FedNaWi: Selecting the Befitting Clients for Robust Federated Learning in IoT Applications

Zhu, Ran ; Yang, Mingkun; Yang, Jie; Wang, Qing

DOI
10.1109/SECON58729.2023.10287422

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)

2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON) | 979-8-3503-0052-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/SECON58729.2023.10287422

FedNaWi: Selecting the Befitting Clients for Robust Federated Learning in IoT Applications

Ran Zhu[*], Mingkun Yang[*], Jie Yang, and Qing Wang
Delft University of Technology, The Netherlands
{r.zhu-1, m.yang-3, j.yang-3, qing.wang}@tudelft.nl

Abstract— **Federated Learning (FL) is an important privacy-preserving learning paradigm that is expected to play an essential role in the future Intelligent Internet of Things (IoT). However, model training in FL is vulnerable to noise and the statistical heterogeneity of local data across IoT clients. In this paper, we propose FedNaWi, a *"Go Narrow, Then Wide"* client selection method that speeds up the FL training, achieves higher model performance, while requiring no additional data or sensitive information transfer from clients. Our method first selects reliable clients (i.e., *going narrow*) which allows the global model to quickly improve its performance and then includes less reliable clients (i.e., *going wide*) to exploit more IoT data of clients to further improve the global model. To profile client utility, we introduce a unified Bayesian framework to model the client utility at the FL server, assisted by a small amount of auxiliary data. We conduct extensive evaluations with 5 state-of-the-art FL methods, on 3 IoT tasks and under 7 different types of label and feature noise. We build an FL testbed with 38 IoT nodes (20 nodes run on Raspberry Pi 4B and 18 nodes run on Jetson Nano) for the evaluation. Our results show that FedNaWi improves the FL accuracy substantially and significantly reduces energy consumption. In particular, FedNaWi improves the accuracy from 35% to 75% in the non-IID Dirichlet setting, and reduces the average energy consumption by 55%.**

I. Introduction

Thanks to the Internet of Things (IoT) and the evolutions of onboard sensory and computing capabilities, our world now has tens of billions of IoT devices deployed for various tasks. They usually upload the data to servers where advanced techniques such as deep learning can be leveraged to perform tasks. However, uploading raw/processed data to cloud/edge servers raises privacy issues [1]. *Federated learning (FL)*, a distributed training paradigm, is thus proposed to preserve privacy. A cloud/edge server coordinates the local training of devices (i.e., *clients*) where clients only report weights of their local models to the server in a federated fashion, and FL learns global model gradually by *aggregating clients' weights* in each round. The required number of rounds varies with exact tasks, data quantity, quality and distribution, targeted accuracy, etc.

In conventional FL methods such as FedAvg [1], the server *randomly* selects a pre-defined number of clients in each round and aggregates their updated local weights. The selection holds the assumption that clients contribute equally to the training of the global model, which is further described as the implicit assumption that all clients are reliable in that data for local training is of high quality. In real applications, however, such

[*]Both authors contributed equally to this work.

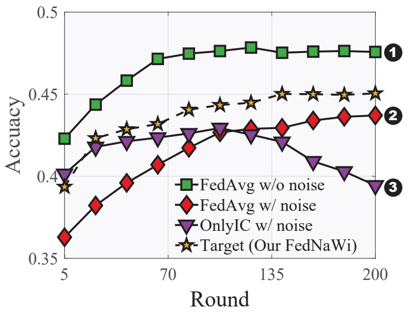


Fig. 1: Illustration of the negative impact of noisy IoT clients.

an assumption usually does not hold: studies have reported that *existing datasets could easily contain more than 30% label errors* [2, 3]; the issue is further complicated by the non-IID (identically and independently distributed) data distributions across clients. These issues have a significant effect on the model performance. To illustrate such an effect, without loss of generality, we show the dynamics of FL performance throughout the training process on the CIFAR10, in a noisy and non-IID setting: each client holds 2 classes of data with an overall 16.5% labels corrupted (cf. Section IV-A for more details). As shown in Figure 1, we can observe the following:

- **Observation 1: unreliable clients slow the learning of the global model and reduce its accuracy.** When there are erroneous samples in clients' local data, the maximal achieved accuracy is reduced from about 48% (curve ❶) to 44% (curve ❷). Such drops are usually considered significant in image recognition. Besides, when there are unreliable clients, the number of rounds required to obtain a certain accuracy is significantly prolonged. Taking a targeted accuracy of 44% for example, the required number of rounds goes from less than 50 to at least 200, leading to a factor of 4×.
- **Observation 2: selecting only clients with inerrant data accelerates the learning speed of the global model but does not improve model performance.** Considering scenarios where label noise is present, we observe at the early stage of training, FL shows better performance when the server only aggregates weights from inerrant clients (curve ❸ "OnlyIC w/ noise") compared with that when updates from all clients are aggregated (curve ❷). Selecting inerrant clients, therefore, allows to reach relatively high accuracy within fewer rounds. The global model performance however does not benefit from excluding unreliable clients in the long run: the learning

curve starts to drop after 100 rounds, which is likely due to model over-fitting on the limited data volume.

- **Observation 3: unreliable clients still contribute to learning a more accurate global model.** Interestingly, we observe although noisy clients slow the convergence of the learning curve, they actually contribute to learning a better global model in the long run. This can be seen in the figure (comparing ❷ with ❸): after 100 rounds, the accuracy of FedAvg trained with all clients is higher than that with only inerrant clients. We speculate the global model benefits from the larger volume of training data.

The above observations further hint on how to properly select clients for training the global model at the FL server. Ideally, the learning curve in the context of noisy-data scenarios should first increase sharply to a certain accuracy by an elaborate selection of reliable clients for aggregation (i.e., *going narrow*), and then take a gentle slope to reach a higher accuracy by a broad selection of clients to include less reliable ones for aggregation (i.e., *going wide*). In FedNaWi, we target such a learning curve of the global model for robust FL.

To achieve this goal, a key challenge is how to adaptively select suitable clients when training proceeds. An important criterion of the selection should be the utility of clients which can reflect the quality of their local data. It is, however, nontrivial to infer such utility due to the lack of accessibility to local data. Therefore, A key problem is how to infer client utility without compromising privacy. To tackle this, we propose to introduce high-quality, auxiliary data for utility estimation. Such data shouldn't be large due to extra cost incurred for data collection. This poses another challenge in how to effectively infer client utility with limited auxiliary data. To address this issue, we explore two complementary strategies to exploit the auxiliary data for client utility inference: 1) creating synthetic clients based on auxiliary data to build a discriminator that distinguishes reliable clients from unreliable ones based on their weight parameters, and 2) evaluating the performance of local models as an indicator of client utility. We propose a unified Bayesian framework that seamlessly integrates these two strategies and further introduces a variational inference algorithm that allows the inference based on these strategies to benefit from each other, thereby reaching an effect where the whole is greater than the sum of its part.

By carefully selecting clients for FL, FedNaWi benefits from both the inerrant clients in speeding up model training and the noisy clients in improving overall model performance. It can handle data noise of various types (e.g., random and structural noise) in both the feature and label space while remaining client transparent: no additional data or sensitive information transfer from clients is needed. More specifically, we make following key contributions (summarised in Table I):

- We introduce the robustness problem of FL under noisy local data. We propose adaptive client selection as an approach to address it. We specifically propose a two-phase client selection method that first selects reliable clients and then includes less reliable ones for aggregation.

