FedProf: Selective Federated Learning based on Distributional Representation Profiling

Journal:	*Transactions on Parallel and Distributed Systems*
Manuscript ID	TPDS-2022-08-0496.R1
Manuscript Type:	Regular
Keywords:	C.2.4 Distributed Systems < C.2 Communication/Networking and Information Technology < C Computer Systems Organization, I.2.6.g Machine learning < I.2.6 Learning < I.2 Artificial Intelligence < I Computing Methodologies, I.4.7.a Feature representation < I.4.7 Feature Measurement < I.4 Image Processing and Computer Vision < I Computing Methodologie, I.2.6.c Connectionism and neural nets < I.2.6 Learning < I.2 Artificial Intelligence < I Computing Methodologies

FedProf: Selective Federated Learning based on Distributional Representation Profiling

Wentai Wu, *Member, IEEE,* Ligang He, *Member, IEEE,* Weiwei Lin, *Member, IEEE,* Carsten Maple, *Member, IEEE*

Abstract—Federated Learning (FL) has shown great potential as a privacy-preserving solution to learning from decentralized data that are only accessible to end devices (i.e., clients). The data locality constraint offers strong privacy protection but also makes FL sensitive to the condition of local data. Apart from statistical heterogeneity, a large proportion of the clients, in many scenarios, are probably in possession of low-quality data that are biased, noisy or even irrelevant. As a result, they could significantly slow down the convergence of the global model we aim to build and also compromise its quality. In light of this, we first present a new view of local data by looking into the representation space and observing that they converge in distribution to Normal distributions before activation. We provide theoretical analysis to support our finding. Further, we propose FEDPROF, a novel algorithm for optimizing FL over non-IID data of mixed quality. The key of our approach is a distributional representation profiling and matching scheme that uses the global model to dynamically profile data representations and allows for low-cost, lightweight representation matching. Using the scheme we sample clients adaptively in FL to mitigate the impact of low-quality data on the training process. We evaluated our solution with extensive experiments on different tasks and data conditions under various FL settings. The results demonstrate that the selective behavior of our algorithm leads to a significant reduction in the number of communication rounds and the amount of time (up to 2.4× speedup) for the global model to converge and also provides accuracy gain.

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

1 INTRODUCTION

WITH the advances in Artificial Intelligence (AI), we are seeing a rapid growth in the number of AI-driven applications as well as the volume of data required to train them. However, a large proportion of data used for machine learning are often generated outside the data centers by distributed resources such as mobile phones and IoT (Internet of Things) devices. It is predicted that the data generated by IoT devices will account for 75% of the total in 2025 [1]. Under this circumstance, it will be very costly to gather all the data for centralized training. More importantly, moving the data out of their local devices (e.g., mobile phones) is now restricted by law in many countries, such as the General Data Protection Regulation (GDPR)[1] enforced in EU.

The restriction on data sharing brings unprecedented challenges to the way machine learning models are built and trained. Data-centralized training is not feasible in privacy-sensitive situations, whilst traditional distributed training methods often require non-exclusive access to local data, incur heavy communication traffics, and risk the leakage of user privacy. In the meantime, the rapid development of microprocessors and AI hardware have empowered a variety of mobile devices and IoT devices. This further pushes the trend that both data and computation are sinking to the network edge, which calls for a new paradigm of distributed training.

We face three main difficulties in learning from decentralized data: i) massive scale of end devices; ii) limited communication bandwidth at the network edge; and iii) uncertain data distribution and data quality. As an promising solution, Federated Learning (FL) [2] is a framework for efficient distributed machine learning with privacy protection (i.e., no data exchange). An FL system involves a group of devices (called *clients* in the FL context) that are in possession of private data and coordinated by a central server to perform model training in a collaborative manner. A typical process of FL is organized in rounds, and in each round the devices and the server interact in the following steps: (step 1) *client sampling*: the server selects a subset of clients for this round of training; (step 2) *model distribution*: the selected clients download the latest global model from the server; (step 3) *local training*: the selected clients perform local training on their data and then upload their updated local models to the server; (step 4) *model aggregation*: the server aggregates the local models into a new global model (i.e., the federated model).

Compared to traditional distributed learning methods, FL is naturally more communication-efficient at scale [3], [4]. Nonetheless, the data locality constraint inevitably introduces new challenges. On the one hand, the server has no control over the data sources and is unaware of their distribution and quality. On the other hand, the heterogeneity

- *W. Wu* is with Peng Cheng Laboratory, Shenzhen 518000, China. E-mail: nnwtwu@pcl.ac.cn*
- *L. He (corresponding author) is with the Department of Computer Science, University of Warwick, Coventry CV4 7AL, United Kingdom. E-mail: ligang.he@warwick.ac.uk*
- *W. Lin is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. E-mail: linww@scut.edu.cn*
- *C. Maple is with the WMG, University of Warwick, Coventry CV4 7AL, United Kingdom. E-mail: CM@warwick.ac.uk*

**Work performed while at the University of Warwick.*
Manuscript received xxx, yyy; revised xxx, yyy.

1. https://gdpr.eu/what-is-gdpr/

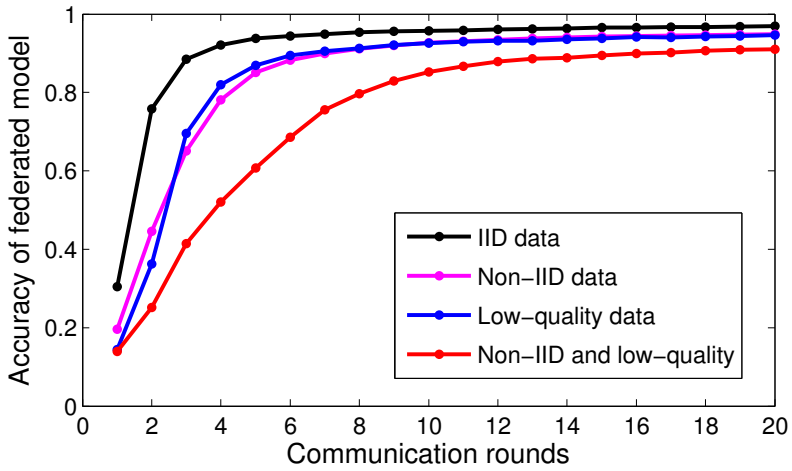


Fig. 1: (Preliminary experiment, with more details in Section 3.1) The global model's convergence in FL over 100 clients on different data conditions where the data are 1) *independent and identically distributed (IID)*: noiseless, class-balanced and evenly distributed across the clients, 2) *non-IID*: locally class-imbalanced and globally heterogeneous, 3) *low-quality*: blended with noise and irrelevant samples, or 4) *non-IID and low-quality*: heterogeneous in distribution and affected by noise and useless samples.

of data in terms of distribution and quality largely impacts the efficacy of FL and the generalization of the federated model. These problems stand out especially in cross-device FL scenarios such as crowdsourcing systems [5], [6] where a large number of users are recruited and a strong discrepancy in data quality is expected.

1.1 Motivation

1) FL is susceptible to biased and low-quality local data. Partial participation is a common practice in FL [7], [8]. The standard FL algorithm FEDAVG[2] uses random selection (sampling), which implies that every client (and its local data) is considered equally important. This makes the training process susceptible to local data with strong heterogeneity and of low quality (e.g., user-generated texts [9] and noisy photos). In some scenarios, local data may contain irrelevant or even poisonous samples from malicious clients [10], [11]. Due to the lack of global knowledge, filtering or augmenting local data may introduce noise and bias [12], [13] and risk information leakage [14], whilst simply excluding potentially noisy clients is often impractical because i) the quality of the data depends on the learning task and is difficult to gauge, and ii) sometimes low-quality data are very common across the clients.

In Fig. 1 we demonstrate the impact of involving low-quality data by running FL over 100 clients to learn a CNN model on MNIST using FEDAVG. The setup of this preliminary experiment is detailed in Section 3.1. From the traces we can see that training over clients with problematic or strongly biased data can compromise the efficiency and efficacy of FL, resulting in an inferior global model that takes more rounds to converge.

2) Learned representations can reflect data distribution and quality. Learned representations capture the intrinsic structure of data [15]. Their value lies in the fact that they characterize the domain of the learning task and provide task-specific knowledge. In the field of FL, recent studies use raw representations for refining model update rules [16], [17] or facilitating unsupervised learning [18], [19]. however, the distributional difference between representations of heterogeneous data is not yet explored. On this point, our key hypothesis is that local data distribution can be examined in a consensus representation space as a byproduct of the global model (Fig. 2). Our study is also motivated by a key observation that *representations from neural networks follow the Normal Distribution* (we demonstrate these patterns in Section 3.2). This implies that it is theoretically possible to condense the knowledge provided by representations from a distributional perspective, which in turn facilitates the examination of data quality and the estimation of their training value to the FL system.

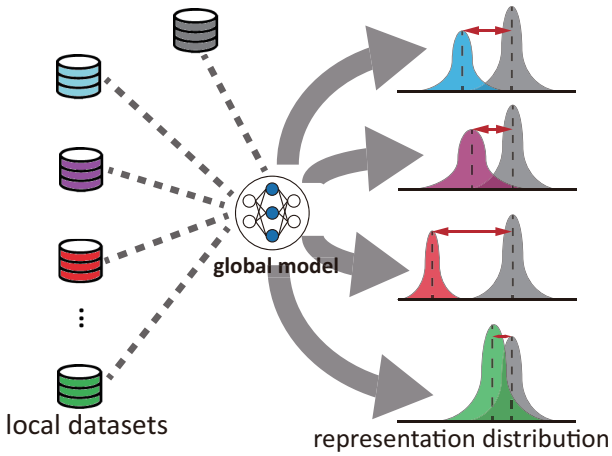


Fig. 2: The heterogeneity and quality of local data can be reflected by the distribution of representations.

1.2 Contributions

The aim of our research is to improve the performance and efficiency of FL over a group of clients with unknown conditions of data that can be heterogeneous in distribution (i.e., non-IID) and/or comprised of low-quality samples. Empirically, we first observe that the hidden layers of neural networks generate representations that follow the normal distribution. Theoretically, we prove our observations via the Central Limit Theorem (CLT) and design a novel scheme to statistically condense representations into profiles for fast, secure comparison of representations. We further propose a novel FL algorithm that features an adaptive client sampling strategy which selects clients based on their profile dissimilarity to avoid involving low-quality data. Experimental evaluation of the proposed solution shows improvement in the global model's accuracy and a significant reduction in the costs for convergence.

Our contributions are summarized as follows:

- We first provide theoretical proofs for the observation that data representations from neural nets exhibit normality in distribution. To the best of our knowledge, this is the first study on statistical distribution of representations and its application to FL.
- We propose a representation profiling and matching scheme based on the statistical distribution of raw representations and the special property of Kullback–Leibler (KL) divergence when applied to normal distributions. This allows for fast, low-cost representation comparison in a secure manner.
- We present a novel FL algorithm FEDPROF with adaptive client sampling based on representation profile dissimilarity.
- Through extensive experiments we show that FEDPROF reduces the number of communication rounds by up to 63%, shortens the overall training time (up to 2.4× speedup) while improving the accuracy by up to 6.8% over FEDAVG and its variants.

2 Related Work

Different from traditional distributed training methods (e.g., [20], [21]), Federated Learning assumes strict constraints of data locality and limited communication capacity [22]. Much effort has been made in optimizing FL and covers a variety of perspectives including communication [23], [24], update rules [25], [26], [27], [28], flexible aggregation [4], [29] and personalization [30], [31].

The control of device participation is imperative in cross-device FL scenarios [32], [33] where the quality of local data is uncontrollable and the clients show varied value for the training task [6]. However, little attention has been paid to the problems caused by low-quality data and their impact on FL's efficiency and effectiveness. Tuor et al. [6] figure out that clients of FL are likely to contain a lot of useless data and only a subset of them is valuable for the task. They further propose a data selection method based on loss distribution in order to filter out noisy and irrelevant data on local devices. However, their solution requires a baseline loss distribution produced by a pre-trained model, which could be difficult to set up without sufficient prior knowledge. Evidence shows that the selection of clients is pivotal to the convergence of FL over heterogeneous data and devices [34], [35]. Non-uniform client sampling/selection is widely adopted in existing studies and has been theoretically proven with convergence guarantees [36], [37]. Many approaches sample clients based on their performance [38] or aim to jointly optimize the model accuracy and training time [39], [40], [41]. A popular strategy is to use loss as the information to guide client selection [42], [43]. For example, Goetz et al. [42] proposed a loss-oriented client selection strategy, in which they prioritize the clients with higher loss feedback. This strategy tends to select clients on which the model yields higher error, but it is potentially susceptible to noisy and unexpected data that yield illusive loss values, in which case the holders of these low-quality data may turn out to be in favor. Chai et al. [44] developed a tier-based FL system called TiFL where clients are assigned to different groups by performance (i.e., training speed). TiFL adopts a tier-level client sampling strategy which dynamically adjusts the probability that each tier gets selected according to the model accuracy. This strategy favors the tiers (of clients) that report lower accuracy, which however, can be misguided due to the existence of low-quality data.

Recent studies exploit data representations to facilitate vertical FL [17], personalized FL [45] and unsupervised FL [18]. Li et al. [16] introduces representation similarities into local objectives. This contrastive learning approach uses the raw representation vectors from multiple models to regularize local training losses, which guides the direction of local model update to avoid model divergence. However, this solution can only apply to Cross Entropy loss whilst the computation costs of high-dimensional representations makes another problem. In this paper, we seek to utilize the knowledge behind representations in a more secure and efficient manner from a distributional perspective. To the best of our knowledge, the study on the distribution of data representations is still lacking whilst its connection to clients' training value is hardly explored either.

3 Preliminary Experiment

3.1 Impact of Data Quality on FL

We set up an FL system to train a LeNet-5 model on MNIST for demonstration purpose. We set up 100 clients with a sampling fraction of 0.3, i.e., cohort size=30. For local training, SGD with momentum is used as the local optimizer. We experimented under both IID (Identical and Independently Distributed) and non-IID data settings. More specifically, we partitioned the MNIST dataset into 100 subsets in four different ways: 1) *IID* (**black** curve in Fig. 1): all the data are noiseless and evenly distributed across the clients, which represents the ideal condition of local data; 2) *non-IID* (**magenta** curve in Fig. 1): local datasets are heterogeneous and class-imbalanced. On each client the dominant class accounts for >50% of the samples; 3) *low-quality* (**blue** curve in Fig. 1): low-quality data cover 65% of the clients—15% with irrelevant images, 25% with blurred images, and 25% with images affected by "salt and pepper" noise; 4) *non-IID and low-quality* (**red** curve in Fig. 1): all local datasets are non-IID (i.e., as in the *Non-IID* case) while 65% of them are filled with low-quality data (as in the *low-quality* case).

