Benchmarking and Optimizing Federated Learning with Hardware-related Metrics

Kai Pan[1]
pankai22s@ict.ac.cn
Yapeng Tian[2]
yapeng.tian@utdallas.edu
Yinhe Han[†1]
yinhes@ict.ac.cn
Yiming Gan[†1]
ganyiming@ict.ac.cn

[1] Institute of Computing Technology Chinese Academy of Science Beijing, China

[2] The University of Texas at Dallas Texas, USA

Abstract

Federated learning (FL) serves as an effective way of preserving data privacy at network training through offloading training tasks to different client hardware and aggregation. Real hardware-related metrics such as latency and energy consumption directly decide the performance and accuracy trade-off in federated learning frameworks, yet most FL optimizations do not use real hardware metrics.

In this work, we propose to benchmark federated learning with real measured hardware metrics and optimize FL frameworks through tailoring training hyper-parameters before offloading tasks each round given hardware metrics. With two examples FedAvg and FedOpt, we demonstrate we can significantly save training energy by up to 97.2% and training latency by up to 96.0% while maintaining training accuracy. Source code can be found at https://github.com/RLC-Lab/FEDHW.git.

1 Introduction

Federated learning has been proven to be an effective privacy-preserving training alternative owing to its collaborative nature among independent clients [12, 48]. Nevertheless, the practice of offloading time and energy-consuming training tasks to resource-constrained clients, coupled with frequent data communication, typically results in significantly longer convergence latency and higher energy consumption in federated learning. Consequently, optimizing the latency and energy consumption of federated learning algorithms is equally critical while seeking to improve accuracy.

Recent efforts within the federated learning community have primarily aimed to reduce communication costs to mitigate latency and save energy consumption, given that many federated learning algorithms involve sharing heavy model weights or gradients [24, 34, 43]. However, we argue that a sole focus on communication optimization may not be sufficient

† Corresponding Author

for two fundamental reasons. First, there is a limited body of work that examines the real hardware performance regarding the latency and energy consumption of each process within federated learning algorithms. Amdahl's Law[1] suggests that exclusively concentrating on one aspect of the entire system may result in constrained overall system improvements. Second, attempts to reduce or compress weights shared between clients and central servers can lead to performance degradation [42, 48].

In this work, we take the first step of benchmarking and optimizing federated learning algorithms using real hardware performance. We systematically characterize various federated learning frameworks based on their system-wide latency and energy consumption, leveraging real hardware performance metrics. The benchmarking methodology we propose incorporates hardware-related metrics for all three key components of federated learning algorithms: the latency and energy consumption during the training process on clients, the model sharing between clients and the server, and the model processing on the server.

We observe that among popular federated learning algorithms, the local model training on clients incurs the highest latency and energy consumption, primarily due to the intensive workloads and frequent data loading from memory. The second-highest latency and energy consumption is associated with central server processing the global model. Notably, transmission between the server and clients exhibits the lowest latency and energy consumption.

Building upon our observations, we introduce an optimization framework for federated learning that incorporates real hardware-related metrics. The framework continuously gathers accuracy results from server-aggregated models and hardware-related metrics from either real hardware or simulators. By formulating the training process as a constrained optimization problem with discrete solution space, our framework schedules the optimal training parameters, such as the number of clients used in each round and the training epoch, with the aim of achieving the lowest latency or energy consumption while maintaining accuracy.

Our results indicate that the framework significantly improves latency and energy consumption in popular federated learning algorithms. Our optimization framework achieves a 62.6% reduction in total latency or 56.4% energy consumption compared to state-of-the-art federated learning algorithms, without sacrificing accuracy.

The main contributions of our paper can be summarized as follows:

- We systematically propose an end-to-end characterization of federated learning algorithms with real hardware metrics such as training latency and energy consumption. We break down the latency and energy consumption of individual components and the results show that a significant amount of latency and energy are spent on the training process happening on local clients and central server processing.

- We propose a framework that collects data from both the server to maintain accuracy and real hardware to reduce latency and energy consumed in the federated learning process. The scheduling process is formulated as a constraint optimization problem.

- We implement two instances of our optimization framework with a simulated annealing optimizer and a genetic algorithm optimizer. We significantly improve the total latency and energy consumption of federated learning without losing any accuracy.