TABLE I: Comparison between FedNaWi and SOTA methods.

Method	New-Client Joining	Client Transparency	Testbed Validation	Multi-type Noise
FedCorr [4]	✗	✗	✗	Only Random noise
FLDebugger [5]	✓	✗	✓	Only Random noise
FLAME [6]	✓	✓	✗	Backdoor attack
Oort [7]	✓	✗	✗	-
PyramidFL [8]	✓	✗	✓	-
Our FedNaWi	✓	✓	✓	Random and Structural noise in both label and feature space (7 types)

- We propose a Bayesian framework to model client reliability and an efficient variational inference algorithm to infer client reliability from observations constructed in two complementary ways: the weight parameters of local clients and their performance.
- We evaluate the client selection and utility inference performance of FedNaWi with thorough testing and compare it with 5 state-of-the-art FL methods on 3 datasets. Especially, we build an FL testbed with 20 Raspberry Pi 4B and 18 NVIDIA Jetson NANO for the evaluations on IoT-related HARBox datasets. Extensive evaluations on 7 different types of label/feature noise under practical non-IID settings show that FedNaWi consistently and substantially improves the robustness of all FL methods and reduces the communication cost. FedNaWi is especially effective in non-IID settings: it can reach $1.5\times$ accuracy improvement in the Dirichlet setting (with data imbalance level $\alpha = 0.5$).

II. Background and Related Work

A. Federated Learning

Under the typical FL protocol, a complete communication process (a.k.a., round) between the involved clients and server consists of the following four steps [9]: (a) At the beginning of i-th round, clients with indices $\mathcal{J}^i$ (a subset of all candidate clients $\mathcal{C}$) are activated, and the server sends the current global model to all participating clients; (b) Each client conducts local training in the same initial state, but on self-contained data; (c) After local training, each client offloads the update for its local model to the server; (d) By the aggregation of updates from clients on the server side, FL iterates until it reaches a certain goal, such as an expected inference accuracy or the maximum round. To be more specific, the j-th ($j \in \mathcal{J}^i$) client runs an optimizer like Stochastic Gradient Descent (SGD) to search for optimal model weight parameters $\mathcal{W}^*_{i,j}$ by minimizing local objective function $\mathcal{L}$ on local data $\mathcal{D}_j$. Each local model is initialized by weights of global model $\bar{\mathcal{W}}^*_{i-1}$ in the last round.

Server updates the global model in the way of aggregating all clients' updates $\{\mathcal{W}^*_{i,j}\}_{j\in\mathcal{J}^i}$, which can be treated as optimizing the global objective function:

$$\bar{\mathcal{W}}^*_i = \arg\min_{\mathcal{W}} \sum_{j\in\mathcal{J}^i} p_j \mathcal{L}_{\mathcal{D}_j}(\mathcal{W}) \ , \tag{1}$$

where $p_j \geq 0$ and $\sum p_j = 1$. p_j is the proportion of the local samples in the samples of all clients [1], which specifies the relative impact of each device by the size of local data [10].

B. Client Selection

Client selection has recently drawn an increasing amount of attention in FL in response to challenges arising from heterogeneous data distribution and systems. Seminal work [7, 8] proposes to guide the client selection in each round based on data distribution and computational efficiency of local clients, which yield superior time-to-accuracy performance than random selection. Work can also be found on reducing the negative impact on training efficacy caused by random client selection criteria. [11] selects high-utility local samples for enabling all clients with various computing resources to complete local training before the same deadline. [12] estimates time consumption by the resource information reported from each client and selects as many as possible clients completing the current round within a deadline. FedMarl [13] builds a decision environment to execute client selection by jointly optimizing model accuracy, processing latency, and communication efficiency via multi-agent reinforcement learning. The above work focuses on client selection to cope only with heterogeneity issues, assuming local data is completely high quality, which is impractical in real-world scenarios.

Relevant studies on client selection with noisy data are [5] and [4] as our experiments involve. [5] adopts a 2-norm distance between local and global weights for client selection, not necessarily approximating client utility. [4] utilizes average Local Intrinsic Dimension (LID) transmitted by clients to build Gaussian Mixture Model for recognizing reliable clients. This method has several drawbacks: it requires clients to send additional information while cannot handle connections from new clients during the FL process. In contrast, our proposed framework FedNaWi does not require any additional information from clients and can flexibly take in new clients to join in the selection and FL. Our work is further related to recent work that aims to build reputation profiles of clients based on their historical performance for future FL tasks, e.g., through blockchain systems [14, 15]. Our proposed framework is complementary to this line of work by providing inferred client utility as reputation input for blockchain systems.

In the context of existing clients with noise and extremely imbalanced local data, our goal is to train an optimal global model by selecting a proper subset of clients whose model weights are used for updating the global model. Let $s_j \in \{0, 1\}$ denote the selection of j-th client involved in i-th round ($j \in \mathcal{J}^i$), the objective for global model is reformulated as

$$\bar{\mathcal{W}}_i^* = \arg\min_{\mathcal{W}} \sum_{j \in \mathcal{J}^i} s_j p_j \mathcal{L}_{\mathcal{D}_j}(\mathcal{W}). \tag{2}$$

Note that in this formulation the constraint of selected clients p_j should satisfy $\sum s_j p_j = 1$. p_j can be set as the proportion of the local samples in the samples of all *selected* clients.

III. Approach and System

A. Overview

Figure 2 depicts the pipeline of FedNaWi. When the server receives weight parameters $\{\mathcal{W}_{i,j}^*\}_{j \in \mathcal{J}^i}$ from all activated clients in current round, FedNaWi first infers the utility of each update using a discriminator $f^{\mathcal{W}_d}(\cdot)$ with learnable parameter $\mathcal{W}_d$. To train this discriminator, an important and perhaps indispensable means is the inclusion of high-quality auxiliary data $\mathcal{D}_a$ with correct labels and a representative of the data distribution that the global model encounters in a real-world deployment. Such an approach does not compromise client privacy given that server-hosted auxiliary data and computation.A practical limitation of such an approach, however, is that the auxiliary data comes at a cost – as it requires human labor for labeling – and thus the data can only be of limited size. We leave the solution to section III-B to introduce in detail a cost-effective way to best leverage a limited amount of auxiliary data for client utility inference. To reduce the computational overhead, FedNaWi further employs a trigger (specified in section III-D) that allows to stop estimating utility and switch over to the client utility inferred in earlier rounds.

After that, FedNaWi adaptively selects the befitting local updates for aggregation following the principle: it first selects updates with high utilities to quickly improve the global model to certain performance and then includes also clients with relatively less utility to further improve performance.

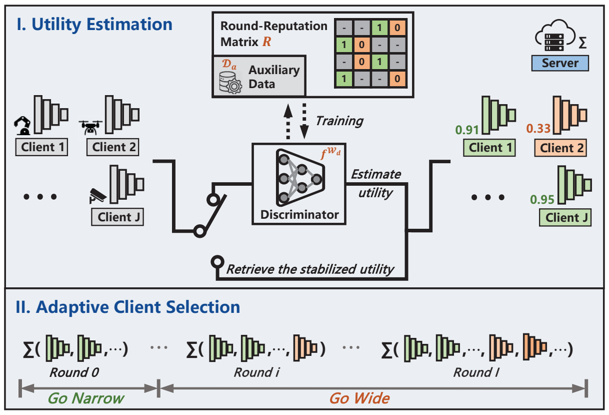


Fig. 2: FedNaWi pipeline: it first leverages a small, auxiliary data on the server to infer client utility, using a Bayesian framework that considers both weight parameters and performance of local models; it can skip utility inference when the inferred utility gets stabilized, thereby reducing computational overhead; based on the inferred client utility, it then selects suitable clients in an adaptive manner that goes through two phases, starting by selecting only high-utility clients to quickly improve global model performance and then selecting also relatively low-utility clients to further improve global model.