From Fig. 1 we can observe that the convergence of the global model is greatly impacted by the distribution of the training data as well as their quality. From a statistical standpoint, the existence of low-quality samples (e.g., noisy and irrelevant images) shifts the marginal distribution of the corresponding domain, which could hinder the model from learning the salient patterns and thus has negative effect on its generalization. Training over clients with low-quality data significantly undermines the effectiveness of FL resulting in an inferior federated model that takes more rounds to converge.

3.2 Distribution of Representations

We also ran preliminary experiment to demonstrate the distribution of representations learned by two popular neural network models during and after training, respectively. For representation extraction and profiling, we define a structured tensor (initialized with the model) that caches the representations in the forward pass.

Fig. 3 shows the distribution of fused representations (in a channel-wise manner) extracted from a plain convolutional layer and a residual block of ResNet-18. The model has been trained till convergence on the CIFAR-100 dataset. In a forward pass we extracted the output feature maps of the corresponding layer/block and examine the distribution of representations from a channel-wise perspective. It can be observed that representations from both plain convolution and residual block converged to normal distribution.

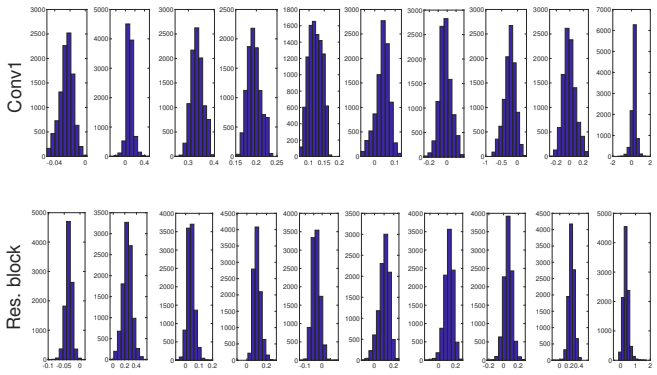


Fig. 3: (Preliminary experiment) Demonstration of fused representations from a standard convolution layer (1st row) and a residual block (2nd row) of a ResNet-18 model trained for 100 epochs on CIFAR-100. Ten (out of 64) channels for each layer are displayed for brevity.

We also investigated the convergence (in distribution) of representations and the impact of low-quality data. Fig. 4 visualizes the representations extracted from the first dense layer (FC-1) of LeNet-5. The three rows in Fig. 4 correspond to the model trained for 1 epoch, 6 epochs and 10 epochs, respectively. We can observe clear patterns of normally distributed representations for both under-trained models and models already converged.[2] Another key observation is that the representations of high-quality data and those of noisy data exhibit obvious difference in distribution. Representations learned on MNIST with noise[3] have shifted means and lower variances.

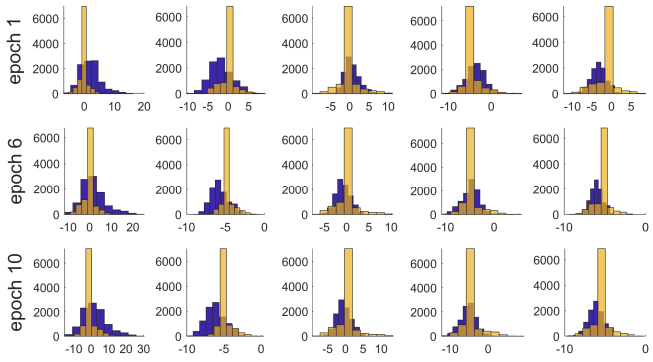


Fig. 4: (Preliminary experiment) Demonstration of pre-activation representations from a dense layer of a LeNet-5 model after being trained for 1, 6 and 10 epochs on MNIST (blue) and noisy MNIST (yellow). Representations from five (out of 128) neurons in the layer are displayed for brevity.

We empirically found that tiny datasets can also exhibit normal distribution patterns of representation using a simple model as the encoder (similar to the role of the global model in FL). More details are provided in Appendix B.1.

4 Data Representation Profiling and Matching

In this paper, we consider a typical cross-device FL setting [32], in which multiple end devices collaboratively perform local training on their own datasets $D_i, i = 1, 2, ..., n$. Every dataset is only accessible to its owner.

Considering the distributional pattern of data representations (Figs. 3 and 4) and the role of the global model in FL, we propose to profile the representations of local data using the global model. In this section, we first provide theoretical proofs to support our observation that representations from neural network models converge to normal distributions. Then we present a novel scheme to profile data representations and define profile dissimilarity for fast and secure representation comparison.

4.1 Normal Distribution of Representations

In this section we provide theoretical explanations for the Normal patterns exhibited by the representations from the hidden layers of neural networks. We first make the following definition to facilitate our analysis.

Definition 4.1 (The Lyapunov's condition)**.** *A set of random variables $\{Z_1, Z_2, \ldots, Z_v\}$ satisfy the Lyapunov's condition if there exists a δ such that*

$$\lim_{v\to\infty} \frac{1}{s^{2+\delta}} \sum_{k=1}^{v} \mathrm{E}\left[|Z_k - \mu_k|^{2+\delta}\right] = 0, \tag{1}$$

where $\mu_k = \mathrm{E}[Z_k]$, $\sigma_k^2 = \mathrm{E}[(Z_k - \mu_k)^2]$ and $s = \sqrt{\sum_{k=1}^{v} \sigma_k^2}$.

2. LeNet-5 converges on MNIST after 10 epochs of training.

3. Gaussian blur and pixel noise are applied to 60% of the data.

The Lyapunov's condition can be intuitively explained as a limit on the overall variation (with $|Z_k - \mu_k|^{2+\delta}$ being the $(2+\delta)$-th moment of Z_k) of a set of random variables.

Now we present Proposition 4.2 and Proposition 4.3. The Propositions provide theoretical support for our representation profiling and matching scheme to be introduced in Section 4.2.

Proposition 4.2. *The representations from linear operators (e.g., a pre-activation dense layer or a plain convolutional layer) in a neural network converge in distribution to a normal distribution if the layer's weighted inputs satisfy the Lyapunov's condition.*

Proposition 4.3. *The fused representations*[4] *from non-linear operators (e.g., a hidden layer of LSTM or a residual block of ResNet) in a neural network converge in distribution to a normal distribution if the layer's output elements satisfy the Lyapunov's condition.*

We base our proofs on the Lyapunov's CLT which assumes independence between the variables. The assumption theoretically holds by using the Bayesian network concepts: let X denote the layer's input and H_k denote the k-th component in its output. The inference through the layer produces dependencies $X \rightarrow H_k$ for all k. According to the *Local Markov Property*, we have H_i independent of any H_j $(j \neq i)$ in the same layer given X. Also, the Lyapunov's condition is typically met when the model is properly initialized and batch normalization is applied.

For brevity, we present the proof of Propositions 4.2 for fully-connected layers in this section. We extend the results to convolutional layers in Appendix A.1. The proof of Proposition 4.3 is provided in Appendix A.2.

Proof. Let $\Omega = \{neu_1, neu_2, ..., neu_q\}$ denote a dense layer (with q neurons) of any neural network model and H_k denote the pre-activation output of neu_k in Ω. We first provide the theoretical proof to support the observation that the distribution of H_k converges to normal distribution.

Let $\chi = \mathbb{R}^v$ denote the input feature space (with v features) and assume the feature X_i (which is a random variable) follows some distribution $\zeta_i(\mu_i, \sigma_i^2)$ with finite mean $\mu_i = \mathrm{E}[X_i]$ and variance $\sigma_i^2 = \mathrm{E}[X_i - \mu_i]$. For each neuron neu_k, let $W_k = [w_{k,1}\, w_{k,2} \dots w_{k,v}]$ denote the neuron's weight vector, b_k denote the bias, and $Z_{k,i} = X_i w_{k,i}$ denote the i-th weighted input. Let H_k denote the output of neu_k. During the forward propagation, we have:

$$\begin{aligned} H_k &= XW_k^T + b_k \\ &= \sum_{i=1}^{v} X_i w_{k,i} + b_k \\ &= \sum_{i=1}^{v} Z_{k,i} + b_k. \end{aligned} \tag{2}$$

Apparently $Z_{k,i}$ is a random variable because $Z_{k,i} = X_i w_{k,i}$ (where the weights $w_{k,i}$ are constants during a forward pass), thus H_k is also a random variable according to Eq. (2).

4. Fused representations refer to the sum of elements in the original representations produced by a single layer (channel-wise for a residual block).

We first consider a special case where the inputs variables $X_1, X_2, \dots, X_v$ follow a multivariate normal distribution, in which case Proposition 4.2 automatically holds due to the property of multivariate normal distribution that every linear combination of the components of the random vector $(X_1, X_2, \dots, X_v)^T$ follows a normal distribution [46]. In other words, $H_k = X_1 w_{k,1} + X_2 w_{k,2} + \ldots + X_v w_{k,v} + b_k$ is a normally distributed variable since $w_{k,i}$ and b_k $(k = 1, 2, \dots, v)$ are constants in the forward propagation. A special case for this condition is that $X_1, X_2, \dots, X_v$ are independent on each other and X_i follows a normal distribution $\mathcal{N}(\mu_i, \sigma_i^2)$ for all $i = 1, 2, \dots, v$. In this case, by the definition of $Z_{k,i}$, we have:

$$Z_{k,i} = X_i w_{k,i} \sim \mathcal{N}\big(w_{k,i}\mu_i, (w_{k,i}\sigma_i)^2\big), \tag{3}$$

where $Z_1, Z_2, \dots, Z_v$ are independent of each other. Combining Eqs. (2) and (3), we have:

$$H_k \sim \mathcal{N}\big(\sum_{i=1}^{v} w_{k,i}\mu_i + b_k, \sum_{i=1}^{v} (w_{k,i}\sigma_i)^2\big), \tag{4}$$

For more general cases where $X_1, X_2, \dots, X_v$ are not necessarily normally distributed, we assume the weighted inputs $Z_{k,i}$ of the dense layer satisfy the Lyapunov's condition (see *definition* 4.1). As a result, we have the following results according to the Central Limit Theorem (CLT) [47] considering that X_i follows $\zeta_i(\mu_i, \sigma_i^2)$:

$$\frac{1}{s_k} \sum_{i=1}^{v} \big(Z_{k,i} - w_{k,i}\mu_i\big) \xrightarrow{d} \mathcal{N}(0, 1) \tag{5}$$

where $s_k = \sqrt{\sum_{i=1}^{v} \big(w_{k,i}\sigma_i\big)^2}$ and $\mathcal{N}(0,1)$ denotes the standard normal distribution. Equivalently, for every neu_k we have:

$$\sum_{i=1}^{v} Z_{k,i} \xrightarrow{d} \mathcal{N}(\sum_{i=1}^{v} w_{k,i}\mu_i, s_k^2) \tag{6}$$

Combining Eqs. (2) and (6) we can derive that:

$$H_k \xrightarrow{d} \mathcal{N}(\sum_{i=1}^{v} w_{k,i}\mu_i + b_k, s_k^2), \tag{7}$$

which means that $H_k (k = 1, 2, \dots, v)$ converges in distribution to a random variable that follows normal distribution, which proves our Proposition 4.2 for fully-connected layers. □

We extend the result to the cases of convolutional layers and non-linear operators in Appendices A.1 and A.2.

Next, we discuss the proposed representation profiling and matching scheme.

4.2 Distributional Profiling and Matching

Based on the normal distribution of representations, we compress the data representations statistically into a condensed form called *representation profiles*. The profile produced by a θ-parameterized global model on a dataset D, denoted by $RP(\theta, D)$, is a vector of distributions:

$$RP(\theta, D) = \big(\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2), \dots, \mathcal{N}(\mu_q, \sigma_q^2)\big), \tag{8}$$

where q is the profile length determined by the dimensionality of the representations. For example, q is equal to the number of kernels for channel-wise fused representations from a convolutional layer. The tuple (μ_i, σ_i^2) contains the mean and the variance of the i-th representation element.

Local representation profiles are generated by clients and sent to the server for comparison (the cost of transmission is negligible considering each profile is only $q \times 8$ bytes). In this paper we also consider a reference dataset D_V as a small group of task-specific samples without quality issue.[5] Let RP_k denote the local profile from client k and RP_V the reference profile generated over D_V. The dissimilarity between RP_k and RP_V, denoted by $div(RP_k, RP_V)$, is defined as the expectation of KL divergence between representations $\mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V)]$ which can be approximated by:

$$div(RP_k, RP_V) = \frac{1}{q}\sum_{i=1}^{q} \mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V), \tag{9}$$

where $\mathrm{KL}(\cdot)$ denotes the Kullback–Leibler (KL) divergence. An advantage of our profiling scheme is that a *much simplified* KL divergence formula can be adopted because of the normal distribution property (see [48, Appendix B] for details), which yields:

$$\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V) = \frac{1}{2}\log\Big(\frac{\sigma_i^V}{\sigma_i^{(k)}}\Big)^2 + \frac{(\sigma_i^{(k)})^2 - (\sigma_i^V)^2}{2(\sigma_i^V)^2} + \frac{(\mu_i^{(k)} - \mu_i^V)^2}{2(\sigma_i^V)^2}, \tag{10}$$

Eq. (10) computes the KL divergence without calculating any integral, which is computationally cost-efficient. Besides, the computation of profile dissimilarity can be performed under the Homomorphic Encryption for minimum knowledge disclosure (see Appendix C.2 for details).

Low-quality data (e.g., with strong noise) deviates from the true marginal distribution $\zeta_i(\mu_i, \sigma_i^2)(i = 1, 2, \ldots, v)$ and misleads the model to learn a different set of representations (see Fig. 4 for example). Mathematically this is reflected on $\mathcal{N}_i^{(k)}(i = 1, 2, \ldots, q)$ with shifted $\mu_i^{(k)}$ and $\sigma_i^{(k)}$, which increases the KL divergence and profile dissimilarity.

5 The Training Algorithm FedProf

Our research aims to optimize the global model over a group of clients (datasets) of disparate training value. Given the client set U and the local datasets $\{D_k\}_{k\in U}$, we formulate the optimization problem in (11) where the coefficient ρ_k differentiates the importance of the local objective function $F_k(\theta)$ and depends on the data in D_k. Our global objective is in a sense similar to the Agnostic Learning scenario [49] where a non-uniform mixture of local data distributions is implied.

$$\arg\min_{\theta} F(\theta) = \sum_{k=1}^{N} \rho_k F_k(\theta), \tag{11}$$

where $N = |U|$ and θ denotes the parameters of the global model $h_\theta \in \mathcal{H} : \chi \to \mathcal{Y}$ over the feature space χ and target space $\mathcal{Y}$. The coefficients $\{\rho_k\}_{k=1}^N$ add up to 1. $F_k(\theta)$ is client k's local objective function of training based on the loss function $\ell(\cdot)$:

$$F_k(\theta) = \frac{1}{|D_k|}\sum_{(x_i,y_i)\in D_k} \ell(h_\theta(x_i), y_i), \tag{12}$$

Consider a task-specific domain $\mathcal{D}$ defined by its feature space χ and the marginal distribution $P(X)$. Our propositions imply that the change in $P(X)$ (by low-quality samples) leads to the distributional shift of representations. Hence, using RP_V as a reference, the dissimilarity between representation profiles is a strong indicator of local data condition. With this motivation we score each client with λ_k each round based on the representation profile dissimilarity:

$$\lambda_k = \exp\big(-\alpha_k \cdot div(RP_k, RP_V)\big), \tag{13}$$

where RP_k and RP_V are generated by an identical global model; α_k is the penalty factor deciding how biased the strategy needs to be against client k. With $\alpha_k = 0$ for all $k = 1, 2, \ldots N$, our strategy is equivalent to random sampling.