2 Related Work

2.1 Federated Learning (FL)

Federated learning is a novel distributed training methodology designed to address privacy concerns. The pioneering work in federated learning, FedAvg [25], trains a global model by weighted averaging different local models trained on their respective partial datasets. Gradient descent occurs on clients, while model aggregation occurs on the server. Building upon FedAvg, other methods have been developed that involve sending partial weights [13, 23] or gradients [30] to the server.

Federated learning indeed presents several challenges alongside its privacy benefits. Since no single client possesses all the data, the accuracy of many federated learning algorithms tends to be weaker compared to centralized training approaches [2]. Moreover, due to the frequent sharing of model weights in federated learning, there is often higher latency and energy consumption during model training [22].

2.2 Optimizing Federated Learning

Federated learning has sparked numerous efforts aimed at enhancing its performance, with a primary focus on improving accuracy. Various strategies have emerged to address accuracy challenges arising from data heterogeneity, including techniques such as partial network acceptance [23], dataset combination [40, 45, 51], and mitigation of aggregation drift [6, 7, 13, 16, 21, 49, 53].

Energy consumption stands out as another key optimization target in federated learning. Given the additional communication costs it incurs, most approaches prioritize minimizing energy usage by optimizing communication energy [20, 27, 38, 44, 46, 47]. Some studies delve deeper into this issue, exploring methods to conserve end-to-end training energy through techniques like reinforcement learning or knowledge distillation [14, 18, 20, 47].

In addition to accuracy and energy considerations, training latency is a significant concern in federated learning. Local model aggregation can sometimes delay training due to its impact on the global model. Researchers have explored various approaches to address this issue, including improving the aggregation process [41], probabilistic local model selection [8, 35], and leveraging reinforcement learning techniques [50].

Despite extensive efforts to optimize federated learning algorithms, two fundamental issues persist. Firstly, many optimization frameworks lack real hardware-measured data, relying instead on assumptions. Secondly, the bottleneck of federated learning frameworks can vary with different underlying hardware platforms, yet most optimization frameworks remain fixed.

3 Benchmarking

Basic federated learning frameworks can be described in Equ. 1. N out of M clients are selected every training round where D represents the dataset. $F_i(x)$ is the loss function of the i^{th} local client model.

$$\min_{D} f(x) = \frac{1}{N} \sum_{i=1}^{N} F_i(x) \tag{1}$$

Three key hyper-parameters in federated learning frameworks will influence the accuracy and cost trade-offs. The number of training epochs in every client E, the batch size of each local client BS, and the fraction of clients that are used in each round F.

Unlike accuracy, benchmarking costs of federated learning algorithms are intricately tied to hardware considerations. Different hardware setups can greatly influence total costs, such as training energy and latency. For instance, in the popular FedAvg framework, the server type handling model aggregation, client type managing local training, and communication network type all impact training energy and latency. Therefore, accurate modeling of energy and latency using real hardware or simulators is essential for precise training cost measurements.

3.1 Energy Modeling

Energy consumption for server computation. In federated learning frameworks, servers primarily handle tasks like client selection, aggregating partially-trained models, and updating the global model, with less frequent involvement in actual model training. These servers typically employ high-end CPUs and GPUs for these operations. The total energy consumption on the server is typically the sum of energy from CPUs and GPUs, denoted as $E_{server} = E_s^{cpu} + E_s^{gpu}$.

Energy consumption for client computation. Energy consumption for clients consists of two parts, energy for computation and data access. Recently, most clients have been equipped with dedicated accelerators such as the embedded GPU or Neural Processing Unit (NPU) for local model training. The computations are mainly matrix multiplication and accumulation (MAC). Given an example of a fully connected layer with an input $X \in \mathbb{R}^{1\times c}$ and weights $W \in \mathbb{R}^{c\times d}$, the energy consumption of computing the layer is shown in Equ. 2, where e_{MAC} is the energy consumption of every MAC operation. Energy consumption for client computation can be either measured using tools like power management firmware or calculated with a cycle-accurate hardware simulator such as SCALE Sim v2 [32, 33].

$$E_{comp} = e_{MAC} \times c \times d \tag{2}$$

Energy consumption for data access. When training local models, frequent data access also consumes a significant amount of energy. The most common memory hierarchy is usually composed of a Dynamic Random Access Memory (DRAM), a cache, and a Register File (RF). In neural network training, inputs, weights, and activation maps are loaded from DRAM to cache, and from cache to RF. Thus, the energy consumption for data access are sum of three components E_{DRAM}, E_{Cache}, and E_{RF}. In reality, E_{Cache} and E_{RF} are usually taken into consideration when measuring the energy consumption of the accelerators. E_{DRAM} needs to be measured separately in most cases.