B. Client Utility Estimation

The discriminator treats the weight parameters reported by the j-th client as the input to infer the update utility θ_j. We specifically consider the weights of the topmost layer of local models, since those weights are most relevant for the given task [16] and hence most discriminative for utility inference. Thus, the discriminator can be formulated as

$$\theta_j = f^{\mathcal{W}_d}(\mathbf{x}_j). \tag{3}$$

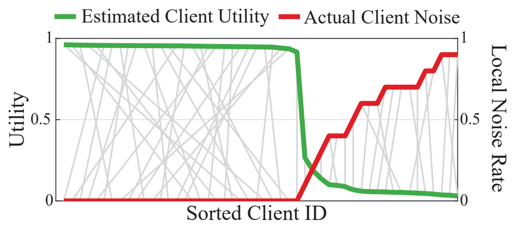


Fig. 3: Utility inference on CIFAR10: 30% clients with noise randomly sampling from (0,1]. Grey curves link each client's utility and its actual noise level.

where $\mathbf{x}_j$ denotes parameters of the topmost layer in reported update $\mathcal{W}^*_{i,j}$. Note that any machine learning model for the discriminator can generate $\theta_j \in [0,1]$ with a sigmoid function.

Training the discriminator is confronted with the lack of labels. Our strategy is to fully leverage the well-labeled auxiliary data $\mathcal{D}_a$ in two ways: 1) we first utilize the auxiliary data $\mathcal{D}_a$ to simulate synthetic reliable clients with all clean data $\mathcal{D}^+_a$, as well as unreliable clients with manually corrupted data $\mathcal{D}^-_a$; 2) we also take the auxiliary data $\mathcal{D}_a$ as the validation data to evaluate and record each client's performance, which can serve to better infer client utility. To this end, we propose to create a round-reputation matrix $\mathbf{R}$ that keeps track of local model performance in different rounds. In such a matrix, the entry $\mathbf{R}_{i,j}$ in the i-th row and j-th column records the performance of j-th client in round i. We devise the following way for determining values in the matrix: $\mathbf{R}_{i,j} = 1$ if the performance of the local model from client j exceeds an empirical threshold, and $\mathbf{R}_{i,j} = 0$ otherwise. However, we can not directly treat the entries in $\mathbf{R}$ as the labels since 1) $\mathbf{R}$ is a sparse matrix due to a relatively small proportion of candidate clients activated in each round. Although this makes $\mathbf{R}$ low storage and operation overheads, the informativeness of a single row of $\mathbf{R}$ is limited, and entries are susceptible to the training randomness; 2) threshold setting should be dynamic across rounds and require prior knowledge of the general utility of clients. For instance, if threshold is the medium performance of activated clients, the implicit assumption would be that there are considerable numbers of high-utility clients whose local data is clean and in a relatively balanced distribution.

A more sophisticated way is to consider client performance consistency across rounds and only consider the clients whose performance consistently exceeds a threshold in multiple rounds as high-utility ones, which gains informativeness from different rounds. To this end, we formalize the discriminator output θ_j and the entry $\mathbf{R}_{i,j}$ into a Bayesian framework, and then train the discriminator using variational inference.

It is reasonable to describe the selection decision s_j of client j by a Bernoulli distribution parameterized by client utility θ_j:

$$s_j \sim Ber(\theta_j). \tag{4}$$

We denote round informativeness as $r_i \in [0,1]$ where $r_i = 1$ means that the i-th round provides sufficient information for inferring client utility and $r_i = 0$ otherwise. To account for the uncertainty in estimating r_i to make our framework more robust, we take a Bayesian view and model the prior probability distribution as a Beta distribution:

$$r_i \sim Beta(\alpha_i, \beta_i), \tag{5}$$

where α_i and β_i are the parameters of the distribution.

We define the likelihood of observed client performance in different rounds, i.e., round-reputation matrix, as the probability conditioned on the informativeness of round r_i and the client selection decision s_j (determined by client utility θ_j):

$$p(\mathbf{R}_{i,j}|r_i, s_j) = r_i^{\mathbb{1}(s_j=\mathbf{R}_{i,j})} + (1-r_i)^{\mathbb{1}(s_j\neq\mathbf{R}_{i,j})}, \tag{6}$$

where $\mathbb{1}(\cdot)$ is the indicator function. The informativeness of the round is higher if there are more tags $\{\mathbf{R}_{i,j}\}_{j\in\mathcal{J}^i}$ satisfying the actual client utility. Parameters of the Bayesian framework are learned by maximizing the likelihood function:

$$p(\mathbf{R}) = \int p(\mathbf{R}, \mathbf{r}, \mathbf{s}|\mathbf{X}; \mathcal{W}_d) d\mathbf{r}, \mathbf{s}, \tag{7}$$

where $\mathbf{r}$ and $\mathbf{s}$ are the latent true informativeness of all rounds and the selection decision for all clients, respectively; $\mathbf{X} = \{\mathbf{x}_j\}_{j\in\mathcal{J}^i}$ is the set of the topmost layers.

Maximizing the above function is computationally unfeasible due to the two latent variables in the integral [17]. We solve this problem by Variational Expectation Maximization (variational EM) that iterates between two steps: 1) the E-step where we approximate the distribution of latent variables $p(\mathbf{r}, \mathbf{s}|\mathbf{R}, \mathbf{X}; \mathcal{W}_d)$ with the variational distribution $q(\mathbf{r}, \mathbf{s})$; 2) M-step where we update the estimate for the parameters of the discriminator $\mathcal{W}_d$ by maximizing the evidence lower bound (ELBO) [18] of Eq. (7) given the updated latent variables.

We take a mean-field approach [18] to update individual latent variables in each iteration of the variational EM algorithm. Updating rules for $q(\mathbf{r}, \mathbf{s})$ in E-step follow given theorems.[1]

Theorem 3.1 (Incremental Client Utility): $q(s_j)$ can be updated based on the output of discriminator θ_j and the parameters α_i and β_i of all rounds with indices $i \in \mathcal{I}^j$ involving j-th client. We can derive the updated rule

$$q(s_j = 1) \propto \begin{cases} \theta_j \prod_{i\in\mathcal{I}^j} \exp\{\Psi(\beta_i) - \Psi(\alpha_i+\beta_i)\} & (\mathbf{R}_{i,j} = 0) \\ \theta_j \prod_{i\in\mathcal{I}^j} \exp\{\Psi(\alpha_i) - \Psi(\alpha_i+\beta_i)\} & (\mathbf{R}_{i,j} = 1) \end{cases} \tag{8}$$

$$q(s_j = 0) \propto \begin{cases} (1-\theta_j) \prod_{i\in\mathcal{I}^j} \exp\{\Psi(\alpha_i) - \Psi(\alpha_i+\beta_i)\} & (\mathbf{R}_{i,j} = 0) \\ (1-\theta_j) \prod_{i\in\mathcal{I}^j} \exp\{\Psi(\beta_i) - \Psi(\alpha_i+\beta_i)\} & (\mathbf{R}_{i,j} = 1) \end{cases} \tag{9}$$

where $\Psi(\cdot)$ is the Digamma function.

