

A APPENDIX

A.1 EXPERIMENT SETTINGS

A.1.1 TESTBED CONFIGURATION

To validate the feasibility of our proposed method, we build a FL testbed with 32 Android-based mobile devices, including 10 models from 7 different manufactures. The detailed hardware configuration for the testbed is listed on the Table 2.

Table 2: Hardware configuration for the mobile device testbed.

Model Name	Internal Memory	RAM	Processor	Amount
OnePlus 9 Pro	256 GB	12 GB	Qualcomm Snapdragon 888	3
OnePlus 7 Pro	256 GB	8 GB	Qualcomm Snapdragon 855	4
SAMSUNG Galaxy S20	128 GB	6 GB	SAMSUNG Exynos 990	2
SAMSUNG Galaxy A53	128 GB	6 GB	Qualcomm SAMSUNG Exynos 1280	4
Google Pixel 6	128 GB	8 GB	Google Tensor	2
Google Pixel 3	64 GB	4 GB	Qualcomm Snapdragon 845	4
LG Velvet 5G	128 GB	6 GB	Qualcomm Snapdragon 765G	4
Moto G Power	64 GB	4 GB	Qualcomm Snapdragon 662	3
Nokia G20	128 GB	4 GB	Mediatek Helio G35	3
Xiaomi Redmi Note 10	128 GB	4 GB	Qualcomm Snapdragon 678	3

A.1.2 COMPUTING INFRASTRUCTURE

All experiments are conducted by CPU/GPU simulation and real-deployment on testbed. The configuration of the testbed was elaborated in Section A.1.1. The simulation experiments are conducted on a computing server with one GPU. The server is equipped with AMD EPYC 7502 32-Core Processor and 1024G memory. The GPU is NVIDIA RTX A4000.

A.1.3 DATASETS AND MODELS

AI Benchmark. AI Benchmark Ignatov et al. (2019) is a public dataset that is designed for evaluating the performance of important AI tasks on mobile devices. AI Benchmark provides diverse models’ training and inference speed across various devices, including chipsets from Qualcomm, HiSilicon, Samsung, MediaTek, and Unisoc. Figure 9a illustrates the distribution of the computation efficiency across clients in the AI Benchmark. The slowest device would take around $13.3\times$ computational times than the fastest device for the same task. To approach the dynamic availability of devices, such as low-power mode or multi-process running, we design a coefficient w as follows:

$$x \sim \mathcal{N}(1, 0.3)$$

$$w = \begin{cases} 1 & x \leq 1 \\ x & 1 \leq x \leq 1.3 \\ 1.3 & x \geq 1.3 \end{cases} \quad (2)$$

In this work, we assign the values from AI Benchmark as base computation time to the clients to emulate real devices, analogous to the usage in FedScale Lai et al. (2021a). We also generate the coefficient w every round for each client to simulate the natural disturbance to availability. The local computation time in each round equals the product of w and the base computation time for each client.

MobiPerf. MobiPerf is a public dataset for measuring network performance on mobile devices, which collects the available cloud-to-edge network throughput of over 100k worldwide mobile clients. Figure 9b illustrates the distribution of communication consumption of MobiPerf. Note that the best communication channel can be $200\times$ better than the worst one. We randomly assign a value from MobiPerf to a simulated device every communication round to emulate intermittent connectivity in a real deployment.

CIFAR-10. The CIFAR-10 dataset Krizhevsky (2009) consists of 60,000 32×32 colour images in 10 classes. There are 50,000 training images and 10,000 test images. We normalize the images by the mean and standard deviation of the dataset. We evaluate the dataset with ResNet-20 He et al.

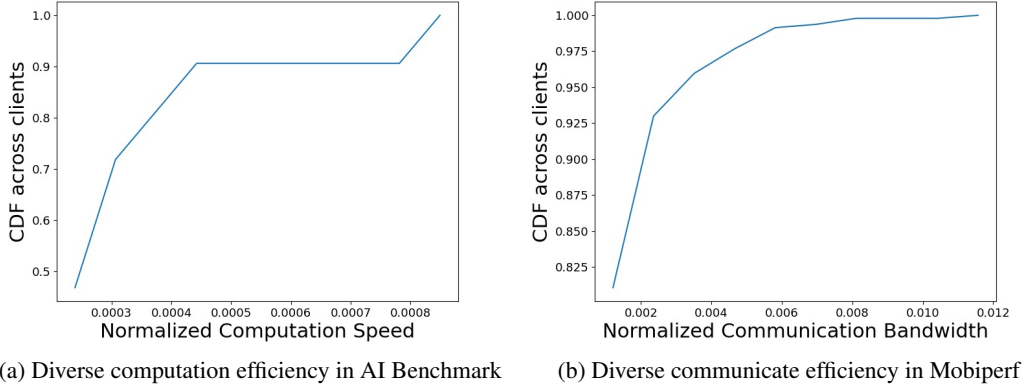


Figure 9: Heterogeneous system utility across simulated clients.