Energy for Communication. Federated learning frameworks generally require frequent communication between clients and servers, leading to high communication energy. Assuming clients and servers are communicating through an ideal WiFi protocol, the communication energy can be calculated using $E_{comm} = P_{wifi} * \frac{Size}{BitRate}$ [5]. where P_{wifi} stands for the power of the Wi-Fi module in the client, $Size$ stands for the amount of data sent to the server, and $BitRate$ stands for transmission speed of the Wi-Fi module.

3.2 Latency Modeling

Training latency is another critical metric in federated learning frameworks, the lower the better. The entire training process needs to undergo T times of "local training - communication - model aggregation" process. For each round, the latency L is determined by the slowest client, which is calculated using $L = L_{server} + L_{client} + L_{comm}$. The entire training latency is formulated by $T \times L$.

4 Optimization Framework for Federated Learning with Hardware-related Metrics

4.1 Optimization Framework

Accurate benchmarking enables us to optimize the training process of federated learning using hardware-related metrics. To achieve this, we design an optimization framework called FEDHW, illustrated in Fig. 1. FEDHW seamlessly integrates with any federated learning framework, operating on the server side. It continuously gathers inputs like latency, energy consumption, and model accuracy, and then uses this data to generate scheduling policies for the next round of training.

Taking energy consumption optimization and FedAvg as examples, FEDHW collects the energy consumption of every client used in the current training round and the model accuracy after aggregating all the local models. It then generates training settings for the next round, focusing on three key parameters: E for the number of epochs in the training round, BS for batch size, and F for the fraction of clients used in the round. Generally, increasing these parameters reduces the training rounds but also leads to higher energy consumption per round. The optimization process aims to find the optimal values for E, BS, and F in each round, as illustrated in Algo. 1.

4.2 Optimizer Design

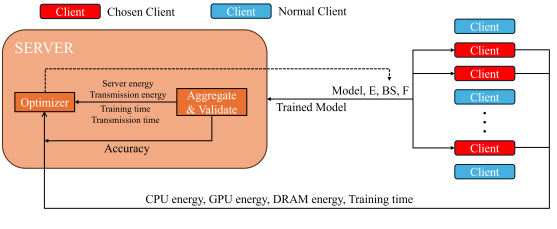


Fig. 1: FEDHW optimization framework.

The design of the actual optimizer plays a crucial role in determining the performance of FEDHW. When designing the optimizer, we prioritize two goals. Firstly, the optimizer must incur low overhead to avoid negatively impacting overall latency or energy consumption. Secondly, since the optimization process is non-convex, the optimizer must capture global optimal points instead of converging to local optima.

We propose two optimizers: FEDHW-SA, which formulates the optimization process as a simulated annealing (SA) algorithm, and FEDHW-GA, which employs a genetic algorithm (GA) approach. In Algo. 2, we describe the workings of FEDHW-SA. Our approach leverages the inherent characteristics of the SA algorithm by treating the hyperparameters E, BS, and F as discrete solution spaces and the energy consumption as the cost function. We introduce random perturbations to E, BS, and F to generate new hyperparameters, which are then evaluated for their simulated consumption. If the new consumption is lower than the

current consumption, the optimizer accepts the new hyperparameters; otherwise, acceptance is based on a probability p. Our design tailors the probability equation to capture long-term energy and latency behavior, as detailed in Equ. 3, where energy as an example.

