

Updating Federated Meta-Weight Models and Selection Probabilities. To estimate the weight ϵ_t and calculate the selection probabilities in the t-th iteration, the server needs to calculate the gradients of the validation loss w.r.t. the model ϵ:

$$\begin{aligned} &\frac{1}{b} \sum_{i=1}^{b} \nabla_\epsilon l(z_i^v; \theta_{t+1}(\epsilon))\Big|_{\epsilon_t} \\ &= \frac{1}{b} \sum_{i=1}^{b} \frac{\partial l(z_i^v; \theta)}{\partial \theta}\Big|_{\theta_{t+1}} \frac{1}{b_k} \sum_{j=1}^{b_k} \frac{\partial \theta_{t+1}(\epsilon)}{\partial f(l(z_{k,j}; \theta_t), \epsilon)} \frac{\partial f(l(z_{k,j}; \theta_t), \epsilon)}{\partial \epsilon}\Big|_{\epsilon_t}. \end{aligned} \tag{11}$$

Substitute Eq. (8) and Eq. (9) into the second term of Eq. (11), we have:

$$\begin{aligned}\frac{\partial \theta_{t+1}(\epsilon)}{\partial f(l(z_{k,j};\theta_t),\epsilon)}\bigg|_{\epsilon_t} =& \frac{\partial}{\partial f(l(z_{k,j};\theta_t),\epsilon)}\frac{1}{u}\sum_{k\in U_t}(\theta_t(\epsilon)-\eta_k\nabla_\theta L_k(\theta))\bigg|_{\epsilon_t}\\ =&-\eta_k\frac{1}{u}\sum_{k\in U_t}\frac{1}{n_k}\sum_{j=1}^{n_k}\frac{\partial l(z_{k,j};\theta)}{\partial\theta}\bigg|_{\theta_t}.\end{aligned} \tag{12}$$

Instead of calculating the above gradients on all samples, we leverage a batch of b_k samples from a randomly selected client k to estimate them. In this way, the gradient of the validation loss w.r.t. ϵ can be calculated as:

$$\begin{aligned}&\frac{1}{b}\sum_{i=1}^{b}\nabla_\epsilon l(z_i^v;\theta_{t+1}(\epsilon))\bigg|_{\epsilon_t}\\ &=-\frac{\eta_k}{b_k}\sum_{j=1}^{b_k}(\frac{1}{b}\sum_{i=1}^{b}\frac{\partial l(z_i^v;\theta)}{\partial\theta}\bigg|_{\theta_t}^{\top}\frac{\partial l(z_{k,j};\theta)}{\partial\theta}\bigg|_{\theta_t})\frac{\partial f(l(z_{k,j};\theta_t),\epsilon)}{\partial\epsilon}\bigg|_{\epsilon_t}.\end{aligned} \tag{13}$$

Let $\Gamma_{i,j}=\frac{\partial l(z_i^v;\theta)}{\partial\theta}\Big|_{\theta_t}^{\top}\frac{\partial l(z_{k,j};\theta)}{\partial\theta}\Big|_{\theta_t}$, ϵ_{t+1} can be updated as:

$$\epsilon_{t+1}=\epsilon_t-\eta\eta_k\frac{1}{b_k}\sum_{j=1}^{b_k}(\frac{1}{b}\sum_{i=1}^{b}\Gamma_{i,j})\frac{\partial f(l(z_{k,j};\theta_t),\epsilon)}{\partial\epsilon}\bigg|_{\epsilon_t}. \tag{14}$$

Then, the weights can be updated as $w_{t+1}^{k,i}=f(l(z_{k,i};\theta_t),\epsilon_{t+1})$, $W_{t+1}^k=\frac{1}{n_k}\sum_{i=1}^{n_k}w_{t+1}^{k,i}$; whereas the selection probabilities of sample $z_{k,i}$ and of client k can be updated as $p_{t+1}^{k,i}=w_{t+1}^{k,i}, P_{t+1}^k=W_{t+1}^k/\sum_{k=1}^{K}W_{t+1}^k$, which can be normalized.

Interpreting Selection Probability. From the above formulation, it can be observed that, the probability of a sample $z_{k,i}$ being selected in the t-th iteration is determined by the similarity between the gradient of the sample $z_{k,i}$ on the training loss and the average gradient of the mini-batch validation samples on the validation loss. This means if the gradient of a sample is similar to that of the validation samples, then it is regarded as a positively influential sample for improving the global FL model. Thus, its selection probability shall be increased. Conversely, the selection probability of the client shall be reduced (see Fig. 3(b)). This corroborates with findings in [48].

Estimation Overhead Analysis. Computing the sampling distribution from Eq. (14) is more than an order of magnitude faster compared to computing the second-order derivative for each sample. Specifically, in each iteration, each selected client optimizes the meta-weight network with $O(d)$ operations in contrast with the original $O(Nd^2+d^3)$ operations as illustrated in Section 4.1. Meanwhile, the communication costs are only $O(d+d')$, $d'\ll d$ parameters for each selected client, in contrast with the original $O(Kd^2)$ parameters. For the sample influence score calculation, each client locally conducts the forward prorogation to output the influence score, which only introduces a constant factor of computation cost and no communication cost. Thus, in each iteration, the computation cost of FedCSS is $O(\mu d+N)$, while the communication cost is $O(\mu(d+d'))$.

4.3 Implementation

Joint Client and Sample Selection. The proposed FedCSS approach is illustrated in Algorithm 1. Specifically, the FL server initializes a global model θ_0 and a meta model ϵ_0. The K clients and their local samples start with the same initial selection probability $P_0^k=\frac{1}{K}, p_0^{k,i}=\frac{1}{n_k}, k\in[K], i\in[n_k]$. In the t-th iteration, the server randomly selects a client k, and sends the global model θ_t, meta

model ϵ_t and the average gradient g_t^v of the batch validation samples to it. Client k randomly selects a batch of b_k samples and updates the meta model ϵ_t following Eq. (14) (Line 15), and sends it to the server. The server updates the meta model ϵ_{t+1}, and selects u clients (forming a selected client set U_t) according to their selection probabilities, and distributes the global model θ_t and meta model ϵ_{t+1} to them. Each selected client $k \in U_t$ updates its sample selection probabilities and local model as follows. Firstly, it updates the sample weight w_{t+1}^k through $w_{t+1}^k = f(l(z_{k,i}; \theta_t))$, and calculates probability distribution $\{p_{k,1}, p_{k,2}, \cdots, p_{k,n_k}\}$ for all its samples. Then, client k selects a batch of b_k samples with the probability distribution $\{p_{k,1}, p_{k,2}, \cdots, p_{k,n_k}\}$, and updates the local model over multiple local training steps. Finally, client k updates its weight W_{t+1}^k and sends the local model $\theta_t^{k,E}$ and weight W_{t+1}^k to the server.