Theorem 3.2 (Incremental Round Informativeness): The informativeness distribution of i-th round $q(r_i)$ can be updated based on α_i and β_i from the last E-M iteration and true distribution of client utility in current iteration θ':

$$q(r_i) \propto \begin{cases} Beta(\alpha_i + \sum_{j\in\mathcal{J}^i}(1-\theta'), \beta_i + \sum_{j\in\mathcal{J}^i}\theta') & (\mathbf{R}_{i,j} = 0) \\ Beta(\alpha_i + \sum_{j\in\mathcal{J}^i}\theta', \beta_i + \sum_{j\in\mathcal{J}^i}(1-\theta')) & (\mathbf{R}_{i,j} = 1) \end{cases} \tag{10}$$

[1]We omit the detailed proofs due to the space limit. Proofs will be included when space allows in the future.

Algorithm 1: FedNaWi Robust Federated Learning

```
input  : A set of clients with self-contained data: $\mathcal{C}$;
         Two sets of possible utility thresholds: $\Lambda_s$ and $\Lambda_l$;
         Server auxiliary dataset: $\mathcal{D}_a$; Client selection rate: $\gamma$.
output: $\bar{\mathcal{W}}_i^*$
1  for each round $i = 1, 2, \cdots, N$ do
2      $\mathcal{J}^i \leftarrow$ randomly select $max(|\mathcal{C}| \times \gamma, 1)$ clients from $\mathcal{C}$
3      for $j \in \mathcal{J}^i$ in parallel do
           /* client-transparent training */
4          $\mathcal{W}_{i,j}^* \leftarrow$ClientUpdate $(C_j, \bar{\mathcal{W}}_{i-1}^*)$
       Server Executes FedNaWi:
5      initialize $\bar{\mathcal{W}}_0^*, \mathbf{R}_{0\times|\mathcal{C}|}$
       /* client utility estimation */
6      $\mathbf{R}_{i\times|\mathcal{C}|} \leftarrow$MatrixUpdate $(\{\mathcal{W}_{i,j}^*\}_{j\in\mathcal{J}^i}, \mathbf{R}_{i-1\times|\mathcal{C}|}, \mathcal{D}_a)$
7      $\mathcal{W}_d \leftarrow$VariationalInference
         $(\{\mathcal{W}_{i,j}^*\}_{j\in\mathcal{J}^i}, \mathbf{R}_{i\times|\mathcal{C}|}, \bar{\mathcal{W}}_{i-1}^*, \mathcal{D}_a)$
8      $\{\mathbf{x}_j\}_{j\in\mathcal{J}^i} \leftarrow$ top-layer of $\{\mathcal{W}_{i,j}^*\}_{j\in\mathcal{J}^i}$
9      $\{\theta_j\}_{j\in\mathcal{J}^i} \leftarrow f^{\mathcal{W}_d}(\{\mathbf{x}_j\}_{j\in\mathcal{J}^i})$
       /* adaptive client selection */
10     $\Lambda \leftarrow$ search grid from $\{\Lambda_s, \Lambda_l\}$
11     for $j \in \mathcal{J}^i$ do
12         $\{\mathcal{T}^t\}_{t\in\Lambda} \leftarrow$ grid search over $\Lambda$
13     $\mathcal{T} \leftarrow \{\mathcal{T}^t\}_{t\in\Lambda}$ obtaining best accuracy on $\mathcal{D}_a$
14     for $j \in \mathcal{J}^i$ do
15         $s_j \leftarrow (j \in \mathcal{T})$ ? 1 : 0
       /* work with any aggregation scheme */
16     $\bar{\mathcal{W}}_i^* \leftarrow$Aggregation $(\{\mathcal{W}_{i,j}^*\}_{j\in\mathcal{J}^i}, \{s_j\}_{j\in\mathcal{J}^i})$
```

Figure 3 shows an example result of client utility inference. We observe that our learning algorithm can effectively estimate the utility of clients with different noise ratios.

C. Adaptive Client Selection

Given the inferred client utility θ_j, a straightforward approach for client selection would be to sample directly s_j from the Bernoulli distribution parameterized by θ (Eq. (4)). Yet, as we motivated earlier from empirical observations, it is important to account for the different roles of clients with various utilities while the training for the global model proceeds. Specifically, while clients with high utility should be the focus in the initial rounds of training (i.e., *Go Narrow*), overtime it helps to also include clients with less utility that can contribute to the training in a positive way (i.e., *Go Wide*). This can be done by selecting the clients based on client utility θ_j and lowering the threshold for client selection in terms of their utility when training proceeds.

Due to the data- and task-specific nature of the global model, it's likely that there is not a one-size-fits-all configuration of the threshold. We, therefore, introduce the threshold for the client selection by a hyperparameter configured as a range of thresholds (e.g., from 0.1 to 0.5 with a step size of 0.1). With such an idea, FedNaWi adopts an adaptive algorithm that uses grid search to find the optimal utility threshold within a specified range. Given the inferred utility of clients, the algorithm selects ones having the utility above the threshold in Λ and accordingly aggregates their updates for the global

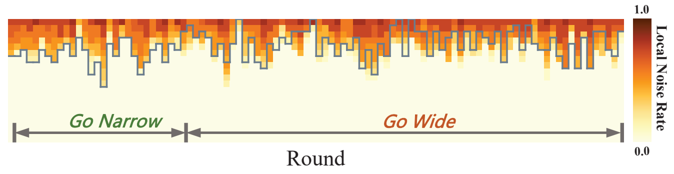


Fig. 4: Client selection results of each round using FedNaWi with adaptive thresholds on CIFAR10 (30% clients with random noise). Each cell represents a client and cells in the same column represent all clients in one round. The color bar denotes noise rate of clients. Below grey line of each column represents selected clients for aggregation in the current round.

model. In this way, we obtain several combinations of the local updates for aggregation. Then we evaluate the corresponding global models on the auxiliary dataset $\mathcal{D}_a$ and choose the threshold by which the aggregated global model has the best performance as the optimal one.

Following the idea of first going narrow and then wide in client selection, the Λ has to be switchable between two client selection strategies. This is motivated by practical considerations including training efficiency and the long-term performance of the global model. On the one hand, we observe that when only clients with high utilities are selected, global model training is highly efficient: we can therefore only involve a small group of relatively high thresholds as Λ_s for the going narrow phase. For instance, candidates in Λ_s are ranging from 0.3 to 0.5 with an interval of 0.1. On the other hand, when the global model performance reaches a predefined expectation, global model training makes conversion into obtaining a higher accuracy in the long run – we can then encompass smaller thresholds to enlarge the search grid to Λ_l with scope between 0.1 to 0.5. Figure 4 gives an example of clients selected in each round using our algorithm. We observe a clear pattern that only high-utility clients are selected in the beginning, and over time, relatively low-utility clients are also included.

Algorithm 1 describes the overall FedNaWi process. In each round, local models in activated clients are first updated as in standard FL settings **(rows 2-4)**. The algorithm then updates on the server the round-reputation matrix and infers client utility by calling the variational utility inference algorithm **(rows 5-9)**. Based on that, it then selects the clients by calling the adaptive client selection algorithm **(rows 10-15)**. It finally obtains the global model by aggregating the weight parameters from selected clients **(row 16)**.