Algorithm 1: FEDHW Optimization Framework

Input: Epoch E, Batch size BS, Fraction F, other FL setting parameters, Target accuracy Acc, Number of clients NC
Output: trained global model

Initialize the global model ω_t;
while $A_t \leq Acc$ **do**
  $S_t \leftarrow$ randomly select $F_t \times NC$ clients;
  for *each client* $m \in S_t$ **do**
    $\omega_{S_t} \leftarrow$ `ClientUpdate` (E_t, BS_t, ω_t, m);
  end
  $\omega_{t+1} \leftarrow$ `ServerUpdate` (ω_{S_t});
  $A_{t+1} \leftarrow$ validate global model accuracy;
  $O_t \leftarrow$ calculate system consumption;
  $E_{t+1}, BS_{t+1}, F_{t+1} \leftarrow$ `Optimizer` $(E_t, BS_t, F_t, A_t, O_t)$;
end

$$p = e^{-\frac{(E_{sim}-E_t)+(A_{t-1}-A_t)-E_{sim}+\delta}{T_t+\sum^t E}} \tag{3}$$

The probability p takes into account both energy consumption and accuracy considerations. E_{sim} represents the simulated energy consumption using the selected training parameters in the new round, while E_t denotes the current energy consumption observed during the current training round. We introduce a noise factor δ. Additionally, A_{t-1} and A_t denote the global model accuracy, where improvements in accuracy and reductions in energy consumption both influence the probability of selecting a setting. T_t is the temperature parameter in SA algorithm.

To prevent the optimizer from favoring only the best training setting for the next round without considering long-term benefits, we incorporate a history energy consumption accumulation term $\sum^t E$ in the denominator of the probability calculation.

In FEDHW-GA, based on the basic genetic algorithm process, the binary representations of the hyper-parameters E, BS, and F are set as the DNA sequences of the population. The consumption function is then used as the fitness of the individuals to identify the optimal solution in the hyper-parameters through natural iteration. Furthermore, we add the mechanism of accepting new solutions with probability to ensure that the model converges during the optimization process.

5 Experiment

5.1 Experiment Setup

FEDHW is designed to seamlessly integrate with any federated learning framework. To demonstrate the flexibility of this approach, we utilize FEDHW with two distinct federated

Algorithm 2: FEDHW-SA Optimizer

SAOptimizer (E, BS, F, A, O) :
  $T \leftarrow$ set temperature for SA;
  while $T > 1.0$ **do**
    $E_{new}, BS_{new}, F_{new} \leftarrow$ add random perturbation to E, BS, F;
    $O_{sim} \leftarrow$ **ConsumptionSim** $(E_{new}, BS_{new}, F_{new})$;
    if $O_{sim} < O$ **then**
      $E, BS, F \leftarrow E_{new}, BS_{new}, F_{new}$;
      $O = O_{sim}$;
    else
      if *random(0,1)* $< p$ **then**
        $E, BS, F \leftarrow E_{new}, BS_{new}, F_{new}$;
      end
    end
    $T = T \times \lambda$;
  end
  return E, BS, F;

learning frameworks, namely FedAvg and FedOpt. The entire federated learning processes are implemented using the PyTorch framework on a desktop computer [2].

Dataset and Network We evaluate FEDHW using different datasets to demonstrate its effectiveness across various workloads. Specifically, for image classification tasks, we employ a 5-layer convolutional neural network on the MNIST dataset [17] and ResNet-18 on the CIFAR-10 dataset [15]. For audio tasks, we utilize the M18 audio classification network [4] on the ESC50 dataset [19]. Finally, for NLP tasks, we employ a two-layer LSTM network [8] on the R8 dataset [19].

Baseline Setup We train all four models using their default settings with both FedAvg and FedOpt [4, 25, 29, 31]. Each client's dataset is Independent and Identically Distributed (IID). Our results align with the reported accuracy for all four networks. Detailed training parameters and results are provided in Tbl. 1.

Hardware Platforms To mimic edge clients, we utilize NVIDIA Jetson TX2 embedded devices as the client platform [28], featuring a 256-core NVIDIA Pascal GPU architecture. For the server, we employ a desktop equipped with an Intel i9-12900K CPU and an NVIDIA RTX3090 GPU. Communication between the server and clients is established through Wi-Fi.

Table 1: Parameter setting for each model. Acc is the accuracy target of each model, E means epochs, BS means batch size, F means fraction, NC means number of clients.