Influence-based Model Updating. The server aggregates local models to obtain the global FL model $\theta_{t+1} = \frac{1}{u} \sum_{k \in [U_t]} \theta_t^{k,E}$ (Line 10), and updates the client selection probabilities by normalizing

Algorithm 1: FedCSS

Input : Initial selection probability $P_0^k = \frac{1}{K}$, $p_0^k = \frac{1}{n_k}$, $k \in [K]$, $i \in [n_k]$, validation set D_v

Output : The optimal global model θ^*

1 //At the FL Server:

2 Initialize global model θ_0, meta model ϵ_0;

3 **for** *each round* $t \in \{0, 1, \cdots, T\}$ **do**

4 $g_t^v = \eta \frac{1}{b} \sum_{i=1}^{b} \frac{\partial l(z_i^v; \theta_t)}{\partial \theta}$;

5 $k \leftarrow$ randomly select a client;

6 $\epsilon_{t+1} \leftarrow$ `LocalMeta`$(k, \theta_t, \epsilon_t, g_t^v)$; // update meta model

7 $U_t \leftarrow$ select u clients $k \sim P_t$;

8 **for** *each client* $k \in U_t$ **do**

9 $\theta_{t+1}^k, W_{t+1}^k \leftarrow$ `LocalUpdate`$(k, \theta_t, \epsilon_t)$;

10 $\theta_{t+1} = \frac{1}{u} \sum_{k \in U_t} \theta_{t+1}^k$; // update global model

11 $P_{t+1}^k = \frac{W_{t+1}^k}{\sum_{k=1}^{K} W_{t+1}^k}, \forall k \in [K]$;

12 //At FL Client $k \in [K]$:

13 **Function** `LocalMeta`$(k, \theta_t, \epsilon_t, g_t^v)$**:**

14 Randomly select a batch of b_k samples

15 $\epsilon_{t+1}^k = \epsilon_t - \eta_k \frac{1}{b_k} \sum_{j=1}^{b_k} g_t^v \frac{\partial l(z_{k,j}; \theta_t)}{\partial \theta} \frac{\partial f(l(z_{k,j}; \theta_t), \epsilon_t)}{\partial \epsilon}$;

16 **Return** ϵ_{t+1}^k;

17 **Function** `LocalUpdate`$(k, \theta_t, \epsilon_t)$**:**

18 $\theta_t^{k,0} = \theta_t$;

19 $w_{t+1}^{k,i} = f(l(z_{k,i}; \theta_t), \epsilon_{t+1}^k), W_{t+1}^k = \frac{1}{n_k} \sum_{i=1}^{n_k} w_{t+1}^{k,i}$; // Update sample probabilities

20 $p_{t+1}^{k,i} \leftarrow w_{t+1}^{k,i}, \forall i \in [n_k]$;

21 **for** *each local epoch* j *from 1 to* E **do**

22 Select a batch of b_k samples $x_{k,i} \sim p_{t+1}^k$;

23 $L_k(\theta_t^{k,j}) = \frac{1}{b_k} \sum_{i=1}^{b_k} l(z_{k,i}; \theta_t^{k,j-1})$;

24 $\theta_t^{k,j} = \theta_t^{k,j-1} - \eta_k \nabla_\theta L_k(\theta_t^{k,j-1})$; // Update local models

25 **Return** $\theta_t^{k,E}, W_{t+1}^k$;

the weights of all clients so that they sum to 1. This way, the server and FL clients collaboratively select the most positively influential samples to construct the global FL model θ^*.

5 ANALYTICAL EVALUATION

Here, we first analyze the privacy property of FedCSS. Then, we show that, under mild conditions, FedCSS achieves a convergence rate of $O(1/\varepsilon^2)$, where $0 \leqslant \varepsilon \leqslant 1$ is a constant.

5.1 Privacy Preservation

FedCSS preserves each client's local training data from exposure to other parties, including the FL server, during the selection and updating processes. Firstly, the training process of FL models and meta models follows the standard FL training protocol. Hence, no local data are transmitted during training [1]. Secondly, during the sample selection process, no local training data are transmitted. During the client selection process, the clients transmit the weights (influence scores) to the FL server without revealing the raw data or the loss distribution of individual samples. In scenarios where even a client's weight raises privacy concerns, clients can add noise to their weight values before uploading them, similar to existing local differential privacy (DP) approaches [49].

5.2 Convergence Analysis

We present convergence analysis results for FedCSS. For simplicity, we denote the loss function $l(z_{k,i};\theta)$ of sample $z_{k,i}$ as $l_{k,i}$, and $l(z_i^v;\theta)$ of validation sample z_i^v as l_i^v, respectively.

Assumption 1. *The validation loss functions are β-Lipschitz continuous with $||\nabla l_i^v - \nabla l_j^v|| \leqslant \beta||z_i^v - z_j^v||, \forall z_i^v, z_j^v \in D_v$.*

Assumption 2. *The loss function $L_k, k \in [K]$ has σ-bounded gradients with $||\nabla L_k(\theta)|| \leqslant \sigma$.*

FedCSS leverages useful information from both the training set and validation set, and converges to an appropriate distribution favored by the validation set. This helps it achieve improved generalization and robustness against biases in the training set (as illustrated in our experiments).

Lemma 1. *(Convergence of FedCSS.) Under the above assumptions, and let the learning rate η_k for client k satisfies $\eta_k \leqslant \frac{2b_k}{\beta\sigma^2}$, where b_k is the training batch size, the validation loss monotonically decreases for any sequence of training batches, $L_v(\theta_{t+1}) \leqslant L_v(\theta_t)$, where $L_v(\theta_t) = \frac{1}{M}\sum_{i=1}^{M} l_i^v(\theta_t(P,p))$. In expectation, the equality holds only when the gradient $\nabla L_v(\theta_t) = 0$, where the expectation is computed over all batches in the t-th iteration.*

Proof. Since $M \ll N$, we assume that the validation dataset is a subset of the training dataset, and is distributed to client k. During training, client k takes a mini-batch of b_k training samples at each iteration. Then, the local parameters θ_{t+1}^k are updated as:

$$\theta_{t+1}^k = \theta_t - \eta_k \frac{1}{b_k} \sum_{i=1}^{b_k} \max\{\nabla L_v^\top \nabla l_{k,i}, 0\} \nabla l_{k,i}. \tag{15}$$

The FL server aggregates updates from local clients, and applies the update $\theta_{t+1} = \frac{1}{u}\sum_{k\in U_t} \theta_{t+1}^k$. Since all gradients are taken from θ_t, we omit θ_t in our notations. Since the validation loss $L_v(\theta)$ is β-Lipschitz continuous, we have,

$$L_v(\theta_{t+1}) \leqslant L_v(\theta_t) + \nabla L_v^\top \Delta\theta + \frac{\beta}{2}||\Delta\theta||^2 \tag{16}$$