(2016) model. To emulate the realistic non-iid distribution, we partition the dataset using a Dirichlet distribution, following the previous works Nguyen et al. (2021).

Google Command The Google Command speech dataset Warden (2018) covers 105,829 audio recordings collected from 2,618 clients. The training set includes recordings from 2,112 speakers, the validation set includes 256 speakers, and the test set includes 250 speakers. The data set is composed of 35 common words from the everyday vocabulary, such as "Yes", "No", "Up", and "Down". We evaluate the dataset with VGG11 Simonyan & Zisserman (2015) model and a lightweight model based on one related work Zhang et al. (2022) for a 35-class keyword spotting task.

For the VGG11-based experiment on Google Speech Commands, we use the Mel-frequency cepstral coefficients (MFCC) method to pre-process the raw audio data. Specifically, a sequence of overlapping Hamming windows is applied to the raw speech signal with a time shift of 10 ms and window size of 25ms. The MFCC is used for training the keyword spotting model.

For the lightweight model experiment, to pre-process the raw audio data, a sequence of overlapping Hamming windows is applied to the raw speech signal with a time shift of 10 ms. We calculate the discrete Fourier transform (DFT) with a frame length of 1,024 and compute the Mel-spectrogram with a dimension of 128. The Mel-spectrogram is used for training the keyword spotting model. We follow Zhang et al. (2022) for this setup.

Reddit Reddit red consists of comments from 1,660,820 users in the Reddit forum. Each client corresponds to a user, whose data are all of their personal posts. Thus it follows the real non-iid data under FL scenarios. In this dataset, we filter the users with less than 20 words in total and restrict to the 30k most frequently used words, as the same settings in the previous work Lai et al. (2021a). Then, we train the lightweight Albert Lan et al. (2020) model for the next-word-prediction task. The performance is evaluated by the perplexity loss (ppl), which lower is better.

A.1.4 HYPERPARAMETER SETTINGS

We searched for the client learning rate in a range from 10^{-6} to 10^0 , server learning rate in a range from 10^{-4} to 10^0 , input batch size in a range from 5 to 1000, and total training round in a range from 1000 to 10000. The aggregation goal and aggregation participation target is searched from 30% to 50% of [training concurrency per round](#) for FedBuff and TimelyFL, respectively.

After hyper-parameter searching, we fixed the following hyperparameters: for CIFAR-10 related experiments, the total training round is 2000, and training concurrency is 128 for all setups. The aggregation goal and aggregation participation target is 50% of the training concurrency for both FedBuff and TimelyFL. For CIFAR-10 with FedAvg related experiments, the batch size is 8, and the client learning rate is 0.8. For CIFAR-10 with FedOpt related experiments, the batch size is 10, the client learning rate is 0.03, and the server learning rate is 0.001 with ADAM as server optimizer.

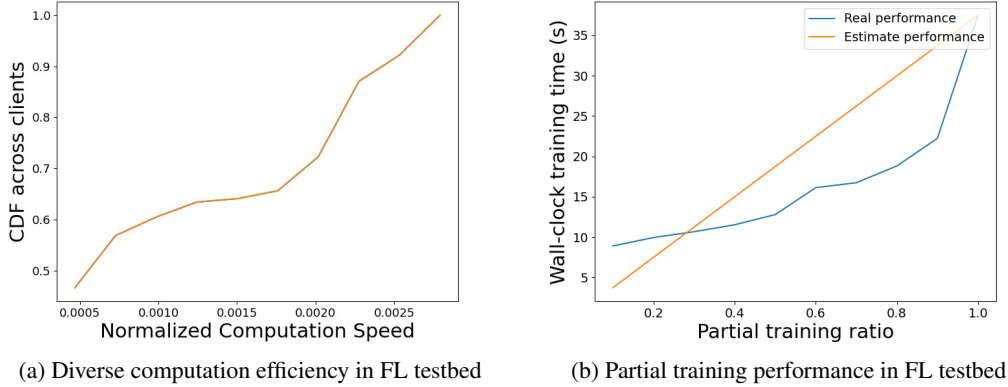


Figure 10: System Performance in FL testbed.