The scores connect the representation profiling and matching scheme to the design of the selective client sampling strategy adopted in our FL algorithm FedProf, which is outlined in Algorithm 1. The key steps of our algorithm are local representation profiling (line 10), reference representation profiling (line 15) and client sampling (lines 7 and 8). More technical details of the proposed algorithm are provided in Appendix C.1.

Algorithm 1 the FedProf algorithm

Input: max number of rounds T_{max}, number of iterations per round τ, sampling fraction C;
1: Initialize global model θ
2: Broadcast the same seed to all clients for an identical initial model
3: Generate reference profile RP_V
4: Collect initial profiles $\{RP_k\}_{k\in U}$ from all clients
5: **for** round $T \leftarrow 1$ **to** T_{max} **do**
6: Distribute θ to the clients
7: Update scores $\{\lambda_k\}_{k\in U}$ and compute $\Lambda = \sum_{k\in U}\lambda_k$
8: $S \leftarrow$ sample $K = N \cdot C$ clients by probability distribution $\{\frac{\lambda_k}{\Lambda}\}_{k\in U}$
9: **for** client k in S **in parallel do**
10: $RP_k \leftarrow$ UpdateProfile$(k, \theta, T-1)$
11: $\theta_k \leftarrow$ LocalTraining(k, θ, τ)
12: **end for**
13: Collect local profiles from the clients in S
14: Update θ via model aggregation
15: Update RP_V
16: Evaluate h_θ
17: **end for**
18: **return** θ

The convergence rate of FL algorithms with opportunistic client sampling has been extensively studied in the literature [50], [4]. We first make the following assumptions to facilitate the analysis:

Assumption 5.1. $F_1, F_2, \ldots, F_N$ are L-smooth, i.e., for any $k \in U$, x and y: $F_k(y) \leq F_k(x) + (y-x)^T\nabla F_k(x) + \frac{L}{2}\|y-x\|_2^2$

5. We assume D_V on the server, but practically one or several local datasets can also serve the purpose.

It is obvious that the global objective F is also L-smooth as a linear combination of $F_1, F_2, \ldots, F_N$ with $\rho_1, \rho_2, \ldots, \rho_N$ being the weights.

Assumption 5.2. $F_1, F_2, \ldots, F_N$ are μ-strongly convex, i.e., for all $k \in U$ and any x, y: $F_k(y) \geq F_k(x) + (y - x)^T \nabla F_k(x) + \frac{\mu}{2}\|y - x\|_2^2$

Assumption 5.3. The variance of local stochastic gradients on each device is bounded: For all $k \in U$, $\mathrm{E}\|\nabla F_k(\theta_k(t), \xi_{t,k}) - \nabla F_k(\theta_k(t))\|^2 \leq \epsilon^2$

Assumption 5.4. The squared norm of local stochastic gradients on each device is bounded: For all $k \in U$, $\mathrm{E}\|\nabla F_k(\theta_k(t), \xi_{t,k})\|^2 \leq G^2$

These assumptions are commonly adopted in the literature [51], [52], [50]. Inspired by these studies, we present Theorem 5.5 to guarantee the global model's convergence for our algorithm. The notations and key lemmas are formally presented in Appendix D.

Theorem 5.5. *Using partial aggregation and our client sampling strategy, the global model $\theta(t)$ converges in expectation given an aggregation interval $\tau \geq 1$, a decreasing step size (learning rate) $\eta_t = \frac{2}{\mu(t+\gamma)}$, and the condition that $\alpha_k \mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)} || \mathcal{N}_i^V)] = -ln(\Lambda \rho_k)$:*

$$\mathrm{E}\big[F(\theta(t))\big] - F^* \leq \frac{L}{(\gamma + t)}\Big(\frac{2(\mathcal{B} + \mathcal{C})}{\mu^2} + \frac{\gamma + 1}{2}\Delta_1\Big), \quad (14)$$

where $t \in T_A = \{n\tau | n = 1, 2, \ldots\}$, $\gamma = \max\{\frac{8L}{\mu}, \tau\} - 1$, $\mathcal{B} = \sum_{k=1}^N \rho_k^2 \epsilon_k^2 + 6L\Gamma + 8(\tau - 1)^2 G^2$, $\mathcal{C} = \frac{4}{K}\tau^2 G^2$, $\Gamma = F^* - \sum_{k=1}^N \rho_k F_k^*$, $\Delta_1 = \mathrm{E}\|\bar{\theta}(1) - \theta^*\|^2$, $\Lambda = \sum_{k=1}^N \lambda_k$, and $K = |S(t)| = N \cdot C$.

The proof of Theorem 5.5 is provided in Appendix D.

Computationally, local profiling is performed in the forward pass of inference and, similar to Batch Normalization, only incurs arithmetic computation for means and variances, which is not costly. The memory space for caching representations depends on the data size and the number of neurons (or channels), which can be flexibly configured for memory efficiency. Once encoded, the distribution-based profiles, as plain arrays, are ultra-light for transmission.

6 Experiments

We conducted extensive experiments to evaluate FedProf under various FL settings. Apart from FedAvg [2] as the baseline, we also reproduced several FL algorithms for comparison. These include CFCFM [29], FedAvg-RP [50], FedProx [25], FedADAM [27], AFL [53], and Clustered Sampling [54] (based on model similarity). We also drew comparison to FedBN [55] on the learning task that uses batch normalization. For fair comparison, the algorithms are grouped by their aggregation method (i.e., full aggregation and partial aggregation) and configured based on the parameter settings suggested in their papers. Note that our algorithm can adapt to both aggregation methods.

6.1 Experimental Setup

We built a discrete event-driven, simulation-based FL system and implemented the training logic under the Pytorch framework (Build 1.7.0). To evaluate the algorithms in disparate FL scenarios, we set up three different tasks using three public datasets: *GasTurbine*[6], *EMNIST*[7] and *CIFAR-10*, which are commonly used in the literature [8], [28]. With GasTurbine the goal is to learn a carbon monoxide (CO) and nitrogen oxides (NOx) emission prediction model over a network of 50 sensors. Using EMNIST and CIFAR-10 we set up two image classification tasks with different models (LeNet-5 [56] for EMNIST and ShuffleNet v2 [57] for CIFAR-10). We experimented at various scale—500 mobile clients for EMNIST and 10 dataholders for CIFAR-10—to emulate cross-device and cross-silo scenarios [32] respectively. The penalty factors $\boldsymbol{\alpha}$ are set to $(a, a, \ldots, a)$ where a=10.0, 10.0 and 25.0 for GasTurbine, EMNIST and CIFAR-10, respectively.

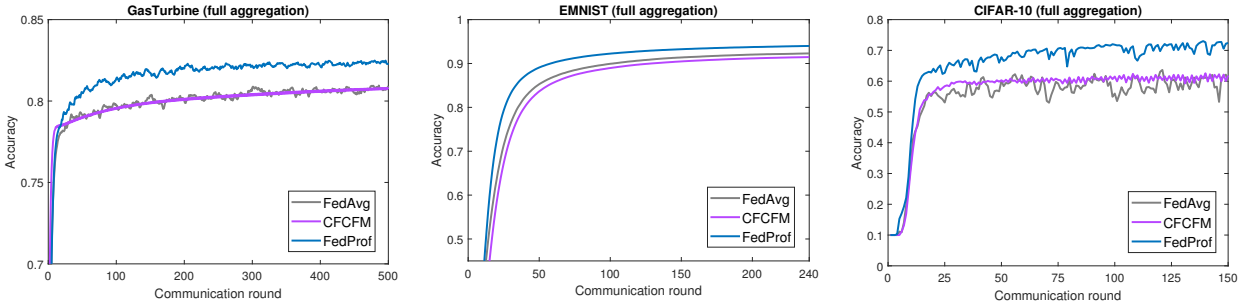
In all the tasks, data sharing is not allowed between any parties and the data are non-IID across the clients. We made local data statistically heterogeneous by forcing class imbalance or size imbalance. Each client has a dominant class that accounts for 60% (for EMNIST) or 37% (for CIFAR) of the samples. For GasTurbine, the sizes of local datasets follow the normal distribution $\mathcal{N}(514, 101^2)$. We introduce a diversity of noise into the local datasets to simulate the discrepancy in data quality: for GasTurbine, 50% of the sensors produce noisy data (including 10% polluted); for EMNIST and CIFAR-10, a certain percentage of local datasets are irrelevant images or low-quality images (blurred or affected by salt-and-pepper noise). Aside from data heterogeneity, all the clients are also heterogeneous in performance and communication bandwidth. Considering the client population and based on the scale of the training cohort suggested by [32], the sampling fraction C is set to 0.2, 0.05 and 0.5 for the three tasks, respectively. We use part of the source data as the reference set D_V. More experimental settings are detailed in Appendix B.2.

Time and energy costs are two key indicators in the evaluation. Apart from the costs of local training and device-server communication, for FedProf we additionally consider the time and energy consumed in generating and uploading profiles. The overhead of our solution boils down to the size of representation profiles. Recall that the size of a profile is independent of the volume and dimensionality of the source data. As formulated in Section 4, each profile is a plain array of distribution parameters which is ultra-light for storage (for instance, 192 Bytes for ShuffleNetv2 in our experiment) and thus incurs minimal traffic in transmission. More details on cost formulation are provided in Appendix B.3.

6.2 Empirical Results

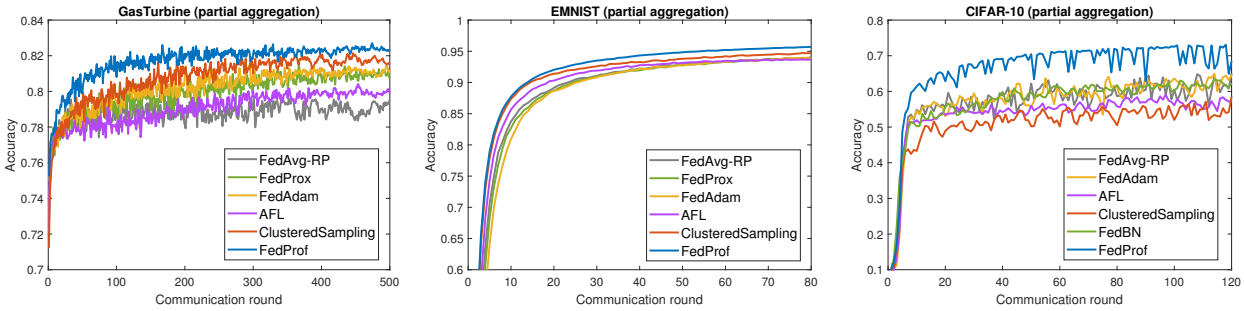
We evaluated the performance of our FedProf algorithm in terms of the efficacy (i.e., best accuracy achieved) and efficiency (i.e., costs for convergence) in establishing a global model for the three tasks. Tables 1, 2 and 3 report the averaged results (including best accuracy achieved and convergence costs given a preset accuracy goal) of multiple runs with standard deviations. In Figs. 5 and 6 we plot the

6. http://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set
7. https://www.nist.gov/itl/products-and-services/emnist-dataset



(a) GasTurbine (b) EMNIST (c) CIFAR-10

Fig. 5: The traces of the global model's accuracy (full aggregation group).



(a) GasTurbine (b) EMNIST (c) CIFAR-10

Fig. 6: The traces of the global model's accuracy (partial aggregation group).

traces of the test accuracy of the global model for the full and partial aggregation groups, respectively.

1) Convergence in different aggregation modes: Our results show a great difference in convergence rate under different aggregation modes. From Figs. 5 and 6, we observe that partial aggregation facilitates faster convergence of the global model than full aggregation, which is consistent with the observations made by Li et al. [50]. The advantage is especially obvious on EMNIST where FEDAVG-RP requires <30 communication rounds to reach the 90% accuracy whilst the standard FEDAVG needs 100+. On all three tasks, our FEDPROF algorithm significantly improves the convergence speed in both groups of comparison especially for the full aggregation mode.

2) Best accuracy of the global model: Throughout the FL training process, the global model was evaluated each round on the server who kept track of the best accuracy achieved. As shown in the 2nd column of Tables 1, 2 and 3, our FEDPROF algorithm achieved up to 1.8%, 1.7% and 6.8% accuracy improvement over the baselines FEDAVG and FEDAVG-RP on GasTurbine, EMNIST and CIFAR-10, respectively. We observe a significant gap of accuracy on CIFAR-10 where the federated model converged at 73.5% accuracy using FEDPROF, whilst none of the baseline algorithms reached over 68%. This implies that the training of deep models such as ShuffleNet is more sensitive to the quality of data. In table 3 we also notice that localized batch normalization (FEDBN) cannot improve the convergence or accuracy of the federated model.

3) Communication rounds for convergence: The number of communication rounds required for reaching convergence is a key efficiency indicator. On GasTurbine, our algorithm took *less than half* the communication rounds required by other algorithms in most cases. On EMNIST, our algorithm reached 90% accuracy within 60 rounds whilst FEDAVG and CFCFM needed more than 100 with full aggregation. CLUSTERED SAMPLING and AFL select clients based on model similarities and loss feedback, respectively. They showed competitive convergence speed (but lower accuracy) on EMNIST but failed to reach the 60% accuracy mark on CIFAR-10. A possible explanation is that neither of them is consistently effective in assessing the informativeness of data, which will misguide the model update to local optima.

4) Total time needed for convergence: The overall time consumption is closely related to total communication rounds needed for convergence and the time cost for each round. Algorithms requiring more rounds to converge typically take longer to reach the accuracy target except the case of CFCFM, which priorities the clients that work faster. By contrast, our algorithm accelerates convergence by addressing the heterogeneity of data and data quality, providing a 2.1× speedup over FEDAVG on GasTurbine and 2.4× speedup over FEDAVG-RP on CIFAR-10. Our strategy also has a clear advantage over CFCFM, FEDADAM and AFL for all three tasks, yielding a significant reduction (up to 65.7%) in the wall-clock time consumption until convergence.

5) Energy costs to end devices: A main concern for the end devices, as the participants of FL, is their power usage (in training and communications). Full aggregation methods experience slower convergence and thus endure higher energy cost on the devices. For example, with a small fraction

TABLE 1: Summary of the evaluation results on *GasTurbine*. The best accuracy is achieved by running for long enough. Other metrics are recorded upon the global model reaching the 0.8 accuracy mark. Standard deviations of multiple runs are shown after the ± symbol.