Substituting Eq. (15) into Eq. (16), we have:

$$L_v(\theta_{t+1}) \leqslant L_v(\theta_t) - J_1 + J_2, \tag{17}$$

where,

$$\begin{aligned} J_1 &= \frac{1}{u}\sum_{k\in U_t}\eta_k\frac{1}{b_k}\sum_{i=1}^{b_k}\max\{\nabla L_v^\top \nabla l_{k,i},0\}\nabla L_v^\top \nabla l_{k,i} \\ &= \frac{1}{u}\sum_{k\in U_t}\eta_k\frac{1}{b_k}\sum_{i=1}^{b_k}\max\{\nabla L_v^\top \nabla l_{k,i},0\}^2, \end{aligned} \tag{18}$$

and,

$$\begin{aligned} J_2 &= \frac{\beta}{2}\left\|\frac{1}{u}\sum_{k\in U_t}\eta_k\frac{1}{b_k}\sum_{i=1}^{b_k}\max\{\nabla L_v^\top \nabla l_{k,i},0\}\nabla l_{k,i}\right\|^2 \\ &\leqslant \frac{\beta}{2u}\sum_{k\in U_t}\frac{\eta_k^2}{b_k^2}\sum_{i=1}^{b_k}\max\{\nabla L_v^\top \nabla l_{k,i},0\}^2\sigma^2. \end{aligned} \tag{19}$$

If we denote $\alpha_t = \sum_{i=1}^{b_k}\max\{\nabla L_v^\top \nabla l_{k,i},0\}^2$ for the t-th iteration, then:

$$L_v(\theta_{t+1}) \leqslant L_v(\theta_t) - \frac{1}{u}\sum_{k\in U_t}\frac{\eta_k}{b_k}\alpha_t\left(1-\frac{\beta\eta_k\sigma^2}{2b_k}\right). \tag{20}$$

Note that by definition, α_t is non-negative. Since $\eta_k \leqslant \frac{2b_k}{\beta\sigma^2}$, it follows that $L_v(\theta_{t+1}) \leqslant L_v(\theta_t)$ for any t. Then, we prove that $\mathbb{E}_t[\alpha_t] = 0$ if and only if $\nabla L_v = 0$, and $\mathbb{E}_t[\alpha_t] > 0$ if and only if $\nabla L_v \neq 0$, where the expectation is computed over all training batches in the t-th iteration. It is obvious that when $\nabla L_v = 0$, $\mathbb{E}_t[\alpha_t] = 0$. If $\nabla L_v \neq 0$, we observe that there exists a validation sample $z_j^v, j \in [M]$ such that $\nabla L_v^\top \nabla l_j^v > 0$ as follows:

$$0 < ||\nabla L_v||^2 = \nabla L_v^\top \nabla L_v = \frac{1}{M}\sum_{i=1}^{M}\nabla L_v^\top \nabla l_j^v. \tag{21}$$

Then, there exists a none-zero possibility p to sample a batch B_k that contains data z_j:

$$\mathbb{E}_t[\alpha_t] \geqslant p\sum_{i\in B_k}\max\{\nabla L_v^\top \nabla l_i^v,0\}^2 \geqslant p\{\nabla L_v^\top \nabla l_j^v,0\}^2. \tag{22}$$

Therefore, if we take expectation over the batch on both sides of Eq. (20), we can conclude that:

$$\mathbb{E}_t[L_v(\theta_{t+1})] \leqslant L_v(\theta_t), \tag{23}$$

where the inequality holds if and only if $\nabla L_v = 0$. Thus, we have proven the convergence of FedCSS. □

6 EXPERIMENTAL EVALUATION

In this section, we first evaluate the effectiveness of FedCSS by comparing it against state-of-the-art baselines in terms of model accuracy and training overhead. Then, we provide analysis on how FedCSS learns robust FL models.

6.1 Experimental Settings

Implementation. We implemented FedCSS and in an FL system consisting of one server and 10 clients. To further investigate the performance of FedCSS in large-scale FL systems, we also tested it in an environment with up to 1,000 clients. Our implementation is based on Python 3.9 and Pytorch 1.8.1 [50]. All the experiments are performed on Ubuntu 16 operating system equipped with two 20-core Intel (R) Xeon (R) CPU, 512G of RAM and 8 Tesla V100 GPUs.

Datasets. As presented in Table 1, we use five real-world datasets of different scales and different modalities, including image data and text data, for our FL tasks.

- **MNIST** [51]: It is used for handwritten recognition task with digits between 0 and 9.
- **FEMNIST** [52]: This dataset consists of a federated version of the MNIST, maintained by the LEAF project [53]. It has 62 different classes and contains a total of 817,851 images.
- **CIFAR-10** [54]: It consists of 60,000 colour images in 10 classes, with 6,000 images per class.
- **CIFAR-100** [55]: This dataset is a subset of the Tiny Images dataset and consists of 60,000 color images in 100 classes.
- **AGNews** [56]: This dataset is a collection of news articles which have been gathered from more than 2000 news sources.

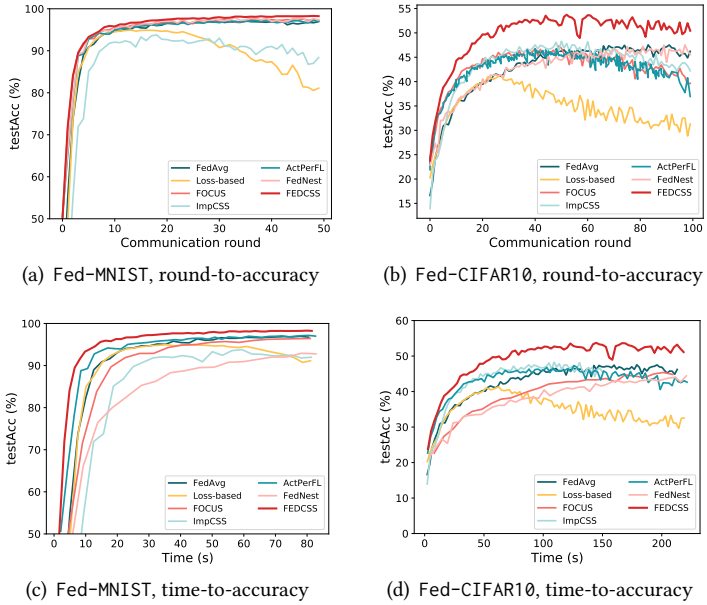
For these datasets, we randomly select 1,000 instances from the training sets as the validation sets without introducing new data.

Table 1. Datasets for different FL tasks.

Modality	Dataset	Notation	Size	Description
Image	MNIST	D_M	60,000	original training data of MNIST
		D_M^m	60,000	D_M with 10%-70% mislabeled samples
		D_M^v	1,000	images randomly select from D_M
		D_M^I	10,000	original testing data of MNIST
	FEMNIST	D_F	736,066	original training data of FEMNIST
		D_F^m	736,066	D_F with 10%-70% mislabeled samples
		D_F^v	1,000	images randomly select from D_F
		D_F^I	81,785	original testing data of FEMNIST
	CIFAR-10	D_C	50,000	original training data of CIFAR-10
		D_C^m	50,000	D_C with 10%-70% mislabeled samples
		D_C^v	1,000	images randomly select from D_C
		D_C^I	10,000	original testing data of CIAFR-10
	CIFAR-100	D_I	50,000	original training data of CIFAR-100
		D_I^m	50,000	D_I with 10%-70% mislabeled samples
		D_I^v	1,000	images randomly select from D_I
		D_I^I	10,000	original testing data of CIFAR-100
Text	AGNews	D_A	120,000	original training data of AGNews
		D_A^m	120,000	D_A with 10%-70% mislabeled samples
		D_A^v	1,000	images randomly select from D_A
		D_A^I	7,600	original testing data of AGNews

Label Noise. We take the corrupted mislabeled data as the training set noise. Specifically, given a mislabeling ratio r, the label of each sample is independently changed to a random class with probability $(1 - r)$ following the same setting in [57].