D. Early Stopping Strategy

The utility estimated by FedNaWi gets stabilized after the client has been selected for several rounds when the learning proceeds. This allows for improving the learning efficiency by designing an early stopping mechanism that excludes clients from participating in utility inference, thus reducing computational overhead. We take the mean of recorded utility as the frozen utility when clients are consistently marked as high-utility or low-utility in the round-reputation matrix $\mathbf{R}$ for consecutive several rounds. Figure 5 shows the effectiveness of

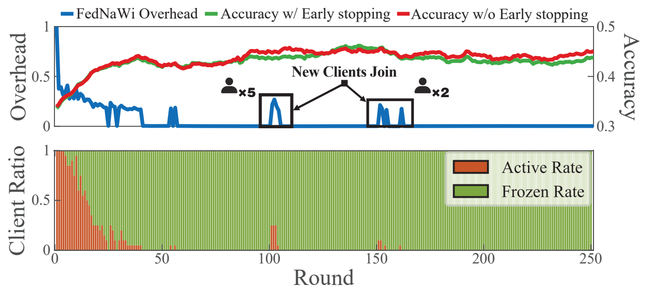


Fig. 5: Effectiveness of early stopping mechanism: (normalized) time overhead of FedNaWi on server (top) and proportion of clients selected for early stopping (bottom). Black boxes denote new connections. Active Rate means the ratio of clients in the current round whose utility is still undetermined.

the early stopping mechanism. We observe that over time, the proportion of clients triggering the early stopping (i.e., frozen rate) increases, and along with that, the overhead of FedNaWi module (mainly on utility inference) decreases. These dynamics have little impact on model accuracy (green curve v.s. red curve in Figure 5). The early stopping mechanism also shows flexibility when new clients join the FL progress. The extra overhead introduced by FedNaWi only lasts for several rounds until the inferred utility of new clients converges.

IV. Performance Evaluation

A. Experimental Settings

1) Datasets: We evaluate FedNaWi using three datasets: CIFAR10 [19], Google Speech [20], HARBox [21]. CIFAR10 is an image dataset for object recognition (10 classes including *horse*, *ship*, etc.). Google Speech is an audio dataset for spotting command keywords in IoT applications; it contains 105,829 utterances of 1-second duration each, corresponding to 35 command keywords (e.g., numbers from *zero* to *nine*, *up, down, stop, go*, etc.). HARBox is a human activity recognition database collected from on-board inertial sensors (accelerometer and gyroscope) for distinguishing daily activities: *walking, hopping, phone calls, waving, and typing*. These datasets cover three modalities (i.e., image, audio, and physical measurements) concerning three typical tasks on edge devices for IoT. Google Speech and HARBox are collected from real participants thus are inherently suitable for FL scenarios. For all the above datasets, the split of the test set and training set in our experiments is based on the scheme given by the dataset providers. We implement task-specific deep learning models for each dataset: shallow CNNs [20, 21] for Google Speech and HARBox, with different hyper-parameters to fit the input dimension and MobileNet-v2 for CIFAR10.

2) Baselines: FedNaWi is generic in that it can be applied to improve existing FL methods. We evaluate FedNaWi on the following SOTA FL methods as baselines. 1) **FedAvg [1]** is the original FL scheme. Clients conduct SGD on local data and offload the model updates to the central server where the global model gets updated by equally aggregating all local updates. 2) **FedAvgM [22]** adds momentum when updating the global model to dampen oscillations caused by the sparse distribution across local data. Using Nesterov accelerated gradient [23] with momentum β, central aggregation considers accumulative gradient history. 3) **FedProx [24]** modifies the local objective by adding a proximal term measuring the similarity between local weights and global weights, which will restrict the local model to be closer to the global model when minimizing the objective function. A hyper-parameter μ is used to control the weight of the proximal term in the objective function.

We further compare FedNaWi with three other client selection methods. 1) **OnlyIC** is a hypothetical client selection method that only selects updates from the inerrant clients (those without noisy data) for aggregation. This gives the upper bound of model performance when the aim is to select only reliable clients for aggregation. 2) **FLDebugger [5]** is a client selection method using the 2-norm distance between local weight parameters and global weight parameters, that is, $\|\mathcal{W}^*_{j,i} - \bar{\mathcal{W}}^*_j\|$. 3) **FedCorr [4]** calculates average Local Intrinsic Dimension (LID) of local prediction vectors $\{f^{\mathcal{W}^*_{j,i}}(x)\}_{x\in\mathcal{D}_i}$ for each client. The server applies a Gaussian Mixture Model on received LID scores to partition candidate clients $\mathcal{C}$ into two subsets: noisy clients and clean clients. Note that the noisy client selection in [4] is a pre-processing step that has to be completed before FL training starts.

3) Data Partition: Our experimental setup considers various three types of non-IID partitions that are commonly found in real-world scenarios. Suppose that a dataset has n training samples in c classes, and there are m candidate clients involved in the FL task. 1) **Non-IID (H2C)** means that all the clients hold only two classes of samples. The size of local data is n/m in each client, where each of the two classes occupies $n/2m$ samples. To achieve this, we first divide the training sample of each class into $2m$ shards and then randomly assign two shards belonging to different classes to each client. 2) **Non-IID (Dirichlet)** uses Dirichlet distribution to synthesize non-IID partition following previous work [22]. Local data follows the Dirichlet allocation parameterized by a vector $q \in \mathbb{R}^n$ where $q \backsim Dir(\alpha q)$. A smaller concentration parameter α leads to a more diverse distribution from the prior class distribution q. 3) **Non-IID (Non-Synthetic)** follows the natural split of the dataset by different users. We follow the suggestion of the dataset authors to assign the motion data of the 100 participants in HARBox in original training sets to the clients.

4) Noise Injection: To better simulate the local noise, the kinds of noise and noise level for the corrupted clients are equally important to the noise injection. Although there are several datasets inherently containing label noise [4, 25, 26], it is challenging to realize the dynamic level of noise across corrupted clients as the difficulty picking the corrupted samples from the whole dataset. To better manipulate the noise injection, we designate the noisy clients with varied levels of synthetic noise considering both label and feature noise.

To this end, we choose clients in the proportion of ϵ_c as corrupted clients and then inject the noise to local samples

TABLE II: Performance of FL methods trained with FedNaWi and with other client selection methods.