Dataset	Model	Parameters	FedAvg_Acc	FedOpt_Acc
MNIST	CNN	(E = 20, BS = 10, F = 0.1, NC = 100)	98%	98%
CIFAR10	ResNet-18	(E = 5, BS = 64, F = 0.1, NC = 100)	90%	90%
ESC50	M18	(E = 20, BS = 10, F = 0.2, NC = 50)	68%	68%
R8	LSTM	(E = 50, BS = 5, F = 0.2, NC = 50)	92%	92%

Hardware Metric Acquisition Our method relies on real hardware metrics, specifically latency, and energy consumption. Acquiring latency on clients is relatively straightforward; we monitor training latency on the Jetson TX2 under different settings and introduce a noise parameter to capture variations in latency. Energy consumption by clients is broken down into computation energy and data access energy. To measure computation energy, we read power numbers from special registers on the Jetson TX2 and multiply them by latency. For data access energy on clients, we utilize DRAMSys, a flexible DRAM subsystem design

space exploration framework, to simulate DRAM power during client training [36, 37]. The energy consumption of data access is then calculated by multiplying DRAM power by client training latency. Server energy consumption is recorded using Intel Performance Counter Monitor (PCM) for CPU energy consumption and the pyNVML package for GPU energy consumption [9].

5.2 Evaluation

Energy consumption optimization In all optimization scenarios reported, we ensure that the model achieves the same accuracy compared to the baseline over 3 runs. The energy consumption can be saved up to 97.2%. Energy consumption results are presented in Tbl. 2. Compared to FedAvg, FEDHW reduces energy consumption by 53.5% by tailoring E, BS, and F in each round, while for FedOpt, the reduction is 59.2%. Specifically, FEDHW-SA achieves a 60.6% reduction in energy consumption, while FEDHW-GA achieves a 59.2% reduction.

Table 2: Energy consumption optimization.

Model	FedAvg	SA	GA	FedOpt	SA	GA
CNN	1×	0.43×	0.98×	1×	0.23×	0.29×
ResNet18	1×	0.34×	0.22×	1×	0.61×	0.63×
M18	1×	0.69×	0.90×	1×	0.62×	0.52×
LSTM	1×	0.08×	0.07×	1×	0.08×	0.03×

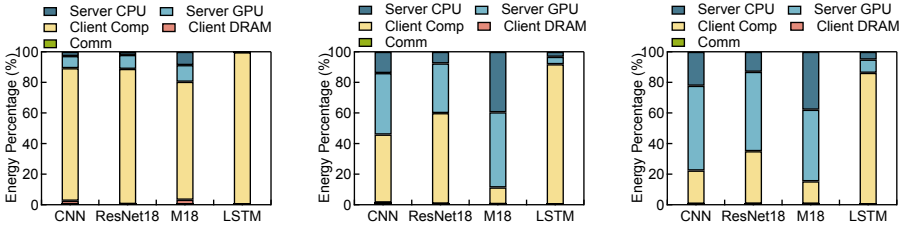
We observe that FEDHW-GA outperforms FEDHW-SA on larger datasets, as the GA optimizer tends to converge to optimal settings more quickly. Conversely, FEDHW-SA performs better on smaller datasets, as its optimization process results in more aggressive hyperparameters which means smaller E and BS, leading to faster model aggregation and lower energy consumption. Besides, different hyperparameter initialization settings have a large impact on the learning process. Finding the right parameter settings is important for energy consumption and latency in the federated learning training process.

Latency optimization The effectiveness of FEDHW optimizer extends to latency optimization, as demonstrated in Tbl. 3. The training latency can be reduced up to 96.0%. On average, FEDHW reduces training latency by 79.3% compared to the FedAvg baseline and 78.0% compared to FedOpt. Notably, FEDHW-GA shows a slight improvement over FEDHW-SA in latency optimization, both in terms of total latency and latency per training round.

Table 3: Latency optimization.