Data Distribution. We partitioned datasets D_M, D_M^m over 10 clients in both iid (independent and identically distributed) and non-iid (non-independent and identically distributed) settings following the setting in [1]. We divided images of sorted digits into 600 shards of size 100 and assigned each client 60 shards. When the 60 shards contain images of 10 different digits, this is



(a) Fed-MNIST, round-to-accuracy (b) Fed-CIFAR10, round-to-accuracy

(c) Fed-MNIST, time-to-accuracy (d) Fed-CIFAR10, time-to-accuracy

Fig. 4. Test accuracy of different communication rounds and different time periods for training models.

referred to as the iid setting; otherwise, non-iid one. Specifically, according to the mislabelled data distribution, we consider two iid settings (*e.g.*, iid (1), iid (2)) and two non-iid settings. They are: 1) same noisy data distributions, in which we distribute the mislabeled data samples randomly to all clients for both iid and non-iid settings; and 2) different noisy data distributions, in which we distribute the mislabeled data samples randomly to five clients for both iid and non-iid settings. We denote non-iid [c], *e.g.*, $1 \leq c \leq 10$, to indicate that each client has samples from c random categories out of all ten categories. We set $c = 0.6 * C$ where C is the number of categories, and further evaluate how FedCSS behaves as c varies in Section 6.3. We partitioned datasets D_F^m, D_C^m, D_I^m, D_A^m in both iid and non-iid settings as for dataset D_M^m (see Table 2). The validation sets D_M^v, D_F^v, D_C^v, D_I^v, D_A^v and testing sets D_M^T, D_F^T, D_C^T, D_I^T, D_A^v are located at the server.

Table 2. Experiment settings for training different models.

Model	Dataset	Setting	Noise ratio
Fed-MNIST	D_M^m	iid (1)/ non-iid (1)	40%
		iid (2)/ non-iid (2)	30%
Fed-FEMNIST	D_F^m	iid (1)/ non-iid (1)	40%
		iid (2)/ non-iid (2)	30%
Fed-CIFAR10	D_C^m	iid (1)/ non-iid (1)	40%
		iid (2)/ non-iid (2)	30%
Fed-CIFAR100	D_I^m	iid (1)/ non-iid (1)	20%
		iid (2)/ non-iid (2)	15%
Fed-AGNews	D_A^m	iid (1)/ non-iid (1)	40%
		iid (2)/ non-iid (2)	40%

FL Models. We have implemented FedCSS (Algorithm 1) and four residual networks based on ResNet-32 [58]: Fed-MNIST, Fed-FEMNIST, Fed-CIFAR10 and Fed-CIAFR100 for classifying images in MNIST, FEMNIST, CIFAR-10 and CIFAR-100, and a neural network [59]: Fed-AGNews for

Table 3. Test accuracy comparison under different settings for different datasets.

Dataset	Setting	Test accuracy (%) ± standard deviations						
		FedAvg	Loss-based	FOCUS	ImpCSS	ActPerFL	FedNest	**FedCSS**
D_M^m	iid (1)	96.95± 0.27	94.98± 0.03	97.23± 0.05	93.76± 0.32	97.11 ± 0.13	96.25± 0.28	**98.28± 0.11**
	iid (2)	96.70± 1.02	93.82± 0.05	94.82± 0.11	94.2± 2.52	95.41± 0.19	95.32± 0.23	**98.36± 0.02**
	non-iid (1)	96.01± 0.34	89.98± 0.69	96.23± 0.12	94.43± 0.57	95.51± 0.06	94.78± 0.18	**98.23± 0.09**
	non-iid (2)	93.07± 0.34	81.42± 0.32	95.13± 0.13	93.42± 0.37	95.32± 0.27	93.61± 0.16	**98.03± 0.18**
D_F^m	iid (1)	87.58± 0.21	86.23± 0.22	88.04± 0.16	87.31± 0.24	86.83± 0.17	86.62± 0.03	**88.42± 0.16**
	iid (2)	88.40± 0.62	87.79± 0.12	88.23± 0.13	87.52± 1.27	86.12± 0.12	87.01± 0.08	**88.62± 0.12**
	non-iid (1)	84.82± 0.41	75.51± 1.18	84.31± 0.21	81.23± 0.17	83.54± 0.27	84.05± 0.12	**86.13± 0.14**
	non-iid (2)	83.10± 0.41	75.05± 1.18	84.31± 0.21	82.12± 0.17	82.89± 0.72	83.19± 0.13	**85.51± 0.14**
D_C^m	iid (1)	47.99± 0.83	39.49± 0.42	46.26± 1.83	48.06± 1.30	45.88± 0.19	46.82± 0.31	**53.73± 0.57**
	iid (2)	44.01± 3.98	41.53± 3.06	50.52± 1.32	51.21± 1.22	51.76 ± 0.08	50.61± 0.27	**57.63± 1.12**
	non-iid (1)	41.29± 1.21	36.79± 1.51	43.69± 1.51	36.73± 0.24	44.01 ± 0.16	43.91± 0.81	**45.60± 0.45**
	non-iid (2)	39.01± 2.34	29.33± 1.69	39.23± 1.13	39.10± 2.07	39.98± 0.72	38.92± 0.24	**40.65± 0.12**
D_I^m	iid (1)	26.12± 0.27	17.61± 0.21	27.03± 0.16	27.18± 0.56	25.08± 0.28	26.91± 0.71	**33.51± 0.35**
	iid (2)	27.27± 0.48	17.97± 0.26	27.87± 0.34	29.87± 0.32	27.85± 0.21	28.01± 0.17	**35.06± 0.23**
	non-iid (1)	22.41± 1.22	14.69± 1.52	23.12± 2.32	24.17± 1.38	24.04± 0.49	24.56± 0.29	**28.73± 1.31**
	non-iid (2)	25.01± 1.34	16.11± 2.69	24.83± 1.03	25.42± 1.27	25.57±0.35	23.86± 0.12	**28.27± 1.07**
D_A^m	iid (1)	82.33± 0.17	81.61± 0.21	**84.51± 0.07**	82.75± 0.56	84.01± 0.08	83.46± 0.26	84.09± 0.05
	iid (2)	81.27± 0.18	81.57± 0.62	81.87± 0.14	80.87± 0.12	81.19± 0.16	80.86± 0.29	**82.26± 0.21**
	non-iid (1)	81.41± 0.22	82.67± 0.31	81.12± 0.32	81.17± 0.38	82.67± 0.18	80.26± 0.19	**82.73± 0.31**
	non-iid (2)	79.71± 0.41	80.14± 0.61	**80.83± 0.09**	79.42± 0.27	79.96± 0.17	78.76± 0.28	80.27± 0.27

Fig. 5. Confusion matrices of Fed-CIFAR10.