ϵ_c	FedAvg				FedAvgM				FedProx			
	w/o	**w/ FedNaWi**	**w/ FedCorr**	**w/ FLDebugger**	**w/o**	**w/ FedNaWi**	**w/ FedCorr**	**w/ FLDebugger**	**w/o**	**w/ FedNaWi**	**w/ FedCorr**	**w/ FLDebugger**
0.1	0.470	**0.473** $_{\uparrow 0.5\%}$	0.464 $_{\downarrow 1.3\%}$	0.369 $_{\downarrow 21.5\%}$	0.458	**0.462** $_{\uparrow 0.8\%}$	0.457 $_{\downarrow 0.2\%}$	0.363 $_{\downarrow 20.7\%}$	0.459	**0.464** $_{\uparrow 0.9\%}$	0.458 $_{\downarrow 0.2\%}$	0.375 $_{\downarrow 18.3\%}$
0.2	0.437	**0.478** $_{\uparrow 9.4\%}$	0.444 $_{\uparrow 1.6\%}$	0.391 $_{\downarrow 10.5\%}$	0.427	**0.452** $_{\uparrow 5.9\%}$	0.437 $_{\uparrow 2.3\%}$	0.402 $_{\downarrow 5.9\%}$	0.434	**0.452** $_{\uparrow 4.1\%}$	0.440 $_{\uparrow 1.4\%}$	0.392 $_{\downarrow 9.7\%}$
0.3	0.441	**0.460** $_{\uparrow 4.3\%}$	0.435 $_{\downarrow 1.4\%}$	0.414 $_{\downarrow 6.1\%}$	0.399	**0.438** $_{\uparrow 9.8\%}$	0.430 $_{\uparrow 7.8\%}$	0.405 $_{\uparrow 1.5\%}$	0.433	**0.439** $_{\uparrow 1.4\%}$	0.435 $_{\uparrow 0.5\%}$	0.417 $_{\downarrow 3.7\%}$
0.4	0.434	**0.449** $_{\uparrow 3.4\%}$	0.431 $_{\downarrow 0.5\%}$	0.417 $_{\downarrow 3.8\%}$	0.409	**0.453** $_{\uparrow 10.8\%}$	0.433 $_{\uparrow 5.9\%}$	0.407 $_{\downarrow 0.5\%}$	0.432	**0.440** $_{\uparrow 1.8\%}$	0.428 $_{\uparrow 1.0\%}$	0.415 $_{\downarrow 3.9\%}$
0.5	0.404	**0.424** $_{\uparrow 5.0\%}$	0.416 $_{\uparrow 3.0\%}$	0.396 $_{\downarrow 2.0\%}$	0.398	**0.428** $_{\uparrow 7.5\%}$	0.411 $_{\uparrow 3.3\%}$	0.399 $_{\uparrow 0.3\%}$	0.405	**0.428** $_{\uparrow 5.7\%}$	0.413 $_{\uparrow 2.0\%}$	0.403 $_{\downarrow 0.5\%}$

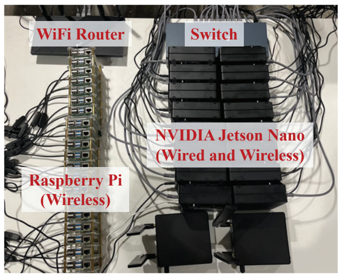


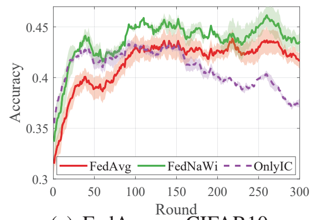
Fig. 6: Our built testbed: 20 Raspberry Pi and 18 NVIDIA Jetson Nano connect to an FL server via WiFi/Ethernet.

in the proportion of ϵ_f. To better emulate the real-world user noise, we vary the flipping rate across corrupted clients by randomly sampling $\epsilon_f \in [0.1, 1]$ for different clients. For label noise, we consider *random* and *structural* label noise. Random label noise is injected by flipping the labels of corrupted samples to other classes following a uniform distribution. The conversion can be formulated by the following symmetric transition matrix $\mathbf{T}$ [27] that describes the probability of flipping i-th class to j-th class: $\mathbf{T}_{i,i} = 1 - \epsilon_f$, and $\mathbf{T}_{i,j} = \frac{\epsilon_f}{c-1}$. For structural label noise, we consider asymmetric transition matrices created in (three) different ways (i.e., *Weighted, Targeted, and Out of Vocabulary*). Besides label noise, we further consider feature noise (i.e., *Gaussian Noise, Resolution, and Corruption*). The details are in Section IV-D.

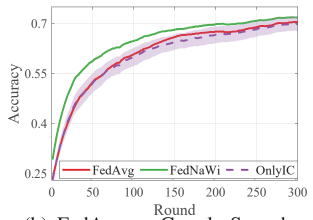
5) Implementation Details: We implement all FL methods in Python and neural networks with PyTorch. Besides, for evaluation on the IoT-related HARBox dataset, we implement an FL testbed containing 20 Raspberry Pi 4B and 18 Jetson NANO as clients, and a laptop that has an NVIDIA 3080 GPU as a server. Each device runs multiple clients and connects to the server through WiFi/Ethernet, as shown in Figure 6.

B. Effectiveness on H2C Data

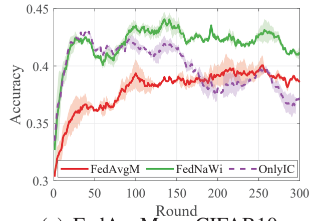
We proceed to the non-IID (H2C) setting where each client only holds two classes of samples. Figures 7 shows the evolving performance of STOA FL methods FedAvg, FedAvgM, and FedProx trained with and without FedNaWi in different communication rounds, respectively with client flip rate $\epsilon_c = 0.3$ on CIFAR10 and Google Speech. First and foremost, we observe that FedNaWi substantially improves the performance of all three baselines FL methods on both datasets. Second, we observe that compared with OnlyIC which selects only inerrant clients, the performance of all the three FL methods trained without FedNaWi is significantly lower on CIFAR10 in the initial communication phase. Noise, therefore, has a strong impact on FL. On the other hand, we also observe that the performance of FL methods (trained without FedNaWi) continues



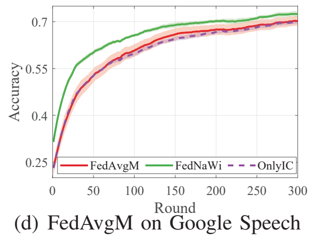
(a) FedAvg on CIFAR10



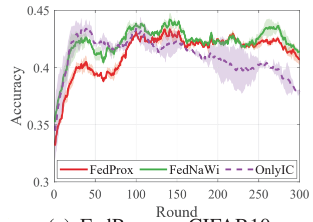
(b) FedAvg on Google Speech



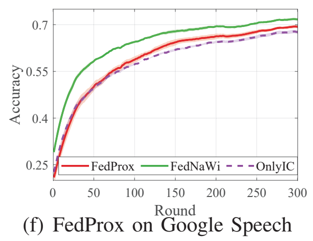
(c) FedAvgM on CIFAR10



(d) FedAvgM on Google Speech



(e) FedProx on CIFAR10



(f) FedProx on Google Speech

Fig. 7: Performance of SOTA FL baselines trained with and without FedNaWi in different communication rounds on CIFAR10 (left) and Google Speech (right), with $\epsilon_c = 0.3$.

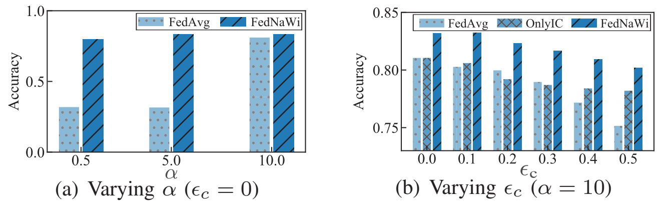
to grow when the training proceeds, which can sometimes (but not always) lead to higher performance than OnlyIC (e.g., FedAvg). These results indicate a tension between the positive and negative effects unreliable clients with noisy data can have in terms of their contribution to FL. The contribution depends on the specific FL method, as we have observed on CIFAR10, but also on the specific dataset: on Google Speech, a less dataset where FL can reach much higher performance than on CIFAR10, we observe no significant difference in FL performance between selecting only inerrant clients and selecting also unreliable ones. Importantly, when trained with FedNaWi, we observe on both datasets that all FL methods match or outperform OnlyIC performance in initial rounds and later on, continuously improve global model to a performance significantly higher than both OnlyIC and original versions without FedNaWi. These results confirm that FedNaWi can get the best of both worlds: it takes advantage of reliable clients, in the beginning, to quickly improve global model performance and further benefits from less reliable clients to continuously improve the global model in the long run.