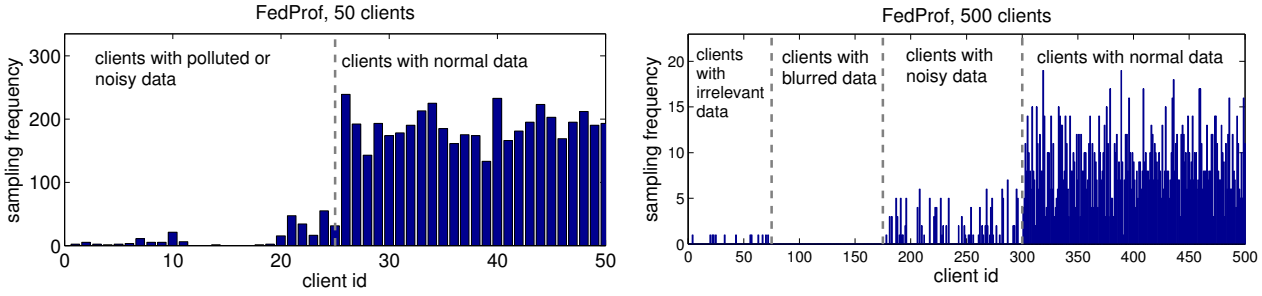
	Best accuracy	For accuracy@0.8		
		Rounds needed	Time (minutes)	Energy (Wh)
GasTurbine (full aggregation)				
FedAvg	0.817±0.005	82±73	47.7±42.3	4.59±4.11
CFCFM	0.809±0.006	167±147	68.9±60.8	8.15±7.17
FedProf (ours)	**0.832±0.005**	**38±23**	**22.3±13.7**	**2.15±1.29**
GasTurbine (partial aggregation)				
FedAvg-RP	0.820±0.007	28±11	16.8±7.1	1.62±0.68
FedProx	0.829±0.003	35±19	20.3±11.4	1.78±0.98
FedADAM	0.828±0.006	46±35	27.2±21.2	2.59±1.98
AFL	0.818±0.003	54±51	30.2±27.1	2.99±2.81
ClusteredSampling	0.826±0.004	58±28	34.5±18.4	3.29±1.62
FedProf (ours)	**0.838±0.005**	**19±9**	**11.0±5.5**	**1.07±0.53**

TABLE 2: Summary of the evaluation results on *EMNIST*. The best accuracy is achieved by running for long enough. Other metrics are recorded upon the global model reaching the 0.9 accuracy mark. Standard deviations of multiple runs are shown after the ± symbol.

	Best accuracy	For accuracy@0.9		
		Rounds needed	Time (minutes)	Energy (Wh)
EMNIST (full aggregation)				
FedAvg	0.923±0.004	103±13	115.8±16.2	15.83±2.04
CFCFM	0.918±0.008	136±42	**46.2±14.5**	15.02±4.59
FedProf (ours)	**0.940±0.004**	**59±5**	67.1±12.4	**9.49±0.66**
EMNIST (partial aggregation)				
FedAvg-RP	0.941±0.003	23±3	26.5±4.3	3.60±0.37
FedProx	0.941±0.005	23±3	27.7±6.8	3.69±0.70
FedADAM	0.941±0.003	26±3	29.1±4.3	3.99±0.47
AFL	0.939±0.005	19±4	22.4±6.8	2.93±0.57
ClusteredSampling	0.947±0.003	15±3	17.7±5.6	**2.31±0.48**
FedProf (ours)	**0.957±0.003**	15±1	**16.1±3.3**	2.43±0.25

TABLE 3: Summary of the evaluation results on *CIFAR-10*. The best accuracy is achieved by running for long enough. Other metrics are recorded upon the global model reaching the 0.6 accuracy mark. Standard deviations of multiple runs are shown after the ± symbol.

	Best accuracy	For accuracy@0.6		
		Rounds needed	Time (minutes)	Energy (Wh)
CIFAR (full aggregation)				
FedAvg	0.665±0.011	38±10	91.4±26.3	93.52±17.14
CFCFM	0.638±0.008	32±5	74.0±8.7	79.90±15.18
FedProf (ours)	**0.733±0.003**	**14±1**	**38.8±3.7**	**39.67±1.14**
CIFAR (partial aggregation)				
FedAvg-RP	0.674±0.004	24±2	56.7±3.5	59.73±9.88
FedADAM	0.669±0.014	28±2	68.5±4.9	68.96±5.10
AFL	0.599±0.007	-	-	-
ClusteredSampling	0.591±0.009	-	-	-
FedBN	0.645±0.005	44±6	105.4±18.4	108.68±6.70
FedProf (ours)	**0.735±0.007**	**9±1**	**23.5±1.29**	**24.54±1.11**



(a) GasTurbine (b) EMNIST

Fig. 7: The frequency that clients get sampled by the proposed algorithm throughout the training on GasTurbine and EMNIST. For clarity, clients are indexed according to their local data quality.

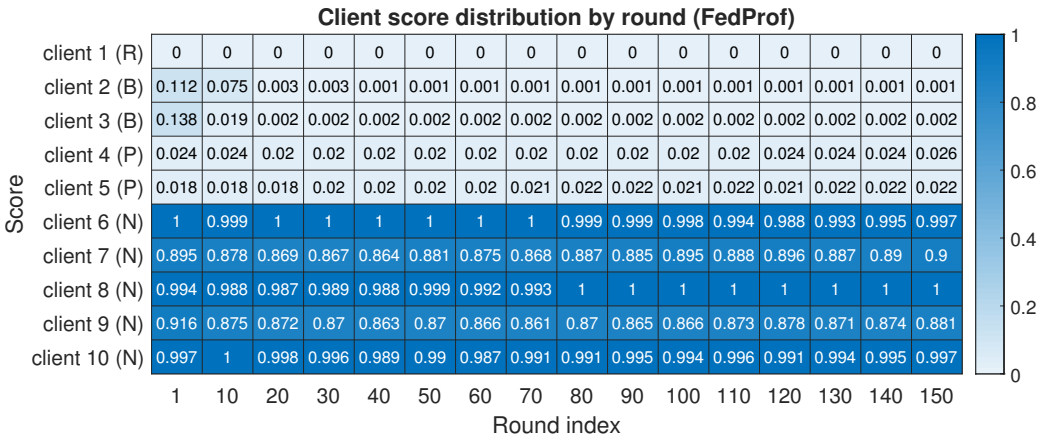


Fig. 8: A heatmap illustrating the dynamic distribution of client scores, i.e., normalized λ, for FEDPROF on CIFAR-10. The annotations 'R', 'B' or 'P' indicate the possession of image data that are irrelevant, blurred or affected by pixel-level noises whilst 'N' means that of normal data. All the data are non-IID across the clients.

TABLE 4: Average time and device energy costs of generating and transmitting representation profiles for FedProf.

	GasTurbine	EMNIST	CIFAR
Time cost/round (s)	0.134	1.340	17.735
Energy cost/round (Wh)	4.24e-5	6.74e-3	3.99e-1

(C=0.05) and a large scale (N=500) for the EMNIST task, FEDAVG and CFCFM consumed over 15 Wh to reach the target accuracy, in which case our algorithm reduced the energy cost by over 37%. Under the CIFAR-10 FL setting with partial aggregation, the reduction by our algorithm reached 58.9% and 64.4%—saving over 35 Wh—as compared to FEDAVG-RP and FEDADAM, respectively. In Table 4 we show the overhead of our solution which comes from local profiling and the upload of profiles. This part of cost is minimal when compared to that of FL.

6) Differentiated participation with FEDPROF: In figs. 7 and 8 we visualize the behavior of our sampling strategy. Fig. 7 shows how many times clients were selected under the GasTurbine and EMNIST settings. We can observe that the clients with polluted samples or noisy data were significantly less involved in the regression task (GasTurbine). On EMNIST, our algorithm also effectively limited (basically excludes) the clients who hold image data of poor quality (i.e., irrelevant or severely blurred), whereas the clients with moderately noisy images were selected with reduced frequency as compared to those with normal data. Fig. 8 reveals the *distribution of client scores* by round throughout the training process on CIFAR-10. Our strategy effectively avoided the low-value clients since the initial round, demonstrating that our profiling and matching scheme can provide strong evidence of local data's quality and informativeness. A potential issue of having the preference towards some of the devices is about fairness. Nonetheless, one can apply our algorithm together with an incentive mechanism (e.g., [58]) to address it.

7 CONCLUSION

Federated learning provides a privacy-preserving approach to decentralized training but is vulnerable to the heterogeneity and uncertain quality of on-device data. In this paper, we use a novel approach to address the issue without violating the data locality restriction. We first provide key insights into the distribution of data representations and then develop a dynamic data representation profiling and matching scheme, which provides a new view of data in the representation space and allows for fast, secure information exchange and comparison. Based on the scheme we propose a selective FL algorithm that features adaptive sampling based on profile dissimilarity. We have conducted extensive experiments on public datasets under various FL settings. Evaluation results show that our algorithm significantly improves the efficiency of FL and the global model's accuracy whilst reducing the time and energy costs for the global model to converge.

Our future study may involve extending our selective strategy to work with client reputation management and incentive mechanisms. An important direction is to consider dynamic data scenarios where the size, quality and distribution of local data vary over time.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (61872150, 62072187), PCL Major Project (PCL2021A09), Key-Area Research and Development Program of Guangdong Province(2021B0101420002), Guangdong Major Project of Basic and Applied Basic Research (2019B030302002), Guangdong Marine Economic Development Special Fund Project(GDNRC[2022]17), and Guangzhou Development Zone Science and Technology Project (2021GH10, 2020GH10).

REFERENCES

[1] Robert van der Meulen. What edge computing means for infrastructure and operations leaders. https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders/, 2018. Accessed: 2021-07-10.

[2] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282. PMLR, 2017.

[3] Zhenyi Lin, Xiaoyang Li, Vincent KN Lau, Yi Gong, and Kaibin Huang. Deploying federated learning in large-scale cellular networks: Spatial convergence analysis. *IEEE Transactions on Wireless Communications*, 2021.

[4] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019.

[5] Shashi Raj Pandey, Nguyen H. Tran, Mehdi Bennis, Yan Kyaw Tun, Aunas Manzoor, and Choong Seon Hong. A crowdsourcing framework for on-device federated learning. *IEEE Transactions on Wireless Communications*, 19:3241–3256, 2020.

[6] Tiffany Tuor, Shiqiang Wang, Bong Jun Ko, Changchang Liu, and Kin K. Leung. Overcoming noisy and irrelevant data in federated learning. *arXiv preprint arXiv:2001.08300*, 2020.

[7] Wenchao Xia, Wanli Wen, Kai-Kit Wong, Tony QS Quek, Jun Zhang, and Hongbo Zhu. Federated-learning-based client scheduling for low-latency wireless communications. *IEEE Wireless Communications*, 28(2):32–38, 2021.

[8] Zachary Charles, Zachary Garrett, Zhouyuan Huo, Sergei Shmulyian, and Virginia Smith. On large-cohort training for federated learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[9] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[10] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[11] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[12] Jaejun Yoo, Namhyuk Ahn, and Kyung-Ah Sohn. Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2020.

[13] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.

[14] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. How does data augmentation affect privacy in machine learning? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10746–10753, 2021.

[15] Zhao Zhang, Yan Zhang, Mingliang Xu, Li Zhang, Yi Yang, and Shuicheng Yan. A survey on concept factorization: From shallow to deep representation learning. *Information Processing & Management*, 58(3):102534, 2021.

[16] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021.

[17] Siwei Feng and Han Yu. Multi-participant multi-class vertical federated learning. *arXiv preprint arXiv:2001.11154*, 2020.

[18] Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated unsupervised representation learning. *arXiv preprint arXiv:2010.08982*, 2020.

[19] Bram van Berlo, Aaqib Saeed, and Tanir Ozcelebi. Towards federated unsupervised representation learning. In *Proceedings of the third ACM international workshop on edge systems, analytics and networking*, pages 31–36, 2020.

[20] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *International Conference on Machine Learning*, pages 5325–5333. PMLR, 2018.

[21] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation. In *International Conference on Machine Learning*, pages 4120–4129. PMLR, 2017.

[22] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[23] Laizhong Cui, Xiaoxin Su, Yipeng Zhou, and Yi Pan. Slashing communication traffic in federated learning by transmitting clustered model updates. *IEEE Journal on Selected Areas in Communications*, 2021.

[24] Chunmei Xu, Shengheng Liu, Zhaohui Yang, Yongming Huang, and Kai-Kit Wong. Learning rate optimization for federated learning exploiting over-the-air computation. *IEEE Journal on Selected Areas in Communications*, 39(12):3742–3756, 2021.

[25] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *The 3rd MLSys Conference*, 2020.

[26] Wentai Wu, Ligang He, Weiwei Lin, and Rui Mao. Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1539–1551, 2021.

[27] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. Federated learning for keyword spotting. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6341–6345. IEEE, 2019.

[28] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, pages 3581–3607. PMLR, 2022.

[29] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen A Jarvis. Safa: a semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 70(5):655–668, 2021.

[30] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.

[31] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *arXiv preprint arXiv:2103.00710*, 2021.

[32] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[33] Haibo Yang, Minghong Fang, and Jia Liu. Achieving linear speedup with partial worker participation in non-iid federated learning. In *International Conference on Learning Representations*, 2020.

[34] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020.

[35] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2020.

[36] 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.

[37] Mohammad Mohammadi Amiri and Gunduz Deniz. Federated learning over wireless fading channels. *IEEE Transactions on Wireless Communications*, 19(5):3546–3557, 2020.

[38] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[39] Mingzhe Chen, Nir Shlezinger, H Vincent Poor, Yonina C Eldar, and Shuguang Cui. Communication-efficient federated learning. *Proceedings of the National Academy of Sciences*, 118(17), 2021.

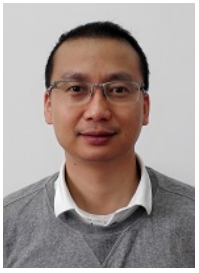
[40] Wenqi Shi, Sheng Zhou, and Zhisheng Niu. Device scheduling with fast convergence for wireless federated learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020.

[41] Mingzhe Chen, H Vincent Poor, Walid Saad, and Shuguang Cui. Convergence time optimization for federated learning over wireless networks. *IEEE Transactions on Wireless Communications*, 20(4):2457–2471, 2020.

[42] Jack Goetz, Kshitiz Malik, Duc Bui, Seungwhan Moon, Honglei Liu, and Anuj Kumar. Active federated learning. *arXiv preprint arXiv:1909.12641*, 2019.