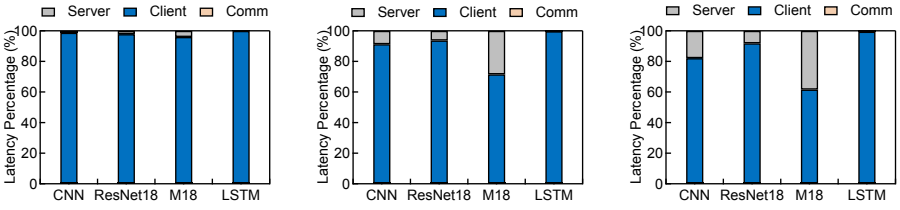
Model	FedAvg	SA	GA	FedOpt	SA	GA
CNN	1×	0.25×	0.64×	1×	0.23×	0.29×
ResNet18	1×	0.34×	0.28×	1×	0.70×	0.78×
M18	1×	0.79×	0.61×	1×	0.46×	0.15×
LSTM	1×	0.21×	0.17×	1×	0.05×	0.04×

5.3 Energy Model And Latency Model Breakdown



(a) FedAvg (b) FEDHW-SA (c) FEDHW-GA

Fig. 2: Energy Consumption Breakdown



(a) FedAvg (b) FEDHW-SA (c) FEDHW-GA

Fig. 3: Latency Consumption Breakdown

We break down energy and latency results and show them in Fig. 2 and Fig. 3. As we expect, in the FedAvg framework, the energy and latency consumed by clients take the highest proportion. The average energy consumed by clients is 63.0% and can be up to 75.6%. Local training latency takes over 96.0% on four cases. Communication between clients and the central server, although frequently happens, only consumes 0.07% and 0.66% energy and latency in the process.

Thus, our optimization approach is reasonable as tailoring E, BS, F every training round mainly saves client latency and energy. Specifically, we reduce client energy by 36.4% with the SA optimizer and 48.7% with the GA optimizer. The local training latency is reduced by 66.2% and 66.0%. After our optimization, the energy and latency proportion spent on the server significantly increase as we show in Fig. 2(b) and Fig. 3(b).

From our evaluation, we find that for the first several rounds of training, the optimizer tends to choose larger E and F as more knowledge can be learned in the early stages of model training and facilitates model aggregation. With the accuracy of the model gradually increasing, smaller E and F can result in better energy and latency while maintaining the final accuracy. Also, a larger epoch number does not generally mean higher energy consumption or latency, as a larger epoch number can enable local training with better results, which further translates to less training rounds and reduces total energy and latency. For BS, it's not always the case that the smaller the batch, the more consumption-efficient it is. And we should find the appropriate batch size for each dataset.

5.4 Ablation Study

FEDHW is ubiquitous to all kinds of hardware, as long as accurate energy and latency metrics are provided. We here show an ablation study where we change the underlying hardware from GPU to a neural processing unit (NPU), change the communication protocol from WiFi to LTE and convert the data distribution to not independent and identically distributed (Non-IID). We validate our optimizer still works.

GPU/NPU NPU is largely used in edge devices such as smart phones [11] and smart furniture [26, 39] for the extremely low power and high compute capability. We here use an NPU with a basic 32×32 systolic array and clocked at 1GHz. After synthesizing it with a 16nm technology, its power is at 1.157W and its peak throughput is 2.048 TOPS [52]. We use a cycle-accurate simulator SCALE Sim v2 [32, 33] to simulate the cycles of NPU on different workloads and calculate the energy consumption through $E = P \times \frac{cycles}{frequency}$.

We use the FedAvg algorithm and CIFAR-10 dataset as an example, FEDHW still works after changing GPU to NPU in the client platform. The energy consumption percentage of

client training drops from 53.8% to 15.2% and 14.5% of the total energy consumption with optimizers, which is similar to using the GPU on the client. The NPU only takes 1.3% energy consumption to get the same model performance compared to GPU, which is the reason we recommend using NPU on edge devices.

WiFi/LTE In our evaluation, we use WiFi as communication network. However, in real-world scenarios, usually, clients do not work in an environment with a WiFi network. We change the communication protocol to LTE and perform the evaluation. Results show that FEDHW framework saves 71.0% energy consumption with the SA optimizer and 76.0% energy consumption with the GA optimizer, similar to using a WiFi network.

Non-IID Non-IID datasets are better suited to real world application scenarios and tend to take more time to train the model compared to IID datasets. We change distribution of MNIST dataset in each client, making the variety of data in each client imcomplete but the amount of data consistent and train the CNN model to meet the same accuracy as the IID scenario[10]. Results show that FEDHW framework saves 74.6% energy consumption and 89.3% latency consumption compared to the FedAvg algorithm.