For Google command related experiments with VGG11 model, the total training round is 1000, and training concurrency is 20 for all setups. The aggregation goal and aggregation participation target is 50% of the training concurrency for both FedBuff and TimelyFL. The batch size is 32, and the client learning rate is 0.01. Under the FedOpt, the server learning rate is 0.001 with ADAM as server optimizer.

For Google command related experiments with the lightweight model, the total training round is 5000, and training concurrency is 106 for all setups. The aggregation goal and aggregation participation target is 50% of the training concurrency for both FedBuff and TimelyFL. The batch size is 16, and the client learning rate is 0.1 under the FedAvg. Under the FedOpt, the client learning rate is 0.05 for synchronous FL and TimelyFL, and the client learning rate is 0.2 for FedBuff. The server learning rate is 0.001 with ADAM as server optimizer for all setups.

Finally, for Reddit related experiments, the total training round is 500, and training concurrency is 20 for all setups. The aggregation goal and aggregation participation target is 50% of the training concurrency for both FedBuff and TimelyFL. The batch size is 20, and the client learning rate is 0.0005 for SyncFL and TimelyFL, and 0.0003 for FedBuff. Under the FedOpt, the server learning rate is 0.001 with ADAM as server optimizer.

A.2 SYSTEM PERFORMANCE

A.2.1 PERFORMANCE ON TESTBED

In this work, we implement our strategy and its baselines on the FL testbed. To demonstrate the device heterogeneity in the testbed, we record the computation time for one full epoch training of ResNet20 on each device. As shown in Figure 10a, the fastest device is $4.96\times$ faster than the slowest one.

A.2.2 PARTIAL TRAINING PERFORMANCE

Due to different parameters and tensor shapes among different layers, the training time (computational time of the forward and backward propagation) is not strictly linear to the trainable layer numbers and varies with the model structures. For simplicity and generality, we define the training time of the partial model as the linear multiplication of the training time of the full model and the training ratio. This linear relationship is verified through our real measurement on a Samsung Galaxy S20 with ResNet-20 model using MNN Jiang et al. (2020) library. As shown in Figure 10b, most of the test results are below the linear straight line (except the ratio is below 0.2), justifying the rationality of our choice.

Table 3: Wall clock training time to reach target validation accuracy on benchmark datasets (lower is better).

Dataset	Agg. function	Accuracy	TimelyFL	FedBuff	SyncFL
Google Speech	FedAvg	70%	$2.23 \pm 2.1\%$ hr	$3.55 \pm 1.9\%$ hr (1.59 \times)	$18.37 \pm 0.6\%$ hr (8.24 \times)
		80%	$4.16 \pm 1.3\%$ hr	$6.13 \pm 1.4\%$ hr (1.47 \times)	$32.46 \pm 0.4\%$ hr (7.80 \times)
	FedOpt	70%	$0.48 \pm 1.7\%$ hr	$1.66 \pm 1.0\%$ hr (3.46 \times)	$4.61 \pm 2.1\%$ hr (9.60 \times)
		80%	$1.13 \pm 1.2\%$ hr	$3.25 \pm 0.8\%$ hr (2.88 \times)	$7.47 \pm 1.1\%$ hr (6.61 \times)

A.3 ADDITIONAL EXPERIMENTS

To investigate the effectiveness of the lightweight model on the TimelyFL framework, we implement one lightweight model on the Google Speech Commands dataset for the keyword spotting task.

Following one previous work Zhang et al. (2022), we choose the model that consists of two convolution layers followed by one Gated Recurrent Units (GRU) layer. An average pooling layer is connected to the GRU output, which is then fed through two dense layers to generate the predictions. The parameter size of this model is equal to 79044.

We adapt the same baseline selections as in the Section 4. The hyperparameters for the experiments are listed in the Section A.1.4. The experiment results are summarized in the Table 3. TimelyFL achieves a higher convergence speed compared with the other two strategies before reaching the test accuracy, which confirms the simulation results elaborated in Section 4.2.