[43] 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.
[44] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. Tifl: A tier-based federated learning system. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, pages 125–136, 2020.
[45] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, pages 2089–2099. PMLR, 2021.
[46] Don S Lemons. *An Introduction to Stochastic Processes in Physics*. Johns Hopkins University Press, 2003.
[47] Patrick Billingsley. *Probability and measure*. John Wiley & Sons, 2008.
[48] Stephen J Roberts and Will D Penny. Variational bayes for generalized autoregressive models. *IEEE Transactions on Signal Processing*, 50(9):2245–2257, 2002.
[49] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625. PMLR, 2019.
[50] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2019.
[51] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in Neural Information Processing Systems*, 31:4447–4458, 2018.
[52] Yuchen Zhang, John C Duchi, and Martin J Wainwright. Communication-efficient algorithms for statistical optimization. *The Journal of Machine Learning Research*, 14(1):3321–3363, 2013.
[53] Jonathan Goetz. *Active Learning in Non-parametric and Federated Settings*. PhD thesis, University of Michigan, 2020.
[54] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In *International Conference on Machine Learning*, pages 3407–3416. PMLR, 2021.
[55] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.
[56] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[57] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.
[58] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. A fairness-aware incentive scheme for federated learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 393–399, 2020.
[59] 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.
[60] Nguyen H Tran, Wei Bao, Albert Zomaya, Minh NH Nguyen, and Choong Seon Hong. Federated learning over wireless networks: Optimization model design and analysis. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1387–1395. IEEE, 2019.
[61] Jie Song, Tiantian Li, Zhi Wang, and Zhiliang Zhu. Study on energy-consumption regularities of cloud computing systems by a novel evaluation model. *Computing*, 95(4):269–287, 2013.
[62] Aaron Carroll, Gernot Heiser, et al. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*, volume 14, pages 21–21. Boston, MA, 2010.
[63] Pijush Kanti Dutta Pramanik, Nilanjan Sinhababu, Bulbul Mukherjee, Sanjeevikumar Padmanaban, Aranyak Maity, Bijoy Kumar Upadhyaya, Jens Bo Holm-Nielsen, and Prasenjit Choudhury. Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage. *IEEE Access*, 7:182113–182172, 2019.
[64] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

Wentai Wu received his Bachelor and Master degrees in computer science from South China University of Technology in 2015 and 2018, respectively. Sponsored by CSC, he received the Ph.D. degree in Computer Science in 2022 from the University of Warwick, United Kingdom. He currently works as an assistant researcher at PCL. His research interests mainly include distributed systems, distributed machine learning and sustainable computing. He serves as reviewer for multiple high-impact journals and conferences such as IEEE TPDS, IEEE TMC, ICML and NeurIPS.



Ligang He received the Ph.D. degree in Computer Science at the University of Warwick, United Kingdom, and worked as a post-doctoral researcher at the University of Cambridge, UK. He then joined the Department of Computer Science at the University of Warwick as Assistant Professor, and was then promoted to Associate Professor. He is now a Reader in the department. He has published over 150 articles in journals and conferences, such as IEEE TC, TPDS, TACO, IPDPS, ICPP, SC, VLDB. His research interests focus on parallel and distributed processing, and big data processing.

Weiwei Lin received his B.S. and M.S. degrees from Nanchang University in 2001 and 2004, respectively, and the PhD degree in Computer Application from South China University of Technology in 2007. Currently, he is a professor with the School of Computer Science and Engineering, South China University of Technology. His research interests include distributed systems, cloud computing, big data computing and AI application technologies. He has published more than 150 papers in refereed journals and conference proceedings. He has been the reviewers for many international journals including TC, TCYB, TSC, TCC, INS, FGCS. He is a senior member of CCF and a member of the IEEE.

Carsten Maple is a Professor of Cyber Systems Engineering in WMG, University of Warwick, and a Principal Investigator of the NCSC-EPSRC Academic Center of Excellence in Cyber Security Research at the University. He is the Transport & Mobility leader of the PETRAS National Center of Excellence for IoT Systems Cybersecurity. He has published over 200 peer-reviewed papers and provided evidence and advice to governments and organizations across the world, including being a high-level scientific advisor for cyber security to the European Commission. He is a member of various national and international boards and expert groups and is a fellow of the Alan Turing Institute. Carsten is an executive committee member of the UK-RAS Network and the IoT Security Foundation, and an expert group member on automotive security for ENISA and Interpol.

APPENDIX A
PROOF OF PROPOSITIONS

A.1 Proof of Proposition 1

Without loss of generality, we provide the proof of Proposition 4.2 for the pre-activation representations from dense (fully-connected) layers and standard convolutional layers, respectively. The results can be easily extended to other linear neural operators.

For fully-connected (dense) layers, please refer to the proof provided in Section 4.

Convolutional layers

Proof. Standard convolution in CNNs is also a linear transformation of the input feature space and its main difference from dense layers rests on the restricted size of receptive field. Without loss of generality, we analyze the representation (output) of a single kernel. To facilitate our analysis for convolutional layers, let C denote the number of input channels and K denote the kernel size. For ease of presentation, we define a receptive field mapping function $\Theta(k,i,j)$ that maps the positions (k for channel index, i and j for indices on the same channel) of elements in the feature map (i.e., the representations) to the input features. For the k-th kernel, let W_k denote its weight tensor (with $W_{k,c}$ being the weight matrix for channel c) and b_k its bias.

Given the corresponding input patch $X_{\Theta(k,i,j)}$, The element $H_{k,i,j}$ of the representations from a convolutional layer can be formulated as:

$$H_{k,i,j} = \sum_{c=1}^{C}\sum_{i'=1}^{K}\sum_{j'=1}^{K}\left(X_{\Theta(k,i,j)} \circ W_{k,c}\right)_{i',j'} + b_k, \tag{15}$$

where $\circ$ denotes Hadamard product. The three summations reduce the results of element-wise product between the input patch and the k-th kernel to the correspond representation element $H_{k,i,j}$ in the feature map. For ease of presentation, here we use the notation $Z^{(k)}_{c,i',j'}$ to replace $\left(X_{\Theta(k,i,j)} \circ W_{k,c}\right)_{i',j'}$ and let $\zeta(\mu_{c,i',j'}, \sigma^2_{c,i',j'})$ be the distribution that $Z^{(k)}_{c,i',j'}$ follows. Note that ζ can be any distribution since we do not make any distributional assumption on $Z^{(k)}_{c,i',j'}$.

With the notations, Eq. (15) can be rewritten in a similar form to Eq. (2):

$$H_{k,i,j} = \sum_{c=1}^{C}\sum_{i'=1}^{K}\sum_{j'=1}^{K} Z^{(k)}_{c,i',j'} + b_k. \tag{16}$$

We use the condition that the random variables $Z^{(k)}_{c,i',j'}$ satisfy the Lyapunov's condition, i.e., there exists a δ such that

$$\lim_{C\times K^2\to\infty} \frac{1}{s^{2+\delta}} \sum_{c=1}^{C}\sum_{i'=1}^{K}\sum_{j'=1}^{K} \mathrm{E}\left[\left|Z^{(k)}_{c,i',j'} - \mu_{c,i',j'}\right|^{2+\delta}\right] = 0, \tag{17}$$

where $s = \sqrt{\sum_{c=1}^{C}\sum_{i'=1}^{K}\sum_{j'=1}^{K}\sigma^2_{c,i',j'}}$.

Then according to the Lyapunov CLT, the following holds:

$$H_{k,i,j} \xrightarrow{d} \mathcal{N}\Big(\sum_{c,i',j'\in\Theta(k,i,j)} \mu_{c,i',j'} + b_k, \sum_{c,i',j'\in\Theta(k,i,j)} \sigma^2_{c,i',j'}\Big), \tag{18}$$

which proves our Proposition 4.2 for standard convolution layers. □

A.2 Proof of Proposition 2

Without loss of generality, we prove Proposition 4.3 for the fused representations from the LSTM layer and the residual block of ResNet models, respectively. The results can be easily extended to other non-linear neural operators.

LSTM

Proof. Long Short-Term Memory (LSTM) models are popular for extracting useful representations from sequence data for tasks such as speech recognition and language modeling. Each LSTM layer contains multiple neural units. For the k-th unit, it takes as input the current feature vector $X_t = (X_{t,1}, X_{t,2}, \ldots)$, hidden state vector H_{t-1} and its cell state $c_{t-1,k}$. The outputs of the unit are its new hidden state $h_{t,k}$ and cell state $c_{t,k}$. In this paper, we study the distribution of $h_{t,k}$. Multiple gates are adopted in an LSTM unit: by $i_{t,k}$, $f_{t,k}$, $g_{t,k}$ and $o_{t,k}$ we denote the input gate, forget gate, cell gate and output gate of the LSTM unit k at time step t. The update rules of these gates and the cell state are:

$$\begin{aligned}
i_{t,k} &= \mathrm{sigmoid}(W_{(i)k}[H_{t-1}, X_t] + b_{(i)k}),\\
f_{t,k} &= \mathrm{sigmoid}(W_{(f)k}[H_{t-1}, X_t] + b_{(f)k}),\\
g_{t,k} &= \tanh(W_{(g)k}[H_{t-1}, X_t] + b_{(g)k}),\\
o_{t,k} &= \mathrm{sigmoid}(W_{(o)k}[H_{t-1}, X_t] + b_{(o)k}),\\
c_{t,k} &= f_{t,k}\cdot c_{t-1,k} + i_{t,k}\cdot g_{t,k},
\end{aligned} \tag{19}$$

where the $W_{(i)k}$, $W_{(f)k}$, $W_{(g)k}$ and $W_{(o)k}$ are the weight parameters and $b_{(i)k}$, $b_{(f)k}$, $b_{(g)k}$ and $b_{(o)k}$ are the bias parameters for the gates.

The output of the LSTM unit $h_{t,k}$ is calculated as the following:

$$h_{t,k} = o_{t,k} \cdot \tanh(c_{t,k}). \tag{20}$$

Using the final hidden states $h_{T,k}$ (with T being the length of the sequence) as the elements of the layer-wise representation, we apply the following layer-wise fusion to further produce H over all the $h_{T,k}$ in a single LSTM layer:

$$H = \sum_{k=1}^{d} h_{T,k}, \tag{21}$$

where d is the dimension of the LSTM layer. Again, by $\zeta(\mu_k, \sigma_k^2)$ we denote the distribution of $h_{T,k}$ (where the notation T is dropped here since it is typically a fixed parameter). With $\{h_{T,k}|k = 1, 2, \ldots, d\}$ satisfying the Lyapunov's condition and by Central Limit Theorem, H converges in distribution to the normal distribution:

$$H \xrightarrow{d} \mathcal{N}(\sum_{i=1}^{d} \mu_k, \sum_{i=1}^{d} \sigma_k^2), \tag{22}$$

which proves the Proposition 4.3 for layer-wise fused representations from LSTM. □

Residual blocks

Proof. Residual blocks are the basic units in the Residual neural network (ResNet) architecture [59]. A typical residual block contains two convolutional layers with batch normalization (BN) and uses the ReLU activation function. The input of the whole block is added to the output of the second convolution (after BN) through a skip connection before the final activation. Since the convolution operators are the same as we formulate in the second part of Section A.1, here we use the notation $\Psi(X)$ to denote the sequential operations of convolution on X followed by BN, i.e., $\Psi(X) \triangleq BN(Conv(X))$. Again, we reuse the receptive field mapping $\Theta(k, i, j)$ as defined in Section A.1 to position the inputs of the residual block corresponding to the element $Z_{k,i,j}$ in the output representation of the whole residual block.

Let X denote the input of the residual block and $Z_{k,i,j}$ denote an element in the output tensor of the whole residual block. Then we have:

$$\begin{aligned} Z_{k,i,j} &= f\Big(X_{k,i,j} + BN\big(Conv\big(f\big(BN(Conv(X_{\Theta(k,i,j)}))\big)\big)\big)\Big) \\ &= f\Big(X_{k,i,j} + \Psi\big(f(\Psi(X_{\Theta(k,i,j)}))\big)\Big), \end{aligned} \tag{23}$$

where f is the activation function (ReLU).

We perform channel-wise fusion on the representation from the residual block to produce H_k for the k-th channel:

$$H_k = \sum_{i=1}^{d_H} \sum_{j=1}^{d_W} Z_{k,i,j}, \tag{24}$$

where d_H and d_W are the dimensions of the feature map and k is the channel index.

Let $\zeta(\mu_{k,i,j}, \sigma_{k,i,j}^2)$ denote the distribution that $Z_{k,i,j}$ follows. Then we apply the Lyapunov's condition to the representation elements layer-wise, i.e.,

$$\lim_{d_W \times d_H \to \infty} \frac{1}{s_k^{2+\delta}} \sum_{i=1}^{d_H} \sum_{j=1}^{d_W} \mathrm{E}\left[|Z_{k,i,j} - \mu_{k,i,j}|^{2+\delta}\right] = 0, \tag{25}$$

where $s_k = \sqrt{\sum_{i=1}^{d_H} \sum_{j=1}^{d_W} \sigma_{k,i,j}^2}$.

With the above condition satisfied, by CLT the distribution of H_k (the fused representation on channel k) converges to the normal distribution:

$$H_k \xrightarrow{d} \mathcal{N}(\sum_{i=1}^{d_H} \sum_{j=1}^{d_W} \mu_{k,i,j}, \sum_{i=1}^{d_H} \sum_{j=1}^{d_W} \sigma_{k,i,j}^2), \tag{26}$$

which proves the Proposition 4.3 for channel-wise fused representations from any residual block. □

APPENDIX B
DETAILS OF THE EXPERIMENTS

B.1 Preliminary Experiments

(Preliminary experiment: the impact of data quality on the convergence of FL) We setup a simulation-based FL system based on PyTorch (Build 1.7.0) to train a LeNet-5 model on MNIST for demonstration purpose. We set up 100 clients with a sampling fraction of 0.3, i.e., cohort size=30. For local training, SGD with momentum is used as the local optimizer. We set the batch size to 32 and learning rate to 0.01 with a decay of 0.99 per round. The global aggregation is performed every 5 epochs. In the figure we trace the test accuracy of the global model.

We experimented under both IID (Identical and Independently Distributed) and non-IID data settings. More specifically, we partitioned the MNIST dataset into 100 subsets in four different ways:

1) *IID data* (black curve in Fig. 1): all the data are noiseless and evenly distributed across the clients with identical class distribution. This represents the best quality condition of local data.
2) *Non-IID data* (magenta curve in Fig. 1): by re-partitioning the data we force class imbalance for each local dataset. In this case, each client has specific affinity to a particular class of samples, and in each local dataset the dominant class accounts for >50% of the samples, which results in a strong class imbalance locally and statistically heterogeneity globally.
3) *Low-quality data* (blue curve in Fig. 1): In this case we mix low-quality samples into local data by means of applying the blurring filter and pixel noise to the source data or replacing them with task-irrelevant synthesized samples. Low-quality data cover 65% of the clients—15% with irrelevant images, 25% with blurred images, and 25% with images affected by "salt and pepper" noise.
4) *Non-IID and low-quality data* (red curve in Fig. 1): all local datasets are non-IID (i.e., as in the *biased* case) while 65% of them are filled with low-quality data (as in the *noisy* case). This represents the worst condition of data in our preliminary experiments.