6 Conclusion

Federated learning has a wide range of application scenarios, especially today with the widespread use of 5G wireless communication technology and the increasing computing power of edge devices. It is one of the key technologies to capitalize on the data silos. This paper provides a new benchmarking of the federated learning process from the hardware perspective, and designs a framework FEDHW with two optimizers, SA and GA, to make the federated learning process more efficient in energy-constraint or latency constraint situations. Unlike other FL optimization frameworks, FEDHW can be applied to most FL algorithms. We believe that when the computing power of edge devices becomes more and more powerful, optimization frameworks that consider the hardware level can more fully utilize the characteristics of the hardware itself, making the federated learning process more efficient. Meanwhile, we will make more explorations on client hardware platform diversity and data distribution diversity. We hope to have more complete realistic benchmarking of the federated learning process from the hardware perspective in the future.

7 Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (Grant No. 62025404)

References

[1] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), page 483–485, New York, NY, USA, 1967. Association for Computing Machinery. ISBN 9781450378956. doi: 10.1145/1465482.1465560. URL https://doi.org/10.1145/1465482.1465560.

[2] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL https://pytorch.org/assets/pytorch2-2.pdf.

[3] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication, 2019.

[4] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. Very deep convolutional neural networks for raw waveforms. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 421–425. IEEE, 2017.

[5] Roy Friedman, Alex Kogan, and Yevgeny Krivolapov. On power and throughput tradeoffs of wifi and bluetooth in smartphones. *IEEE Transactions on Mobile Computing*, 12(7):1363–1376, 2012.

[6] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10112–10121, June 2022.

[7] Yongxin Guo, Xiaoying Tang, and Tao Lin. Fedbr: Improving federated learning on heterogeneous data via local learning bias reduction. In *International Conference on Machine Learning*, pages 12034–12054. PMLR, 2023.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Intel Performance Counter Monitor. Intel performance counter monitor. https://github.com/intel/pcm, Oct 2016. Accessed on: May 1, 2024.

[10] Ashwin R Jadhav. Federated-learning-pytorch. https://github.com/AshwinRJ/Federated-Learning-PyTorch, Nov 2018. Accessed on: May 1, 2024.

[11] Jun-Woo Jang, Sehwan Lee, Dongyoung Kim, Hyunsun Park, Ali Shafiee Ardestani, Yeongjae Choi, Channoh Kim, Yoojin Kim, Hyeongseok Yu, Hamzah Abdel-Aziz, et al. Sparsity-aware and re-configurable npu architecture for samsung flagship mobile soc. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 15–28. IEEE, 2021.

[12] Georgios A Kaissis, Marcus R Makowski, Daniel Rückert, and Rickmer F Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311, 2020.

[13] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[14] Young Geun Kim and Carole-Jean Wu. Autofl: Enabling heterogeneity-aware energy efficient federated learning. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 183–198, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385572. doi: 10.1145/3466752.3480129. URL https://doi.org/10.1145/3466752.3480129.

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[16] Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. Fedscale: Benchmarking model and system performance of federated learning at scale. In *International conference on machine learning*, pages 11814–11827. PMLR, 2022.

[17] 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.

[18] Sangyoon Lee and Dae-Hyun Choi. Federated reinforcement learning for energy management of multiple smart homes with distributed energy resources. *IEEE Transactions on Industrial Informatics*, 18(1):488–497, 2022. doi: 10.1109/TII.2020.3035451.

[19] David Lewis. Reuters-21578 Text Categorization Collection. UCI Machine Learning Repository, 1997. DOI: https://doi.org/10.24432/C52G6M.

[20] Liang Li, Dian Shi, Ronghui Hou, Hui Li, Miao Pan, and Zhu Han. To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pages 1–10, 2021. doi: 10.1109/INFOCOM42981.2021.9488839.

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

[22] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, May 2020. ISSN 1558-0792. doi: 10.1109/msp.2020.2975749. URL http://dx.doi.org/10.1109/MSP.2020.2975749.

[23] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[24] WANG Luping, WANG Wei, and LI Bo. Cmfl: Mitigating communication overhead for federated learning. In *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*, pages 954–964. IEEE, 2019.

[25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, April 2017.

[26] Massimo Merenda, Carlo Porcaro, and Demetrio Iero. Edge machine learning for ai-enabled iot devices: A review. *Sensors*, 20(9):2533, 2020.