In Table II, we compare three SOTA baselines trained with FedNaWi and with other client selection methods. Besides, we also quantify the accuracy improvement over the baseline under each setting. Interestingly, we observe that FLDebugger does not perform well when the client flip rate is small (i.e., $\epsilon_c = 0.1, 0.2$). This is because there is no significant deviation between local weight parameters and global weight parameters. FedCorr yields performance improvements in most cases; yet, FedNaWi still outperforms it consistently. Besides, unlike FedCorr which requires a pre-processing step to select clients before FL, FedNaWi is flexible to take in new clients during the learning process. FedNaWi is therefore more favorable in terms of both its effectiveness and flexibility.

C. *Effectiveness on Dirichlet Data*

We now evaluate FedNaWi on the non-IID (Dirichlet) setting where the local data follows a Dirichlet distribution. We first investigate the negative effect of the heterogeneous local data distribution. Figure 8(a) shows the inference accuracy under the scenarios that data in all clients are clean but the distributions of local samples go from imbalanced ($\alpha = 0.5$, and 5) to nearly IID ($\alpha = 10$). The benefit of FedNaWi is obvious when we compare it with FedAvg – equivalent to OnlyIC when no noise is present, which suffers from severe accuracy decrease when $\alpha = 0.5$ or 5 as compared with $\alpha = 10$. The fact that FedAvg still suffers from data heterogeneity gives explicit evidence of the negative impact given by the imbalanced distribution of data classes on clients. In comparison, FedNaWi retains the performance however imbalanced the data distribution is, which signifies the potential of our FedNaWi in dealing with the issue.

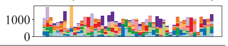
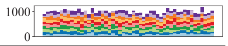
To investigate the impact of noise, we fix the concentration parameter α of Dirichlet distribution to 10 and vary the client flip rate ϵ_c from 0.1 to 0.5. Figure 8(b) shows the result. First, we observe a change in the relative performance between FedAvg and OnlyIC when the number of noisy clients increases: there is a narrow margin between FedAvg with and without OnlyIC when ϵ_c is small (until 0.3); instead, when ϵ_c further increases, OnlyIC starts to show a significant advantage over FedAvg. The initial advantage of FedAvg can be attributed to the robustness of FedAvg in dealing with data noise when the data distribution is not extremely imbalanced. The higher inference accuracy of original FedAvg compared with OnlyIC (i.e., ϵ_c = 0.2 and ϵ_c =0.3) further shows the benefit of involving less reliable clients with noise (note that in our setting with ϵ_c, OnlyIC involves $\epsilon_c \times 100\%$ fewer clients in training global model compared to FedAvg). When the number of noisy clients further increases, data noise starts to play an important role in affecting FL performance. Most importantly, we observe that FedNaWi consistently outperforms both FedAvg and OnlyIC however ϵ_c changes; especially compared to FedAvg, we observe a much smaller performance decrease of FedNaWi when ϵ_c increases. These results signify the effectiveness of FedNaWi in getting the best of both worlds (reliable and less reliable clients) in dealing with data noise.



(a) Varying α ($\epsilon_c = 0$) (b) Varying ϵ_c ($\alpha = 10$)

Fig. 8: Performance on Dirichlet distribution: FedNaWi (i.e., FedAvg w/ FedNaWi) is resilient to imbalanced distribution (Left) while also able to alleviate noise impact (Right).

TABLE III: FedNaWi performance under different noise types (CIFAR10 w/ Dirichlet distribution, varying α ($\epsilon_c = 0.3$)).

Method	Noise	$\alpha = 0.5$(**distribution below**) 			$\alpha = 10$ 		
		w/o	**w/ FedNaWi**		**w/o**	**w/ FedNaWi**	
			Aligned	**Fixed**		**Aligned**	**Fixed**
FedAvg	W/o Noise	0.32	-	0.80 $_{\uparrow 150\%}$	0.81	-	0.83 $_{\uparrow 2.5\%}$
	Random	0.35	0.76 $_{\uparrow 117\%}$	0.76 $_{\uparrow 117\%}$	0.77	0.82 $_{\uparrow 6.5\%}$	0.82 $_{\uparrow 6.5\%}$
	Targeted	0.34	0.75 $_{\uparrow 121\%}$	0.75 $_{\uparrow 121\%}$	0.78	0.81 $_{\uparrow 3.8\%}$	0.81 $_{\uparrow 3.8\%}$
	Weighted	0.36	0.75 $_{\uparrow 108\%}$	0.75 $_{\uparrow 108\%}$	0.78	0.82 $_{\uparrow 5.1\%}$	0.82 $_{\uparrow 5.1\%}$
	Out-of-Voc	0.42	0.73 $_{\uparrow 74\%}$	0.75 $_{\uparrow 79\%}$	0.79	0.82 $_{\uparrow 3.8\%}$	0.82 $_{\uparrow 3.8\%}$
	Gaussian	0.39	0.75 $_{\uparrow 92\%}$	-	0.76	0.82 $_{\uparrow 7.9\%}$	-
	Resolution	0.37	0.77 $_{\uparrow 108\%}$	-	0.78	0.82 $_{\uparrow 5.1\%}$	-
	Corruption	0.36	0.78 $_{\uparrow 117\%}$	-	0.79	0.83 $_{\uparrow 5.1\%}$	-
FedProx	W/o Noise	0.35	-	0.79 $_{\uparrow 126\%}$	0.81	-	0.84 $_{\uparrow 3.7\%}$
	Random	0.38	0.75 $_{\uparrow 97\%}$	0.75 $_{\uparrow 97\%}$	0.77	0.82 $_{\uparrow 6.5\%}$	0.82 $_{\uparrow 6.5\%}$
	Targeted	0.33	0.75 $_{\uparrow 127\%}$	0.75 $_{\uparrow 127\%}$	0.77	0.81 $_{\uparrow 5.2\%}$	0.81 $_{\uparrow 5.2\%}$
	Weighted	0.36	0.75 $_{\uparrow 108\%}$	0.74 $_{\uparrow 105\%}$	0.78	0.82 $_{\uparrow 5.1\%}$	0.82 $_{\uparrow 5.1\%}$
	Out-of-Voc	0.36	0.77 $_{\uparrow 114\%}$	0.77 $_{\uparrow 114\%}$	0.79	0.82 $_{\uparrow 3.8\%}$	0.82 $_{\uparrow 3.8\%}$
	Gaussian	0.35	0.76 $_{\uparrow 117\%}$	-	0.77	0.82 $_{\uparrow 6.5\%}$	-
	Resolution	0.40	0.77 $_{\uparrow 93\%}$	-	0.80	0.82 $_{\uparrow 2.5\%}$	-
	Corruption	0.36	0.78 $_{\uparrow 117\%}$	-	0.80	0.83 $_{\uparrow 3.8\%}$	-

D. *Effectiveness on Diverse Noise Types*

Here we evaluate FedNaWi against structural label noise and feature noise. We created three types of structural noise [28], by 1) assigning labels of corrupted samples to the most often confused class, referred to as *Targeted* noise; 2) *Weighted* noise assigning labels of corrupted samples to other classes with weights determined by the confusion matrix of the learned centralized model; and 3) replacing some training samples with out-of-distribution images from the first 10 classes of CIFAR100 [19], referred to as *Out of Vocabulary* noise [29]. Besides these structural label noise, we also consider feature noise [30] including 1) *Gaussian* random noise (with mean 0.2 and variance 1.0), 2) *Corruption* noise, where 50% of an image is set to black, and 3) *Resolution* distortion, where images are resized to 4×4 and then dilated back to 32×32.