(Preliminary experiment: the distribution of representations) We also ran preliminary experiment to demonstrate the distribution of representations learned during/after gradient-based training on MNIST (Fig. 4) and CIFAR-100 (Fig. 3). We adopt the standard implementations of LeNet-5 (from PyTorch tutorial[8]) and ResNet-18 (from Github repository[9]) and use the default data sources and data loaders in PyTorch. For representation extraction and profiling, we define a structured tensor (initialized with the model) that caches the representations in the forward pass.

We also empirically found that *small-scale datasets* can also exhibit normal distribution patterns in the representation space using a simple model as the encoder (similar to the role of the global model in FL). Fig. 9 shows the data representations from the two hidden layers of a feed-forward network model after training (till convergence) on the Boston Housing dataset[10]. Boston Housing is a textbook dataset for regression modeling. The dataset features 14 attributes (avg. rooms per dwelling, crime rate, etc.) and we use the median of house price in the corresponding area as the prediction target. The dataset is small containing only 506 samples. From Fig. 9 one can observe clear patterns of normal distribution over the neurons' output. It is worth noticing that the raw data is not normally distributed (dimension-wise), as visualized in Fig. 10.

Moreover, Fig. 9 also exhibits the distributional distances between the representations from the training set (in blue) and those from the test set (in orange), which in a sense supports our hypothesis that the difference of raw data distribution can be examined in the representation space.

B.2 Evaluation Setup

Our evaluation environment is a simulation-based, discrete event-driven system built using Python 3.7 and PyTorch (Build: 1.7.0) on a dual-core (CPU model: Intel i5-8500) physical machine equipped with Nvidia GeForce GTX 1050Ti graphics card. The client-side local training logic was implemented under the PyTorch framework with CUDA (version: 11.1) support. Experimental statistics were collected through event processing. End devices were initialized with heterogeneous performance specifications and heterogeneous local datasets. A star topology was established for this system. The client–server communication was implemented via in-memory tensor exchange. The server maintains a simulated wall clock and uses the equations provided in Section B.3 to calculate all the consumptions.

Detailed experimental settings are listed in Table 5. For GasTurbine, the total population is 50 and the data collected by a proportion of the sensors (i.e., end devices of this task) are of low-quality: 10% of the sensors are polluted (with features taking invalid values) and 40% of them produce noisy data. For EMNIST, we set up a relatively large population (500 end devices) and spread the data (from the *digits* subset) across the devices with strong class imbalance—roughly 60% of the samples on each device fall into the same class. Besides, many local datasets are of low-quality: the images on 15% of the clients are irrelevant (valueless for the training of this task), 20% are (Gaussian) blurred, and 25% are affected by the salt-and-pepper noise (random black and white dots on the image, density=0.3). For CIFAR-10, same types of noise are applied

8. https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html
9. https://github.com/weiaicunzai/pytorch-cifar100/blob/master/models/resnet.py
10. https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

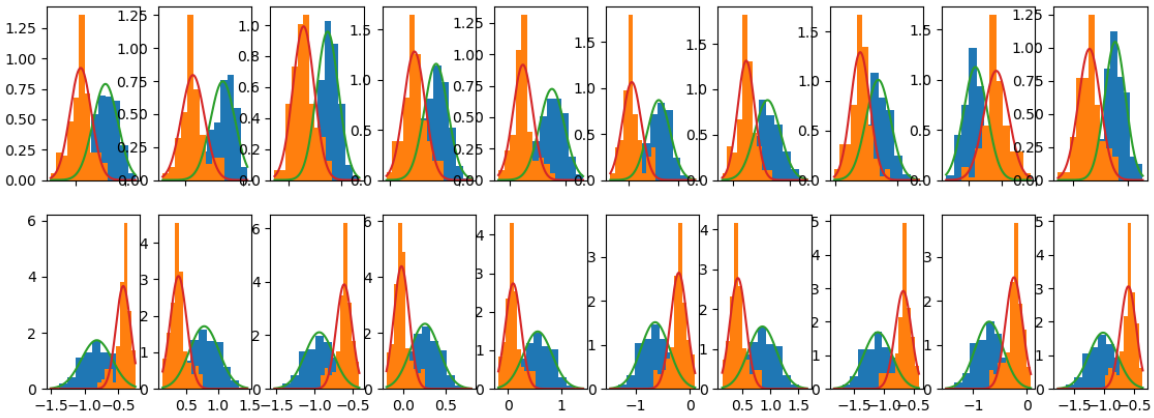


Fig. 9: An example of training a two-layer FNN (each layer corresponding to one row in the figure) on the Boston Housing dataset till convergence. Depicted here are representations (blue for the training set, orange for the test set) extracted from the model's hidden layers in the form of density histograms.

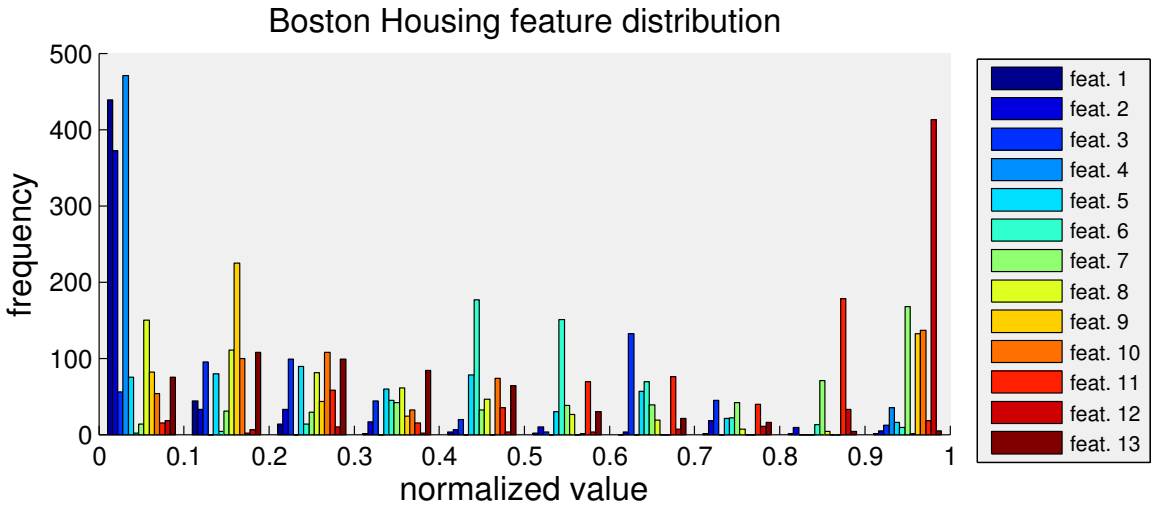


Fig. 10: The dimension-wise distribution of the *raw features* in the Boston Housing dataset. Most of them display long-tail patterns before being encoded into the representation space.

but with lower percentages (10%, 20% and 20%) of clients affected. The class imbalance degree in local data distribution for the 10 clients is set to 37%, i.e., each local dataset is dominated by a single class that accounts for approximately 37% of the total size.

A sufficiently long running time is guarantee for all the three FL tasks. The maximum number of rounds T_{max} is set to 500 for the GasTurbine task. For EMNIST, T_{max} is set to 240 for the full aggregation and 80 for partial aggregation respectively considering their discrepancy in convergence speed. For CIFAR-10, T_{max} is set to 150 for the full aggregation and 120 for partial aggregation respectively, which are adequate for the global model to converge in our settings.

B.3 Details of Cost Formulation

In each FL round, the server selects a fraction (i.e., C) of clients, distributes the global model to these clients and waits for them to finish the local training and upload the models. Given a selected set of clients S, the time cost and energy cost of a communication round are:

$$T_{round} = \max_{k \in S} \{T_k^{comm} + T_k^{train} + T_k^{RP}\}, \tag{27}$$

$$E_k = E_k^{comm} + E_k^{train} + E_k^{RP}. \tag{28}$$

TABLE 5: Experimental setup.

Setting	Symbol	Task 1	Task 2	Task 3
Model	h_θ	MLP	LeNet-5	ShuffleNet v2
Dataset	D	GasTurbine	EMNIST digits	CIFAR-10
Total data size	$\|D\|$	36.7k	280k	60k
reference set size	$\|D_V\|$	11.0k	40k	10k
Client population	N	50	500	10
Data distribution	-	$\mathcal{N}(514, 101^2)$	non-IID, dominant≈60%	non-IID, dominant>30%
Noise applied	-	pollution, Gaussian noise	fake, blur, pixel	fake, blur, pixel
Client specification (GHz)	s_k	$\mathcal{N}(0.5, 0.1^2)$	$\mathcal{N}(1.0, 0.2^2)$	$\mathcal{N}(3.0, 0.4^2)$
Comm. bandwidth (MHz)	bw_k	$\mathcal{N}(0.7, 0.1^2)$	$\mathcal{N}(1.0, 0.3^2)$	$\mathcal{N}(2.0, 0.2^2)$
Signal-to-noise ratio	SNR	7 dB	10 dB	10 dB
Bits per sample	BPS	11*8*4	28*28*1*8	32*32*3*8
Cycles per bit	CPB	300	400	400
# of local epochs	E	2	5	6
Batch size	-	8, 32	32, 128	16, 32
Loss function	ℓ	MSE	NLL	CE
Learning rate	η	5e-3	5e-3	1e-2
learning rate decay	-	0.994	0.99	0.999

where T_k^{comm} and T_k^{train} are the communication time and local training time, respectively. The device-side energy consumption E_k mainly comes from model transmission (through wireless channels) and local processing (training), corresponding to E_k^{comm} and E_k^{train}, respectively. T_k^{RP} and E_k^{RP} estimate the time and energy costs for generating and uploading local profiles and only apply to FEDPROF.

The time cost for communication T_k^{comm} can be modeled by Eq. (29) according to [60], where bw_k is the downlink bandwidth of device k (in MHz); *SNR* is the Signal-to-Noise Ratio of the communication channel, which is set to be constant as in general the end devices are coordinated by the base stations for balanced *SNR* with the fairness-based policies; $msize$ is the size (in MB) of the (encrypted) model; the model upload time is twice as much as that for model download since the uplink bandwidth is set to 50% of the downlink bandwidth.

$$\begin{aligned} T_k^{comm} &= T_k^{upload} + T_k^{download} \\ &= 2 \times T_k^{download} + T_k^{download} \\ &= 3 \times \frac{msize}{bw_k \cdot \log(1+SNR)}, \end{aligned} \tag{29}$$

The local training time T_k^{train} is modeled in Eq. (30), where s_k is the device performance (in GHz) and the numerator computes the total number of processor cycles required for processing E epochs of local training on D_k.

$$T_k^{train} = \frac{E \cdot |D_k| \cdot BPS \cdot CPB}{s_k}, \tag{30}$$

T_k^{RP} consists of two parts: T_k^{RPgen} for generating the profile of D_k (through a forward pass) and T_k^{RPup} for uploading the profile. T_k^{RP} can be modeled as:

$$\begin{aligned} T_k^{RP} &= T_k^{RPgen} + T_k^{RPup} \\ &= \frac{1}{E} T_k^{train} + \frac{RPsize}{\frac{1}{2} bw_k \cdot \log(1+SNR)}, \end{aligned} \tag{31}$$

where T_k^{RPgen} is estimated as the time cost of one epoch of local training; T_k^{RPup} is computed in a similar way to the calculation of T_k^{comm} in Eq. (29) (where the uplink bandwidth is set as one half of the total bw_k); $RPsize$ is the size of a profile, which is equal to $4 \times 2 \times q = 8 \times q$ (four bytes for each floating point number) according to our definition of profile in Eq. (8).

Using Eq. (28) we model the energy cost of each end device by mainly considering the energy consumption of the transmitters for communication (Eq. 32) and on-device computation for local training (Eq. 33). For FEDPROF, there is an extra energy cost for generating and uploading profiles (Eq. 34).

$$E_k^{comm} = P_{trans} \cdot T_k^{comm} \tag{32}$$

$$E_k^{train} = P_f s_k^3 \cdot T_k^{train} \tag{33}$$

$$E_k^{RP} = P_{trans} \cdot T_k^{RPup} + P_f s_k^3 \cdot T_k^{RPgen}, \tag{34}$$

where $P_f s_k^3$ is a simplified computation power consumption model [61] and P_f is the power of a baseline processor. P_{trans} is the transmitter's power. We set P_{trans} and P_f to 0.75 W and 0.7 W respectively based on the benchmarking data provided by [62] and [63].

APPENDIX C
TECHNICAL DETAILS OF OUR SOLUTION

C.1 FedProf Workflow

Algorithm 2 presents a complete pseudo-code of the proposed training algorithm FEDPROF. Our algorithm needs to take a few steps for initialization (lines 1 to 4). First, the server initializes the global model and broadcasts the same seed to all the clients. This is to ensure an identical base point in the parameter space. Then, the server communicates with the clients to gather their initial representation profiles. These local profiles are used to "bootstrap" the algorithm for calculating profile dissimilarity (line 9) and scoring clients (line 10) in the very first round.

Algorithm 2 the FEDPROF algorithm (detailed version)

Input: maximum number of rounds T_{max}, local iterations per round τ, client set U, client fraction C, reference dataset D_V
Output: the global model θ

/* **Server process:** running on the server */
1: Initialize global model θ using a seed
2: $v \leftarrow 0$ ▷ version of the latest global model
3: Broadcast the seed to all clients for identical model initialization
4: Collect initial profiles $\{RP_k\}_{k\in U}$ from all the clients
5: $v_k \leftarrow 0$, $\forall k \in U$
6: Generate initial reference profile $RP_V(0)$ on D_V
7: $K \leftarrow |U| \cdot C$
8: **for** round $T \leftarrow 1$ **to** T_{max} **do**
9: Calculate $div(RP_k(v_k), RP_V(v_k))$ for each client k
10: Update client scores $\{\lambda_k\}_{k\in U}$ and compute $\Lambda = \sum_{k\in U} \lambda_k$
11: $S \leftarrow$ Choose K clients by probability distribution $\{\frac{\lambda_k}{\Lambda}\}_{k\in U}$
12: Distribute θ to the clients in S
13: **for** client k in S **in parallel do**
14: $v_k \leftarrow v$, $\forall k \in S$
15: $RP_k(v_k) \leftarrow$ UPDATEPROFILE(k, θ, v)
16: $\theta_k \leftarrow$ LOCALTRAINING(k, θ, τ)
17: **end for**
18: Collect local profiles from the clients in S
19: Update θ via model aggregation
20: $v \leftarrow T$
21: Evaluate h_θ and generate $RP_V(v)$
22: **end for**
23: **Return** θ

/* **Client process:** running on client k */
24: **procedure** UPDATEPROFILE(k, θ, v) ▷ Update a local profile
25: Generate RP_k on D_k with the global θ received
26: Label profile RP_k with version number v
27: **Return** RP_k
28: **end procedure**
29: **procedure** LOCALTRAINING(k, θ, τ) ▷ Local model training
30: $\theta_k \leftarrow \theta$
31: **for** step $i \leftarrow 1$ **to** τ **do**
32: Update θ_k using gradient-based method
33: **end forReturn** θ_k
34: **end procedure**

In practice, we tag the global model with a version number (line 2, Algorithm 2). This is because profiles need to be compared in a consensus representation space whilst the global model, as the data encoder, is evolving throughout the FL process. The data representation profiles are associated to the global model and thus change over the rounds. Thus we tag every profile (including local and reference profiles) with the version number of the global model so that profile comparisons are version-aligned.