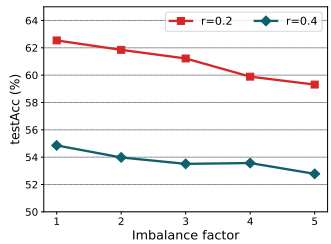


Fig. 6. Test accuracy with imbalanced validation set.

classifying texts in dataset AGNews. During model training, we set the fraction of selected clients to $\phi = 0.4$, each client has a learning rate of $\eta = 0.01$ (without learning rate decay), a momentum of 0.5 for D_M^m, D_C^m in iid setting, 0.9 in non-iid setting, and 0.9 for D_F^m, D_I^m, and batch size of 100. We conduct FL until a pre-specified test accuracy is reached, or a maximum number of iterations has elapsed (*e.g.*, 100 rounds for D_M^m, 200 rounds for D_C^m, D_A^m, and 500 rounds for D_F^m, D_I^m). For all experiments, we perform 5-fold cross validation and report the average results.

Comparison Baselines. We compare FedCSS against six state-of-the-art baselines, including:

(1) **FedAvg** [1]: FedAvg with random client and sample selection.
(2) **Loss-based**: FedAvg with client selection that prioritizes clients with high losses, which is similar with existing works [17, 24].
(3) **FOCUS** [60]: It performs credit weighted FL model aggregation to mitigate label noise by assigning credibility values to clients.
(4) **ImpCSS**: FedAvg with sample-level data selection that prioritizes samples with high importance derived from sample losses [9, 10].
(5) **ActPerFL** [61]: an active personalized FL solution that guides local training and global aggregation via inter- and intra-client uncertainty quantification.

(6) **FedNest** [43]: Federated bilevel optimization with HVP-based approximations.

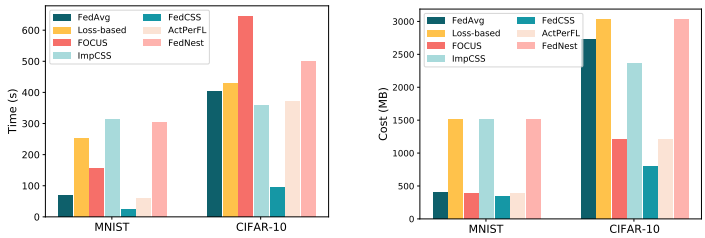
All methods are evaluated in terms of global model test accuracy, the average computation and communication costs.

6.2 Results and Discussion

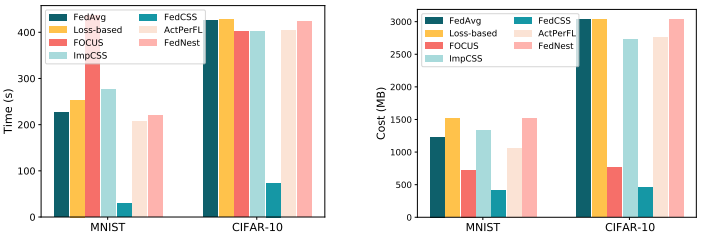
FedCSS improves accuracy. We compare FedCSS with others by training different FL models, and evaluating the test accuracy of the global models under both iid and non-iid settings (see Table 2). The test accuracy of different communication rounds, *i.e.*, round-to-accuracy, and different time periods for datasets D_M^m, D_C^m are shown in Fig. 4. The test accuracy of different communication rounds and different time periods for datasets D_F^m, D_I^m, D_A^m are quite similar to that in Fig. 4. It shows that FedCSS outperforms all other methods in terms of both test accuracy and convergence speed in all scenarios. For example, in the 100-th communication round of training the `Fed-CIFAR10` model on dataset D_C^m in the iid (1) setting, the test accuracy of FedCSS is 53.20%, outperforming the best baseline FOCUS by 6.99%. Besides, we note that the Loss-based and ImpCSS methods are much more vulnerable to noisy samples, because these two methods prioritize clients and samples with large losses, whereas such clients and samples can be erroneous ones. Although these methods filter out noisy samples by setting loss thresholds, those thresholds are hard to determine in practice due to *i.e.*, different proportions of noisy data and different data distributions among clients.

We further present the test accuracy of global models and the standard deviations in Table 3. Here, we adopt the early stop strategy to keep track of the prediction accuracy tested on the test dataset (*i.e.*, we terminate the procedure when the accuracy stops increasing for 10 consecutive epochs). The reason of using the early stop strategy is as follows. From Fig. 4, we can observe that some baselines can easily overfit the training noise, while FedCSS does not. That is to say, some baselines fail to converge in the presence of training noise. To provide a fairer comparison, we use the early stop strategy for both FedCSS and the baseline methods. The results in Table 3 shows that FedCSS achieves higher test accuracy than all other baseline methods. As an example, for model `Fed-CIFAR10` in iid (1) setting, the average test accuracy of FedCSS is 5.67% higher than the best performing baseline. Further, we evaluate the test accuracy of the final global models using FedCSS for all individual categories. We present the confusion matrix results of `Fed-CIFAR10` in Fig.5, which shows that FedCSS achieves high test accuracy among all categories.

FedCSS improves efficiency. To clearly illustrate that FedCSS achieves faster convergence, we present the average computation cost and communication cost for each local client when training FL models using different methods. Similarly, we adopt the early stop strategy under both iid and non-iid settings. That is to say, we run FL either until a pre-specified test accuracy is reached (96.0% for D_M^m, 80.0% for D_F^m, 47.0% for D_C^m, 25.0% for D_I^m and 80.0% for D_A^m) for 10 consecutive communication rounds, or until a maximum number of rounds (*e.g.*, 100 rounds for D_M^m, 200 rounds for D_C^m and D_A^m, and 500 rounds for D_F^m, D_I^m) have elapsed. Fig. 7 presents the example results of the average computation cost and communication cost of each client for training `Fed-MNIST` and `Fed-CIFAR10` under iid settings. The average computation cost and communication cost for training `Fed-FEMNIST` and `Fed-CIFAR100` are quite similar to that in Fig. 7. The results in Fig. 7 show that FedCSS achieves significantly higher saving of computation and communication costs than others. As an example, for model `Fed-CIFAR10` training on D_C^m in the setting of iid (2), the runtime of FedCSS is only 73.7s, which is 1.71× faster runtime than the best performing baseline. The communication cost of FedCSS is only 455.3 MB, which is 40.1% lower than the best performing baseline. The reasons for the faster convergence of FedCSS are as follows. Since some baseline methods can easily overfit the training noise, the training is unstable and the model converges

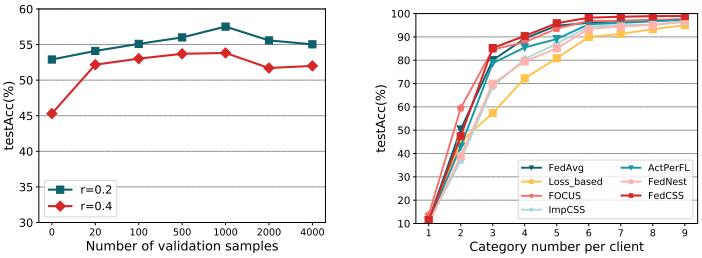


(a) Computation cost, iid (1) (b) Communication cost, iid (1)



(c) Computation cost, iid (2) (d) Communication cost, iid (2)

Fig. 7. The average computation and communication cost for each client for training models under iid settings.