Table III shows the performance of FedAvg and FedProx trained with and without FedNaWi in the presence of different noise types. We specifically consider two scenarios where the noise type of local data is known and unknown; in the former case, we create unreliable auxiliary data $\mathcal{D}_a^-$ with the same noise type as in local data, whereas in the latter case $\mathcal{D}_a^-$ is kept with random label noise, referred to as **Aligned** and **Fixed**, respectively. Comparing the original performance of FedAvg and FedProx, FedNaWi can substantially improve their performance across all types of noises; furthermore, the FL performance can reach a similar level under all noise scenarios, and even when the noise type of data in clients

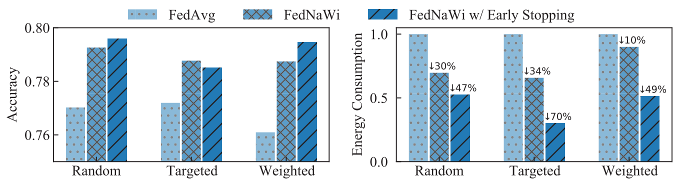


Fig. 9: FedNaWi in resource-constrained scenarios: performance (Left) and energy consumption (Right).

is unknown. These results provide strong evidence about the robustness of FedNaWi to data noise. For both FedAvg and FedProx in the small α condition, noisy updates may improve the global model accuracy. It shows that additional noise has a counter effect on FL performance, which is a very interesting observation for exploration in future work.

E. Resource-Constrained Scenarios

Finally, we evaluate FedNaWi in the resource-constrained scenario across different types of noise and a larger number of testbed clients using the HARBox dataset. HARBox contains data collected from 120 different users, and we take 100 users as the training data and augment the number of clients from 100 to 200 by assigning the data of each user to two clients. Considering data across users in HARBox is low Non-IID (relatively balanced data distribution) [8], such augmentation would enable the testbed to be more challenging and better approximate real-world scenarios. Given the stability issue of client-server communication and that device resources are often constrained in practical applications, we report the best inference accuracy of global models training within limited rounds (i.e., 200 rounds), and the energy consumption (measured by the power meter) when the global models achieve the same performance (i.e., the best accuracy of FedAvg) in Figure 9. We observe that FedNaWi increases FL performance by 3.3%, 2.9%, and 3.5% in the same rounds for three different noise types, respectively, showing the consistent time-to-accuracy advantage of FedNaWi. Besides, FedNaWi is more energy-efficient even with the extra overhead at the server: it reduces the energy consumption by up to 34%. With an early stopping mechanism, energy consumption is further reduced by 70%. This is mainly due to the fewer training rounds required when FL is trained with FedNaWi: it requires only 36.2%, 19.4%, and 61.9% rounds to reach FedAvg's best performance for the three noise types, respectively.

V. Conclusion

In this paper, we observed that unreliable clients in FL can slow the learning of the global model and reduce its accuracy, but they can still contribute to learning a more accurate global model in the long run. Inspired by these empirical observations, we designed an adaptive "Go Narrow, Then Wide" client selection method that first selects reliable clients to allow the global model to improve its performance quickly and then includes less reliable clients to exploit noisy data for further improvement of the global model. Extensive evaluations indicate FedNaWi could help FL systems become more robust in practical deployments

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017.
[2] A. Konovalov, B. Strauss, A. Ritter, and B. O'Connor, "Learning to extract events from knowledge base revisions," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2017.
[3] X. Ren, Z. Wu, W. He, M. Qu, and et.al., "Cotype: Joint extraction of typed entities and relations with knowledge bases," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2017.
[4] J. Xu and et.al., "Fedcorr: Multi-stage federated learning for label noise correction," in *Proceedings of IEEE/CVF Conference on CVPR*, 2022.
[5] A. Li, L. Zhang, J. Wang, J. Tan, F. Han, Y. Qin, N. M. Freris, and X.-Y. Li, "Efficient federated-learning model debugging," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021.
[6] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, and et.al., "{FLAME}: Taming backdoors in federated learning," in *31st USENIX Security Symposium*, 2022.
[7] F. Lai, X. Zhu, H. V. Madhyastha, and et.al., "Oort: Efficient federated learning via guided participant selection," in *USENIX OSDI*, 2021.
[8] C. Li, X. Zeng, and et.al., "Pyramidfl: A fine-grained client selection framework for efficient federated learning," in *Proceedings of Annual International Conference on Mobile Computing And Networking*, 2022.
[9] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, and et.al., "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.
[10] T. Li, A. K. Sahu, and et.al., "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, 2020.
[11] J. Shin, Y. Li, Y. Liu, and S.-J. Lee, "Fedbalancer: Data and pace control for efficient federated learning on heterogeneous clients," 2022.
[12] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE international conference on communications (ICC)*, 2019.
[13] S. Q. Zhang, J. Lin, and Q. Zhang, "A multi-agent reinforcement learning approach for efficient client selection in federated learning," *arXiv preprint arXiv:2201.02932*, 2022.
[14] J. Kang, Z. Xiong, D. Niyato, Y. Zou, and et.al., "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, 2020.
[15] H. Moudoud and et.al., "Towards a secure and reliable federated learning using blockchain," in *IEEE Global Communications Conference*, 2021.
[16] X.-C. Li and D.-C. Zhan, "Fedrs: Federated learning with restricted softmax for label distribution non-iid data," in *Proceedings of ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.
[17] D. G. Tzikas, A. C. Likas, and et.al., "The variational approximation for bayesian inference," *IEEE Signal Processing Magazine*, 2008.
[18] D. M. Blei, A. Kucukelbir, and et.al., "Variational inference: A review for statisticians," *Journal of the American statistical Association*, 2017.
[19] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
[20] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.
[21] X. Ouyang and et.al., "Clusterfl: a similarity-aware federated learning system for human activity recognition," in *Proceedings of International Conference on Mobile Systems, Applications, and Services*, 2021.
[22] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.
[23] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical programming*, 2013.
[24] T. Li, A. K. Sahu, and et.al., "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, 2020.
[25] T. Xiao and et.al., "Learning from massive noisy labeled data for image classification," in *Proceedings of the IEEE conference on CVPR*, 2015.
[26] Y. Chen, X. Yang, X. Qin, and et.al., "Focus: Dealing with label quality disparity in federated learning," *arXiv preprint arXiv:2001.11359*, 2020.
[27] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and et.al., "Co-teaching: Robust training of deep neural networks with extremely noisy labels," *Advances in neural information processing systems*, 2018.
[28] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, "Deep learning is robust to massive label noise," *arXiv preprint arXiv:1705.10694*, 2017.
[29] T. Tuor and et.al., "Overcoming noisy and irrelevant data in federated learning," in *International Conference on Pattern Recognition*, 2021.
[30] Y. Wang, W. Liu, and et.al., "Iterative learning with open-set noisy labels," in *Proceedings of the IEEE conference on CVPR*, 2018.