As the key of our algorithm, the server needs to take four steps in the stage of client sampling:

1) Calculate $\{\lambda_k\}_{k\in U}$ according to Eq. (13) to score each client according to the dissimilarity between local profiles and the reference profile.
2) Select $n \cdot C$ clients using the weighted random sampling function with λ_k being client k's weight. More specifically, given the sampling fraction C (or cohort size $N \cdot C$), we use a weighted random sampling method, e.g., the

`random.choices(U, weights, N*C)` function provided by the Python Standard Library, to sample a subset S of $N \cdot C$ participants from the client set U. We use λ_k as client k's weight in the sampling.

3) Collect the updated local profiles from the clients in S. Only the selected clients are required to update and upload local profiles in each round.
4) Update the reference profile RP_V after global aggregation. If the reference dataset D_V is kept by the server, then the server is responsible for updating the reference profile. Otherwise the owner of D_V updates the reference profile upon receiving the global model from the server.

Fig. 11 illustrates the main workflow of the proposed algorithm.

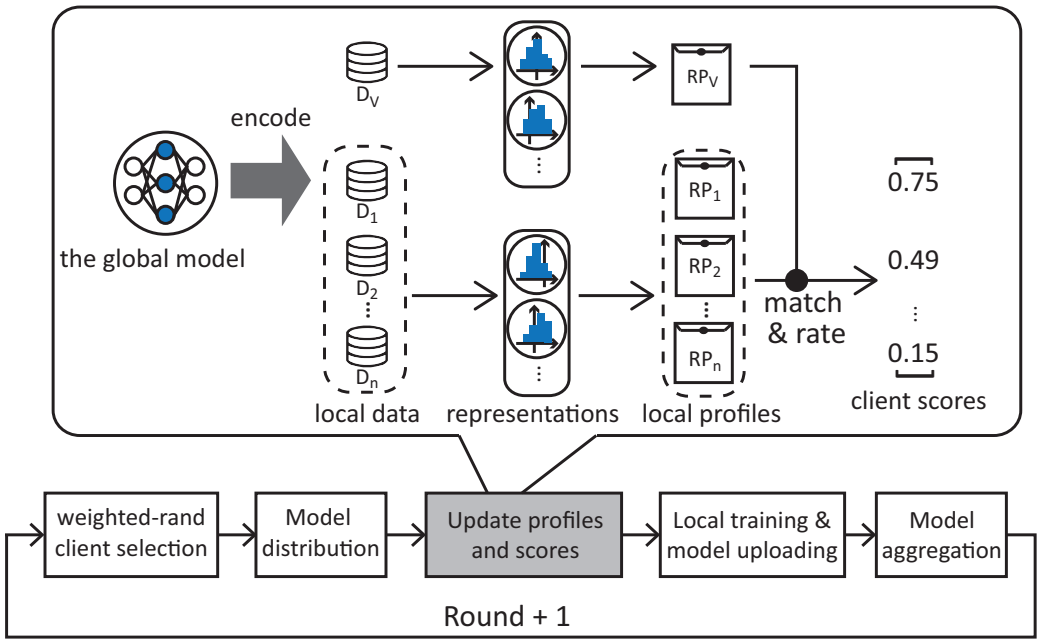


Fig. 11: The workflow of the proposed FEDPROF algorithm.

C.2 Profile Dissimilarity under Homomorphic Encryption

The proposed representation profiling scheme encodes the representations of data into a list of distribution parameters, namely $RP(\theta, D) = \{(\mu_i, \sigma_i^2) | i = 1, 2, \ldots, q\}$ where q is the length of the profile. Theoretically, the information leakage (in terms of the data in D) by exposing $RP(\theta, D)$ is very limited and it is basically impossible to reconstruct the samples in D given $RP(\theta, D)$. Nonetheless, Homomorphic Encryption (HE) can be applied to the profiles (both locally and on the server) so as to guarantee zero knowledge disclosure while still allowing profile matching under the encryption. In the following we give details on how to encrypt a representation profile and compute profile dissimilarity under Homomorphic Encryption (HE).

To calculate (9) and (10) under encryption, a client needs to encrypt (denoted as $[[\cdot]]$) every single μ_i and σ_i^2 in its profile $RP_k(\theta, D_k)$ locally before upload whereas the server does the same for its $RP_V(\theta, D_V)$. Therefore, according to Eq. (10) we have:

$$\begin{aligned}[[\mathrm{KL}(\mathcal{N}_i^{(k)} || \mathcal{N}_i^V)]] =& \frac{1}{2}\log[[(\sigma_i^V)^2]] - \frac{1}{2}\log[[(\sigma_i^{(k)})^2]] \\ &+ \frac{[[(\sigma_i^{(k)})^2]] - [[(\sigma_i^{(V)})^2]] + ([[\mu_i^{(k)}]] - [[\mu_i^V]])^2}{2[[(\sigma_i^V)^2]]},\end{aligned} \tag{35}$$

where the first two terms on the right-hand side require logarithm operation on the ciphertext. However, this may not be very practical because most HE schemes are designed for basic arithmetic operations on the ciphertext. Thus we also consider the situation where HE scheme at hand only provides *additive and multiplicative* homomorphisms [64]. In this case, to avoid the logarithm operation, the client k needs to keep every σ_i^2 in $RP_k(\theta, D_k)$ as plaintext and only encrypts μ_i, likewise for the server. As a result, the KL divergence can be computed under encryption as:

$$\begin{aligned}[[\mathrm{KL}(\mathcal{N}_i^{(k)} || \mathcal{N}_i^V)]] =& \Big[\Big[\frac{1}{2}\log(\frac{\sigma_i^V}{\sigma_i^{(k)}})^2 + \frac{1}{2}(\frac{\sigma_i^{(k)}}{\sigma_i^V})^2 - \frac{1}{2}\Big]\Big] \\ &+ \frac{1}{2(\sigma_i^V)^2}([[\mu_i^{(k)}]] - [[\mu_i^V]])^2\end{aligned} \tag{36}$$

where the first term on the right-hand side is encrypted after calculation with plaintext values $(\sigma_i^k)^2$ and $(\sigma^V)^2$ whereas the second term requires multiple operations on the ciphertext values $[[\mu_i^k]]$ and $[[\mu^V]]$.

Now, in either case, we can compute profile dissimilarity under encryption by summing up all the KL divergence values in ciphertext:

$$[[div(RP_k, RP_V)]] = \frac{1}{q}\sum_{i=1}^{q}[[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V)]] \tag{37}$$

APPENDIX D
CONVERGENCE ANALYSIS

In this section we provide the proof of the proposed Theorem 5.5. The analysis is mainly based on the results provided in [50]. We first introduce several notations to facilitate the analysis.

D.1 Notations

Let U ($|U| = N$) denote the full set of clients and $S(t)$ ($|S(t)| = K$) denote the set of clients selected for participating. By $\theta_k(t)$ we denote the local model on client k at time step t. We define an auxiliary sequence $v_k(t)$ for each client to represent the immediate local model after a local SGD update; $v_k(t)$ is updated from $\theta_k(t-1)$ with learning rate η_{t-1}:

$$v_k(t) = \theta_k(t-1) - \eta_{t-1}\nabla F_k(\theta_k(t-1), \xi_{k,t-1}), \tag{38}$$

where $\nabla F_k(\theta_k(t-1), \xi_{k,t-1})$ is the stochastic gradient computed over a batch of data $\xi_{k,t-1}$ drawn from D_k with regard to $\theta_k(t-1)$.

We also define two virtual sequences $\bar{v}(t) = \sum_{k=1}^{N}\rho_k v_k(t)$ and $\bar{\theta}(t) = Aggregate(\{v_k(t)\}_{k\in S(t)})$ for every time step t (Note that the actual global model $\theta(t)$ is only updated at the aggregation steps $T_A = \{\tau, 2\tau, 3\tau, \ldots\}$). Given an aggregation interval $\tau \geq 1$, we provide the analysis for the partial aggregation rule that yields $\bar{\theta}(t)$ as:

$$\bar{\theta}(t) = \frac{1}{K}\sum_{k\in S(t)} v_k(t), \tag{39}$$

where $S(t)$ ($|S(t)| = K$) is the selected set of clients for the round $\lceil \frac{t}{\tau} \rceil$ that contains step t. At the aggregation steps T_A, $\theta(t)$ is equal to $\bar{\theta}(t)$, i.e., $\theta(t) = \bar{\theta}(t)$ if $t \in T_A$.

To facilitate the analysis, we assume each client always performs model update (and synchronization) to produce $v_k(t)$ and $\bar{v}(t)$ (but obviously it does not affect the resulting $\bar{\theta}$ and θ for $k \notin S(t)$).

$$\theta_k(t) = \begin{cases} v_k(t), \text{ if } t \notin T_A \\ \bar{\theta}(t), \text{ if } t \in T_A \end{cases} \tag{40}$$

For ease of presentation, we also define two virtual gradient sequences: $\bar{g}(t) = \sum_{k=1}^{N}\rho_k\nabla F_k(\theta_k(t))$ and $g(t) = \sum_{k=1}^{N}\rho_k\nabla F_k(\theta_k(t), \xi_{k,t})$. Thus we have $\mathrm{E}[g(t)] = \bar{g}(t)$ and $\bar{v}(t) = \bar{\theta}(t-1) - \eta_{t-1}g(t-1)$.

D.2 Key Lemmas

To facilitate the proof of our main theorem, we first present several key lemmas.

Lemma D.1 (Result of one SGD step)**.** *Under Assumptions 5.1 and 5.2 and with $\eta_t < \frac{1}{4L}$, for any t it holds true that*

$$\mathrm{E}\|\bar{v}(t+1) - \theta^*\|^2 \leq (1-\eta_t\mu)\mathrm{E}\|\bar{\theta}(t) - \theta^*\|^2 + \eta_t^2\mathrm{E}\|g_t - \bar{g}_t\|^2 + 6L\eta_t^2\Gamma + 2\mathrm{E}\Big[\sum_{k=1}^{N}\rho_k\|\theta_k(t) - \bar{\theta}(t)\|^2\Big], \tag{41}$$

where $\Gamma = F^* - \sum_{k=1}^{N}\rho_k F_k^*$.

Lemma D.2 (Gradient variance bound)**.** *Under Assumption 5.3, one can derive that*

$$\mathrm{E}\|g_t - \bar{g}_t\|^2 \leq \sum_{k=1}^{N}\rho_k^2\epsilon_k^2. \tag{42}$$

Lemma D.3 (Bounded divergence of $\theta_k(t)$)**.** *Assume Assumption 5.4 holds and a non-increasing step size η_t s.t. $\eta_t \leq 2\eta_{t+\tau}$ for all $t = 1, 2, \ldots$, it follows that*

$$\mathrm{E}\Big[\sum_{k=1}^{N}\rho_k\|\theta_k(t) - \bar{\theta}(t)\|^2\Big] \leq 4\eta_t^2(\tau-1)^2G^2. \tag{43}$$

Lemmas D.1, D.2 and D.3 hold for both full and partial participation and are independent of the client sampling strategy. We refer the readers to [50] for their proofs and focus our analysis on opportunistic sampling.

Considering the condition that $\alpha_k \mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V)] = -ln(\Lambda\rho_k)$ for $k = 1, 2, \dots N$, the next two lemmas give important properties of the aggregated model $\bar{\theta}$ as a result of partial participation and non-uniform client sampling.

Lemma D.4 (Unbiased aggregation)**.** *For any aggregation step $t \in T_A$ and with $\alpha_k \mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V)] = -ln(\Lambda\rho_k)$ in the selection of $S(t)$, it follows that*

$$\mathrm{E}_{S(t)}[\bar{\theta}(t)] = \bar{v}(t). \tag{44}$$

Proof. Recall that we sample clients based on their divergence-based scores (Eq. 13). By taking the expectation over $S(t)$, we have

$$\begin{aligned}
\mathrm{E}_{S(t)} \sum_{k\in S(t)} v_k(t) &= K\mathrm{E}_{S(t)}[v_k(t)] \\
&= K \sum_{k=1}^{N} \frac{\lambda_k}{\Lambda} v_k(t) \\
&= K \sum_{k=1}^{N} \frac{\exp\big(-\alpha_k \cdot \mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V)]\big)}{\Lambda} v_k(t) \\
&= K \sum_{k=1}^{N} \frac{\exp\big(ln(\Lambda\rho_k)\big)}{\Lambda} v_k(t) \\
&= K \sum_{k=1}^{N} \rho_k v_k(t).
\end{aligned} \tag{45}$$

Take the expectation of $\bar{\theta}(t)$ over $S(t)$ and notice that $\bar{v}(t) = \sum_{k\in U} \rho_k v_k(t)$:

$$\begin{aligned}
\mathrm{E}_{S(t)}[\bar{\theta}(t)] &= \mathrm{E}_{S(t)}\big[\frac{1}{K} \sum_{k\in S(t)} v_k(t)\big] \\
&= \frac{1}{K}\mathrm{E}_{S(t)}\big[\sum_{k\in S(t)} v_k(t)\big] \\
&= \frac{1}{K} K \mathrm{E}_{S(t)}[v_k(t)] \\
&= \sum_{k\in U} \rho_k v_k(t) \\
&= \bar{v}(t).
\end{aligned}$$

□

Lemma D.5 (Bounded variance of $\bar{\theta}(t)$)**.** *For any aggregation step $t \in T_A$ and with a non-increasing step size η_t s.t. $\eta_t \leq 2\eta_{t+\tau-1}$, it follows that*

$$\mathrm{E}_{S(t)}\|\bar{\theta}(t) - \bar{v}(t)\|^2 \leq \frac{4}{K}\eta_{t-1}^2 \tau^2 G^2. \tag{46}$$

Proof. First, one can prove that our strategy yields $v_k(t)$ as an unbiased estimate of $\bar{v}(t)$ for any k:

$$\begin{aligned}
\mathrm{E}_{S(t)}[v_k(t)] &= \sum_{k=1}^{N} \frac{\exp\big(-\alpha_k \cdot \mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V)]\big)}{\Lambda} v_k(t) \\
&= \sum_{k=1}^{N} \frac{\exp\big(ln(\Lambda\rho_k)\big)}{\Lambda} v_k(t) \\
&= \sum_{k=1}^{N} \rho_k v_k(t) \\
&= \bar{v}(t).
\end{aligned} \tag{47}$$