(a) Number of validation samples (b) Non-iid degree

Fig. 8. Sensitivity analysis of test accuracy trained on different models under various factor settings.

slowly. In contrast, through selecting important clients and samples and filtering noisy ones, FedCSS achieves much faster convergence speed and reduces computation and communication cost.

Working mechanism of FedCSS. It is beneficial to understand how FedCSS learns robust FL models. We use a pre-trained model (trained at half of the total rounds) and measure the example weight distribution of a randomly selected client with mislabeled samples on training images. As shown in Fig. 3, FedCSS can correctly distinguish clean and noisy samples. Most of the large weights belong to clean samples, while noisy sample weights are much smaller than that of clean samples. FedCSS selects clean samples with higher probabilities than noisy samples for FL training.

6.3 Sensitivity Analysis

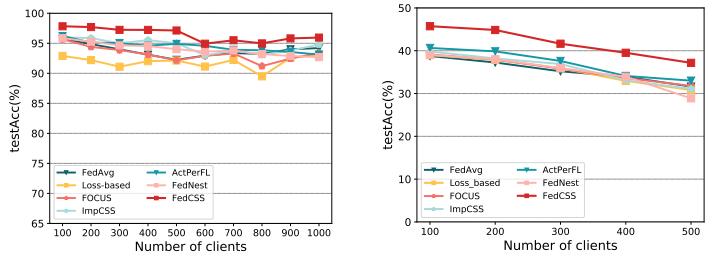
Impact of noise ratio r. We investigate how robust FedCSS is against a variety of noise levels from 0.1 to 0.7. We train `Fed-MNIST`, `Fed-CIFAR10` on D_M^m and D_C^m with different ratios of mislabeled samples among all clients. The results are shown in Table 4. It can be observed that FedCSS achieves the highest test accuracy in almost all cases. For instance, with a noise ratio of 0.1, the test accuracy of `Fed-CIFAR10` trained with FedCSS is 4.32% higher than the best performing baseline. For the

Table 4. Test accuracy comparison under different datasets with various noise ratios.

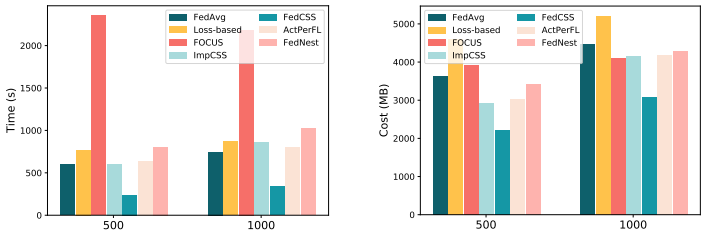
Dataset	Method	Test accuracy (%) ± standard deviations						
		0.1	0.2	0.3	0.4	0.5	0.6	0.7
D_M^m	FedAvg	98.20± 0.07	97.80± 0.02	97.22± 0.21	97.10± 0.34	96.29± 0.05	93.85± 0.55	88.69± 3.67
	Loss-based	96.64± 0.01	95.28± 0.01	95.01± 0.01	94.97± 0.02	91.72± 0.02	90.65± 0.03	82.66± 0.32
	FOCUS	**98.48± 0.04**	98.11± 0.08	97.51± 0.06	97.23± 0.05	95.86± 0.04	93.08± 0.11	88.21± 0.12
	ImpCSS	96.07± 0.01	94.64± 0.23	94.0± 0.21	93.78± 0.32	92.04± 0.75	89.18± 1.64	83.43± 1.22
	ActPerFL	98.27± 0.08	97.85± 0.13	97.34± 0.12	97.11± 0.13	96.15± 0.15	93.12± 0.34	89.13± 0.32
	FedNest	97.37± 0.13	96.89± 0.12	96.61± 0.11	96.25± 0.28	93.56± 0.15	90.12± 0.27	88.47± 0.27
	FedCSS	98.38± 0.08	**98.43± 0.06**	**98.19± 0.06**	**98.24± 0.17**	**98.0± 0.03**	**97.79± 0.04**	**96.73± 0.05**
D_C^m	FedAvg	56.46± 0.84	55.6± 1.56	51.52± 0.90	47.99± 0.83	41.52± 0.26	34.34± 0.50	31.12± 0.82
	Loss-based	46.63± 0.93	44.73± 0.14	43.23± 1.26	39.49± 0.42	36.54± 0.35	33.60± 2.07	29.61± 0.93
	FOCUS	55.42± 0.56	53.02± 0.78	50.03± 0.57	46.16± 0.83	42.57± 0.34	37.47± 0.42	30.04± 0.65
	ImpCSS	55.41± 0.80	52.63± 0.50	49.39± 0.56	48.06± 1.30	43.35± 1.91	41.1± 0.06	32.14± 0.58
	ActPerFL	53.12± 0.12	50.15± 0.15	47.32± 0.26	45.88± 0.19	43.05± 1.01	41.3± 0.16	31.87± 0.43
	FedNest	54.47± 0.18	50.62± 0.16	48.32± 0.16	46.82± 0.31	44.15± 0.51	41.24± 0.16	33.45± 0.62
	FedCSS	**62.05± 0.11**	**59.53± 1.00**	**57.01± 0.04**	**53.73± 0.57**	**50.38± 0.01**	**46.56± 1.20**	**40.66± 0.68**

Table 5. Test accuracy comparison on imbalanced classes.

Imbalance	Test accuracy (%) ± standard deviations						
	FedAvg	Loss-based	FOCUS	ImpCSS	ActPerFL	FedNest	**FedCSS**
20	87.85± 0.19	70.92± 1.92	90.63± 0.23	86.62± 0.24	88.98± 0.21	87.86± 0.08	**96.95± 0.45**
15	88.92± 0.85	80.02± 0.29	91.33± 0.24	88.01± 0.73	91.02± 0.09	89.67± 0.13	**95.23± 0.63**
10	90.01± 0.51	87.61± 0.62	93.16± 0.73	91.17± 0.58	92.13± 0.23	91.20± 0.27	**96.59± 0.52**
5	92.37± 0.42	91.28± 0.38	94.89± 0.43	93.08± 0.46	94.78± 0.12	93.86± 0.18	**98.05± 0.23**



(a) Test accuracy, Fed-MNIST (b) Test accuracy, Fed-CIFAR10



(c) Computation cost, Fed-MNIST (d) Communication cost, Fed-MNIST

Fig. 9. The performance of models trained on datasets D_M^m, D_C^m with mislabeling ratios of 40% for a large number of clients.

only case where FedCSS performs second best with the ratio being 0.1 on Fed-MNIST, in which the performance of FedCSS is comparable to that of FOCUS. In addition, the test accuracy of FedCSS only drops 1.65% when the noise ratio is increased from 0.1 to 0.7 of D_M^m, whereas that of the other

six methods dropped by between 6.31% to 12.54%. In summary, FedCSS is highly robust against different noisy data ratios.