Then by the aggregation rule $\bar{\theta}(t) = \frac{1}{K}\sum_{k\in S(t)} v_k(t)$, we have:

$$\begin{aligned}
\mathrm{E}_{S(t)}\|\bar{\theta}(t) - \bar{v}(t)\|^2 &= \frac{1}{K^2}\mathrm{E}_{S(t)}\|K\bar{\theta}(t) - K\bar{v}(t)\|^2 \\
&= \frac{1}{K^2}\mathrm{E}_{S(t)}\|\sum_{k\in S(t)} v_k(t) - \sum_{k=1}^{K}\bar{v}(t)\|^2 \\
&= \frac{1}{K^2}\mathrm{E}_{S(t)}\|\sum_{k\in S(t)} \big(v_k(t) - \bar{v}(t)\big)\|^2 \\
&= \frac{1}{K^2}\Big(\mathrm{E}_{S(t)}\sum_{k\in S(t)}\|v_k(t) - \bar{v}(t)\|^2 \\
&\quad \underbrace{+\,\mathrm{E}_{S(t)}\sum_{i,j\in S(t), i\neq j}\langle v_i(t) - \bar{v}(t), v_j(t) - \bar{v}(t)\rangle}_{=0}\Big),
\end{aligned} \tag{48}$$

where the second term on the RHS of (48) equals zero because $\{v_k(t)\}_{k\in U}$ are independent and unbiased (see Eq. 47). Further, by noticing $t-\tau \in T_A$ (because $t \in T_A$) which implies that $\theta_k(t-\tau) = \bar{\theta}(t-\tau)$ since the last communication, we have:

$$\begin{aligned}
\mathrm{E}_{S(t)}\|\bar{\theta}(t) - \bar{v}(t)\|^2 &= \frac{1}{K^2}\mathrm{E}_{S(t)}\sum_{k\in S(t)}\|v_k(t) - \bar{v}(t)\|^2 \\
&= \frac{1}{K^2}K\mathrm{E}_{S(t)}\|v_k(t) - \bar{v}(t)\|^2 \\
&= \frac{1}{K}\mathrm{E}_{S(t)}\|\big(v_k(t) - \bar{\theta}(t-\tau)\big) - \big(\bar{v}(t) - \bar{\theta}(t-\tau)\big)\|^2 \\
&\leq \frac{1}{K}\mathrm{E}_{S(t)}\|v_k(t) - \bar{\theta}(t-\tau)\|^2,
\end{aligned} \tag{49}$$

where the last inequality results from $\mathrm{E}[v_k(t) - \bar{\theta}(t-\tau)] = \bar{v}(t) - \bar{\theta}(t-\tau)$ and that $\mathrm{E}\|X - \mathrm{E}X\|^2 \leq \tau\|X\|^2$. Further, we have:

$$\begin{aligned}
\mathrm{E}_{S(t)}\|\bar{\theta}(t) - \bar{v}(t)\|^2 &\leq \frac{1}{K}\mathrm{E}_{S(t)}\|v_k(t) - \bar{\theta}(t-\tau)\|^2 \\
&= \frac{1}{K}\sum_{k=1}^{N}\frac{1}{\Lambda}\exp\big(-\alpha_k\cdot\mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^{V})]\big)\mathrm{E}_{S(t)}\|v_k(t) - \bar{\theta}(t-\tau)\|^2 \\
&= \frac{1}{K}\sum_{k=1}^{N}\frac{1}{\Lambda}\exp\big(-\alpha_k\cdot\mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^{V})]\big)\underbrace{\mathrm{E}_{S(t)}\|\sum_{i=t-\tau}^{t-1}\eta_i\nabla F_k(\theta_k(i),\xi_{k,i})\|^2}_{Z_1}.
\end{aligned} \tag{50}$$

Let $i_m = \arg\max_i \|\nabla F_k\big(\theta_k(i),\xi_{k,i}\big)\|, i \in [t-\tau, t-1]$. By using the Cauchy-Schwarz inequality, Assumption 5.4 and choosing a non-increasing η_t s.t. $\eta_t \leq 2\eta_{t+\tau-1}$, we have:

$$\begin{aligned}
Z_1 &= \mathrm{E}_{S(t)}\|\sum_{i=t-\tau}^{t-1}\eta_i\nabla F_k(\theta_k(i),\xi_{k,i})\|^2 \\
&= \sum_{i=t-\tau}^{t-1}\sum_{j=t-\tau}^{t-1}\mathrm{E}_{S(t)}\langle\eta_i\nabla F_k\big(\theta_k(i),\xi_{k,i}\big),\eta_j\nabla F_k\big(\theta_k(j),\xi_{k,j}\big)\rangle \\
&\leq \sum_{i=t-\tau}^{t-1}\sum_{j=t-\tau}^{t-1}\mathrm{E}_{S(t)}\Big[\|\eta_i\nabla F_k\big(\theta_k(i),\xi_{k,i}\big)\|\cdot\|\eta_j\nabla F_k\big(\theta_k(j),\xi_{k,j}\big)\|\Big] \\
&\leq \sum_{i=t-\tau}^{t-1}\sum_{j=t-\tau}^{t-1}\eta_i\eta_j\cdot\mathrm{E}_{S(t)}\|\nabla F_k\big(\theta_k(i_m),\xi_{k,i_m}\big)\|^2 \\
&\leq \sum_{i=t-\tau}^{t-1}\sum_{j=t-\tau}^{t-1}\eta_{t-\tau}^2\cdot\mathrm{E}_{S(t)}\|\nabla F_k\big(\theta_k(i_m),\xi_{k,i_m}\big)\|^2 \\
&\leq 4\eta_{t-1}^2\tau^2G^2.
\end{aligned} \tag{51}$$

Plug Z_1 back into (50) and notice that $\sum_{k=1}^{N}\frac{1}{\Lambda}\exp\big(-\alpha_k\cdot \mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V)]\big)=1$, we have:

$$\begin{aligned}\mathrm{E}_{S(t)}\|\bar{\theta}(t)-\bar{v}(t)\|^2 &\le \frac{1}{K}\sum_{k=1}^{N}\frac{1}{\Lambda}\exp\big(-\alpha_k\cdot \mathrm{E}[\mathrm{KL}(\mathcal{N}_i^{(k)}||\mathcal{N}_i^V)]\big)4\eta_{t-1}^2\tau^2G^2\\ &=\frac{4}{K}\eta_{t-1}^2\tau^2G^2.\end{aligned}$$

□

D.3 Proof of Theorem 5.5

Proof. Taking expectation of $\|\bar{\theta}(t)-\theta^*\|^2$, we have:

$$\begin{aligned}\mathrm{E}\|\bar{\theta}(t)-\theta^*\|^2 &= \mathrm{E}_{S(t)}\|\bar{\theta}(t)-\bar{v}(t)+\bar{v}(t)-\theta^*\|^2\\ &=\underbrace{\mathrm{E}\|\bar{\theta}(t)-\bar{v}(t)\|^2}_{A_1}+\underbrace{\mathrm{E}\|\bar{v}(t)-\theta^*\|^2}_{A_2}+\underbrace{2\mathrm{E}\langle\bar{\theta}(t)-\bar{v}(t),\bar{v}(t)-\theta^*\rangle}_{A_3}\end{aligned}\tag{52}$$

where A_3 vanishes because $\bar{\theta}(t)$ is an unbiased estimate of $\bar{v}(t)$ by first taking expectation over $S(t)$ (Lemma D.4).

To bound A_2 for $t\in T_A$, we apply Lemma D.1:

$$\begin{aligned}A_2=\mathrm{E}\|\bar{v}(t)-\theta^*\|^2 &\le (1-\eta_{t-1}\mu)\mathrm{E}\|\bar{\theta}(t-1)-\theta^*\|^2+\underbrace{\eta_{t-1}^2\mathrm{E}\|g_{t-1}-\bar{g}_{t-1}\|^2}_{B_1}\\ &\quad+6L\eta_{t-1}^2\Gamma+\underbrace{\mathrm{E}\Big[\sum_{k=1}^{N}\rho_k\|\theta_k(t-1)-\bar{\theta}(t-1)\|^2\Big]}_{B_2}.\end{aligned}\tag{53}$$

Then we use Lemmas D.2 and D.3 to bound B_1 and B_2 respectively, which yields:

$$A_2=\mathrm{E}\|\bar{v}(t)-\theta^*\|^2\le(1-\eta_{t-1}\mu)\mathrm{E}\|\bar{\theta}(t-1)-\theta^*\|^2+\eta_{t-1}^2\mathcal{B},\tag{54}$$

where $\mathcal{B}=\sum_{k=1}^{N}\rho_k^2\epsilon_k^2+6L\Gamma+8(\tau-1)^2G^2$.

To bound A_1, one can first take expectation over $S(t)$ and apply Lemma D.5 where the upper bound actually eliminates both sources of randomness. Thus, it follows that

$$A_1=\mathrm{E}\|\bar{\theta}(t)-\bar{v}(t)\|^2\le\frac{4}{K}\eta_{t-1}^2\tau^2G^2\tag{55}$$

Let $\mathcal{C}=\frac{4}{K}\tau^2G^2$ and plug A_1 and A_2 back into (52):

$$\mathrm{E}\|\bar{\theta}(t)-\theta^*\|^2\le(1-\eta_{t-1}\mu)\mathrm{E}\|\bar{\theta}(t-1)-\theta^*\|^2+\eta_{t-1}^2(\mathcal{B}+\mathcal{C}).\tag{56}$$

Equivalently, let $\Delta_t=\mathrm{E}\|\bar{\theta}(t)-\theta^*\|^2$, then we have the following recurrence relation for any $t\ge1$:

$$\Delta_t\le(1-\eta_{t-1}\mu)\Delta_{t-1}+\eta_{t-1}^2(\mathcal{B}+\mathcal{C}).\tag{57}$$

Next we prove by induction that $\Delta_t\le\frac{\nu}{\gamma+t}$ where $\nu=\max\Big\{\frac{\beta^2(\mathcal{B}+\mathcal{C})}{\beta\mu-1},(\gamma+1)\Delta_1\Big\}$ using an aggregation interval $\tau\ge1$ and a diminishing step size $\eta_t=\frac{\beta}{t+\gamma}$ for some $\beta>\frac{1}{\mu}$ and $\gamma>0$ such that $\eta_1\le\min\{\frac{1}{\mu},\frac{1}{4L}\}$ and $\eta_t\le2\eta_{t+\tau}$.

First, for $t=1$ the conclusion holds that $\Delta_1\le\frac{\nu}{\gamma+1}$ given the conditions. Then by assuming it holds for some t, one can derive from (57) that

$$\begin{aligned}\Delta_{t+1}&\le(1-\eta_t\mu)\Delta_t+\eta_t^2(\mathcal{B}+\mathcal{C})\\ &\le\Big(1-\frac{\beta\mu}{t+\gamma}\Big)\frac{\nu}{\gamma+t}+\Big(\frac{\beta}{t+\gamma}\Big)^2(\mathcal{B}+\mathcal{C})\\ &=\frac{t+\gamma-1}{(t+\gamma)^2}\nu+\underbrace{\Big[\frac{\beta^2(\mathcal{B}+\mathcal{C})}{(t+\gamma)^2}-\frac{\beta\mu-1}{(t+\gamma)^2}\nu\Big]}_{\ge0}\\ &\le\frac{t+\gamma-1}{(t+\gamma)^2}\nu\\ &\le\frac{t+\gamma-1}{(t+\gamma)^2-1}\nu\\ &=\frac{\nu}{t+\gamma+1},\end{aligned}\tag{58}$$

which proves the conclusion $\Delta_t\le\frac{\nu}{\gamma+t}$ for any $t\ge1$.

Then by the smoothness of the objective function F, it follows that

$$\begin{aligned}\mathrm{E}[F(\bar{\theta}(t))] - F^* &\leq \frac{L}{2}\mathrm{E}\|\bar{\theta}(t) - \theta^*\|^2 \\ &= \frac{L}{2}\Delta_t \leq \frac{L}{2}\frac{\nu}{\gamma + t}.\end{aligned} \tag{59}$$

Specifically, by choosing $\beta = \frac{2}{\mu}$ (i.e., $\eta_t = \frac{2}{\mu(\gamma+t)}$), $\gamma = \max\{\frac{8L}{\mu}, \tau\} - 1$, we have

$$\begin{aligned}\nu &= \max\Big\{\frac{\beta^2(\mathcal{B}+\mathcal{C})}{\beta\mu - 1}, (\gamma+1)\Delta_1\Big\} \\ &\leq \frac{\beta^2(\mathcal{B}+\mathcal{C})}{\beta\mu - 1} + (\gamma+1)\Delta_1 \\ &= \frac{4(\mathcal{B}+\mathcal{C})}{\mu^2} + (\gamma+1)\Delta_1.\end{aligned} \tag{60}$$

By definition, we have $\theta(t) = \bar{\theta}(t)$ at the aggregation steps. Therefore, for $t \in T_A$:

$$\begin{aligned}\mathrm{E}[F(\theta(t))] - F^* &\leq \frac{L}{2}\frac{\nu}{\gamma + t} \\ &= \frac{L}{(\gamma+t)}\Big(\frac{2(\mathcal{B}+\mathcal{C})}{\mu^2} + \frac{\gamma+1}{2}\Delta_1\Big).\end{aligned}$$

□

Remark. Random sampling cannot guarantee the convergence of the global model in our setting. This boils down to the mismatch between sampling weights (i.e., an uniform probability distribution for random sampling) and the importance of local data distributions to the global objective. The latter relates to the quality of data and is theoretically non-uniform [49]. Using random sampling to generate $S(t)$ in this case leads to a biased aggregation [50]. The expectation of the aggregated model $\bar{\theta}(t)$ is:

$$\begin{aligned}\mathrm{E}_{S(t)}\bar{\theta}(t) &= \mathrm{E}_{S(t)}\frac{1}{K}\sum_{k\in S(t)} v_k(t) \\ &= K\frac{1}{K}\mathrm{E}_{S(t)}[v_k(t)] \\ &= \sum_{k=1}^{N}\frac{1}{N}v_k(t).\end{aligned} \tag{61}$$

Recall that the global deterministic aggregation $\bar{v}(t) = \sum_{k=1}^{N}\rho_k v_k(t)$ and typically $\rho_k \neq 1/N$. This means the aggregated model deviates in expectation from the deterministic aggregation result $\bar{v}(t)$ for each aggregation step, i.e., $\mathrm{E}_{S(t)}\bar{\theta}(t) \neq \bar{v}(t), \forall t = \tau, 2\tau, \ldots$. In other words, the aggregated model is biased. Hence, Lemmas D.4 and D.5 do not hold true in this random sampling case and thus in Eq. (52), terms A_1 and A_3 are unbounded because of the bias in the aggregated model $\bar{\theta}(t)$ introduced by random sampling. As a result, the convergence of the global model cannot be guaranteed.