Impact of Imbalanced distribution. We investigate how FedCSS behaves under the imbalanced distributions of both the training set and the validation set. We use MNIST dataset and subsample it to generate a class-imbalanced dataset that reduces the number of training samples per class according to an exponential function $n = n_i v_i$, where i is the class index, n_i is the original number of training images of class i and $v_i \in [0, 1]$. The imbalance factor is defined as the number of training samples in the largest class divided by that of the smallest class. Table 5 presents the test accuracy of `Fed-MNIST` under different methods. It shows that even when the dataset is imbalanced, the balanced validation set enables FedCSS to achieve higher test accuracy than all baselines. Similarly, we generate the imbalanced validation set using CIFAR-10 and present the test accuracy of the trained `Fed-CIFAR10` models with noise ratios $r = 0.2, 0.4$, when the imbalance factor of the validation set is varied from 2 to 5. It can be observed that as the imbalance factor increases, the test accuracy decreases slightly. As an example, with an imbalance factor of 5, the test accuracy is 59.31%, a drop of 3.23% compared to the uniformly distributed validation set.

Impact of the size of validation set, M. We further investigate how FedCSS behaves as the size of validation set varies to explore the tradeoff between model performance and acquisition cost for the validation set. Fig. 8(a) presents the test accuracy of the trained `Fed-CIFAR10` models on D_C^m with noise ratios $r = 0.2, 0.4$, when the size of validation set is varied from 0 to 4,000. It can be observed that using 20 validation samples on the server for all classes brings 1.2% and 6.87% performance improvement, respectively. The test accuracy stops increasing after the validation size increases beyond 1,000. Thus, FedCSS can achieve good FL model performance using a small validation set, which is advantageous in real-world application as it incurs low data acquisition.

Impact of non-iid degree c. We also evaluate how FedCSS behaves as the non-iid degree c is varied from 1 to 9, where c indicates the non-iid setting category number for each client. Fig. 8(b) presents the test accuracy of the trained `Fed-MNIST` models on dataset D_M^m with noise ratio $r = 0.4$. It can be observed that FedCSS achieves the highest test accuracy in almost all cases, and is highly robust against different non-iid degree settings.

6.4 Comparison with Centralized Baselines

We compare our design with state-of-the-art works for centralized learning: Camel [62], which selects a coreset from each streaming data batch by solving the submodular maximization problem and uses the coreset for model updating. We use Camel and FedCSS in centralized scenarios, named CentralSS, to train models on datasets D_M^m and D_C^m with various noise ratios. The results in Table 6 show that our sample-aware noise-robust learning method outperforms existing centralized work as well.

Table 6. Test accuracy comparison with centralized baselines.

Dataset	Method	Test accuracy (%) ± standard deviations			
		0.1	0.3	0.5	0.7
D_M^m	Camel	98.35± 0.07	97.51± 0.02	96.89± 0.14	91.13± 0.31
	CentralSS	98.71± 0.01	98.46± 0.01	98.12± 0.07	97.27± 0.03
D_C^m	Camel	80.03± 0.24	76.16± 0.57	69.52± 0.30	58.89± 0.12
	CentralSS	89.42± 0.43	86.51± 0.21	78.93± 0.22	65.18± 0.51

6.5 Large-scale Experiment Results

To further evaluate the scalability of FedCSS, we conduct large-scale client engagement experiments for training `Fed-MNIST`, `Fed-CIFAR10` and `Fed-KDD` on dataset D_M^m with 100 to 1,000 clients. We

present the example test accuracy of models `Fed-MNIST`, `Fed-CIFAR10` with different number of clients in Fig. 9(a), 9(b). Fig. 9(a) shows that the test accuracy of FedCSS is consistently high, *e.g.*, above 95.0% for D_M^m, as the number of client increases to 1,000 and outperforms all comparison baselines. As an example, in the 1,000 client setting of D_M^m, the average test accuracy of FedCSS is 1.28% higher than the best performing baseline.

Further, we present two examples of computation and communication costs for each local client under 500 and 1,000 client settings of `Fed-MNIST` in Fig. 9(c), 9(d), and the costs of other settings are quite similar in Fig. 9(c), 9(d). Similarly, we use the early stop strategy in which we run FL either until a pre-specified test accuracy (*e.g.*, 95.0%) is reached for 10 consecutive communication rounds, or until a maximum number of rounds (*e.g.*, 500 rounds) have elapsed. The results in Fig. 9(c) and Fig. 9(d) show that FedCSS significantly saves both computation and communication costs. For example, in the 1,000 client setting, FedCSS achieves 53.05% lower runtime and 27.54% lower communication cost than the best performing baseline. In summary, under large-scale experiment settings, FedCSS still achieves the highest test accuracy most efficiently.

7 CONCLUSIONS AND FUTURE WORK

In this work, we proposed an efficient bilevel optimization approach, FedCSS, for jointly selecting high quality FL clients and high quality local data samples in order to build high-performance FL models. It leverages meta-learning based online approximation to iteratively update the global FL model, while selecting the most positively influential samples in the presence of training set biases in a privacy-preserving manner. Besides, the proposed hierarchical analysis technique helps it reduce unnecessary influence computation, thereby saving both the computation and communication overhead. To the best of our knowledge, it is the first FL approach which is simultaneously hard sample-aware and noise-robust. Its working mechanism of boosting the selection probabilities of the positively influential data owners and samples, while suppressing those for biased ones is interpretable. In summary, FedCSS is an efficient approach for training high performance FL models via joint client-and-sample selection.

ACKNOWLEDGMENTS

This research/project is supported by the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN); National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-019); National Satellite of Excellence in Trustworthy Software System No. NRF2018NCR-NSOE003-0001, and NRF Investigatorship No. NRF-NRFI06-2020-0001.

REFERENCES

[1] Brendan McMahan, Eider Moore, and Ramage. Communication-efficient learning of deep networks from decentralized data. In *ICML*, 2017.

[2] Ergute Bao, Yizheng Zhu, Xiaokui Xiao, Yin Yang, Beng Chin Ooi, Benjamin Hong Meng Tan, and Khin Mi Mi Aung. Skellam mixture mechanism: a novel approach to federated learning with differential privacy. *Proceedings of the VLDB Endowment*, 15(11):2348–2360, 2022.

[3] Anran Li, Hongyi Peng, Lan Zhang, Jiahui Huang, Qing Guo, Han Yu, and Yang Liu. Fedsdg-fs: Efficient and secure feature selection for vertical federated learning. *arXiv preprint arXiv:2302.10417*, 2023.

[4] Yihang Cheng, Lan Zhang, and Anran Li. Gfl: Federated learning on non-iid data via privacy-preserving synthetic data. In *2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 61–70. IEEE, 2023.

[5] Andrew Hard, Kanishka Rao, Rajiv Mathews, and Ramaswamy. Federated learning for mobile keyboard prediction. *DeepAI*, 2018.

[6] Guangjing Wang, Hanqing Guo, Anran Li, Xiaorui Liu, and Qiben Yan. Federated iot interaction vulnerability analysis. In *2023 IEEE 39th International Conference on Data Engineering (ICDE). IEEE*, 2023.

[7] Abdullatif Albaseer, Bekir Sait Ciftler, and Abdallah. Exploiting unlabeled data in smart cities using federated edge learning. In *International Wireless Communications and Mobile Computing (IWCMC)*, pages 1666–1671. IEEE, 2020.

[8] Ittai Dayan, Holger R Roth, Aoxiao Zhong, Ahmed Harouni, Amilcare Gentili, Anas Z Abidin, Andrew Liu, Anthony Beardsworth Costa, Bradford J Wood, Chien-Sung Tsai, et al. Federated learning for predicting clinical outcomes in patients with covid-19. *Nature medicine*, 27(10):1735–1743, 2021.

[9] Anran Li, Lan Zhang, Juntao Tan, Yaxuan Qin, and Xiang-Yang Li. Sample-level data selection for federated learning. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[10] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[11] Anran Li, Lan Zhang, Jianwei Qian, Xiang Xiao, Xiang-Yang Li, and Yunting Xie. Todqa: Efficient task-oriented data quality assessment. In *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pages 81–88. IEEE, 2019.

[12] Xiang Li, Kaixuan Huang, Wenhao Yang, and Shusen Wang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[13] Anran Li, Lan Zhang, Junhao Wang, Feng Han, and Xiang-Yang Li. Privacy-preserving efficient federated-learning model debugging. *IEEE Transactions on Parallel and Distributed Systems*, 33(10):2291–2303, 2021.

[14] Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*, pages 872–881. PMLR, 2019.

[15] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.

[16] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

[17] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, pages 19–35, 2021.

[18] Anran Li, Rui Liu, Ming Hu, Luu Anh Tuan, and Han Yu. Towards interpretable federated learning. *arXiv preprint arXiv:2302.13473*, 2023.

[19] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[20] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[21] Chuang Zhu, Wenkai Chen, Ting Peng, Ying Wang, and Mulan Jin. Hard sample aware noise robust learning for histopathology image classification. *IEEE Transactions on Medical Imaging*, 41(4):881–894, 2021.

[22] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, pages 2304–2313. PMLR, 2018.

[23] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, and Xu. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31, 2018.

[24] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

[25] Jaemin Shin, Yuanchun Li, Yunxin Liu, and Sung-Ju Lee. Sample selection with deadline control for efficient federated learning on heterogeneous clients. *arXiv preprint arXiv:2201.01601*, 2022.

[26] Zelei Liu, Yuanyuan Chen, Yansong Zhao, Han Yu, Yang Liu, Renyi Bao, Jinpeng Jiang, Zaiqing Nie, Qian Xu, and Qiang Yang. Contribution-aware federated learning for smart healthcare. In *Proceedings of the 34th Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-22)*, 2022.

[27] Junhao Wang, Lan Zhang, and Haoran Cheng. Efficient participant contribution evaluation for horizontal and vertical federated learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 911–923. IEEE, 2022.

[28] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1698–1707. IEEE, 2020.

[29] Monica Ribero and Haris Vikalo. Communication-efficient federated learning via optimal client sampling. *arXiv preprint arXiv:2007.15197*, 2020.

[30] Google AI. Federated learning: Collaborative machine learning without centralized training data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

[31] Anran Li, Lan Zhang, Junhao Wang, Juntao Tan, Feng Han, Yaxuan Qin, Nikolaos M Freris, and Xiang-Yang Li. Efficient federated-learning model debugging. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 372–383. IEEE, 2021.

[32] Lokesh Nagalapatti and Ruhi Sharma Mittal. Is your data relevant?: Dynamic selection of relevant data for federated learning. 2022.

[33] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.
[34] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. PMLR, 2017.
[35] Dongyuan Shi, Woon-Seng Gan, Bhan Lam, Zhengding Luo, and Xiaoyi Shen. Transferable latent of cnn-based selective fixed-filter active noise control. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pages 1–12, 2023.
[36] Tianlin Li, Aishan Liu, Xianglong Liu, Yitao Xu, Chongzhi Zhang, and Xiaofei Xie. Understanding adversarial robustness via critical attacking route. *Information Sciences*, 547:568–578, 2021.
[37] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of exemplar-svms for object detection and beyond. In *2011 International conference on computer vision*, pages 89–96. IEEE, 2011.
[38] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, 30, 2017.
[39] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. *Advances in neural information processing systems*, 31, 2018.
[40] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.
[41] Ravikumar Balakrishnan, Tian Li, Tianyi Zhou, Nageen Himayat, Virginia Smith, and Jeff Bilmes. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*, 2021.
[42] Tiffany Tuor, Shiqiang Wang, Bong Jun Ko, Changchang Liu, and Kin K Leung. Overcoming noisy and irrelevant data in federated learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5020–5027. IEEE, 2021.
[43] Davoud Ataee Tarzanagh, Mingchen Li, Christos Thrampoulidis, and Samet Oymak. Fednest: Federated bilevel, minimax, and compositional optimization. *arXiv preprint arXiv:2205.02215*, 2022.
[44] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.
[45] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.
[46] R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.
[47] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32, 2019.
[48] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.
[49] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.
[50] Pytorch. Pytorch. Public online, 2022.
[51] Yann LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.
[52] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.
[53] Sai Meher Karthik Duddu Sebastian Caldas. Leaf: A benchmark for federated settings. https://leaf.cmu.edu/.
[54] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *semanticscholar*, 2009.
[55] Alex Krizhevsky et al. Cifar-100 dataset. cs.toronto.edu/~kriz/cifar.html.
[56] Antonio Gulli. Ag's corpus of news articles. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html.
[57] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018.
[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
[59] Ben Trevett. Sentiment analysis. https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/6%20-%20Transformers%20for%20Sentiment%20Analysis.ipynb.
[60] Yiqiang Chen, Xiaodong Yang, Xin Qin, Han Yu, Biao Chen, and Zhiqi Shen. Focus: Dealing with label quality disparity in federated learning. *arXiv preprint arXiv:2001.11359*, 2020.
[61] Huili Chen, Jie Ding, Eric Tramel, Shuang Wu, Anit Kumar Sahu, Salman Avestimehr, and Tao Zhang. Actperfl: Active personalized federated learning. 2022.
[62] Yiming Li, Yanyan Shen, and Lei Chen. Camel: Managing data for efficient stream learning. In *Proceedings of the 2022 International Conference on Management of Data*, pages 1271–1285, 2022.

Received 15 January 2023; revised 20 April 2023; accepted 23 May 2023