[27] Xiaopeng Mo and Jie Xu. Energy-efficient federated edge learning with joint communication and computation design. *Journal of Communications and Information Networks*, 6(2):110–124, 2021. doi: 10.23919/JCIN.2021.9475121.

[28] NVIDIA Jetson TX2. Jetson tx2 module. https://developer.nvidia.com/embedded/jetson-tx2. Accessed on: May 1, 2024.

[29] Jiang. Qing-Yuan. Lstm-classification-pytorch. https://github.com/jiangqy/LSTM-Classification-pytorch?tab=readme-ov-file, Aug 2017. Accessed on: May 1, 2024.

[30] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[31] Riccardo Ricevuto. Federated-learning-with-resnet-50-on-cifar-10. https://github.com/ricevutoriccardo/Federated-Learning-with-ResNet-50-on-CIFAR-10, Feb 2023. Accessed on: May 1, 2024.

[32] Ananda Samajdar, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. Scale-sim: Systolic cnn accelerator simulator. *arXiv preprint arXiv:1811.02883*, 2018.

[33] Ananda Samajdar, Jan Moritz Joseph, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. A systematic methodology for characterizing scalability of dnn accelerators using scale-sim. In *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 58–68. IEEE, 2020.

[34] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

[35] 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, 2020. doi: 10.1109/ICC40277.2020.9149138.

[36] Lukas Steiner, Matthias Jung, Felipe S Prado, Kirill Bykov, and Norbert Wehn. Dramsys4. 0: a fast and cycle-accurate systemc/tlm-based dram simulator. In *Embedded*

Computer Systems: Architectures, Modeling, and Simulation: 20th International Conference, SAMOS 2020, Samos, Greece, July 5–9, 2020, Proceedings 20, pages 110–126. Springer, 2020.

[37] Lukas Steiner, Matthias Jung, Felipe S Prado, Kirill Bykov, and Norbert Wehn. Dramsys4. 0: An open-source simulation framework for in-depth dram analyses. *International Journal of Parallel Programming*, 50(2):217–242, 2022.

[38] Yuxuan Sun, Sheng Zhou, Zhisheng Niu, and Deniz Gündüz. Dynamic scheduling for over-the-air federated edge learning with energy constraints. *IEEE Journal on Selected Areas in Communications*, 40(1):227–242, 2022. doi: 10.1109/JSAC.2021.3126078.

[39] Tianxiang Tan and Guohong Cao. Efficient execution of deep neural networks on mobile devices with npu. In *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (Co-Located with CPS-IoT Week 2021)*, pages 283–298, 2021.

[40] Chunlin Tian, Li Li, Zhan Shi, Jun Wang, and ChengZhong Xu. Harmony: Heterogeneity-aware hierarchical management for federated learning system. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 631–645. IEEE, 2022.

[41] Hongda Wu and Ping Wang. Node selection toward faster convergence for federated learning on non-iid data. *IEEE Transactions on Network Science and Engineering*, 9 (5):3099–3111, 2022. doi: 10.1109/TNSE.2022.3146399.

[42] Jinjin Xu, Wenli Du, Yaochu Jin, Wangli He, and Ran Cheng. Ternary compression for communication-efficient federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(3):1162–1176, 2020.

[43] Zhaohui Yang, Mingzhe Chen, Walid Saad, Choong Seon Hong, and Mohammad Shikh-Bahaei. Energy efficient federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications*, 20(3):1935–1949, 2020.

[44] Zhaohui Yang, Mingzhe Chen, Walid Saad, Choong Seon Hong, and Mohammad Shikh-Bahaei. Energy efficient federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications*, 20(3):1935–1949, 2021. doi: 10.1109/TWC.2020.3037554.

[45] Qiying Yu, Yang Liu, Yimu Wang, Ke Xu, and Jingjing Liu. Multimodal federated learning via contrastive representation ensemble. *arXiv preprint arXiv:2302.08888*, 2023.

[46] Qunsong Zeng, Yuqing Du, Kaibin Huang, and Kin K. Leung. Energy-efficient radio resource allocation for federated edge learning. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2020. doi: 10.1109/ICCWorkshops49005.2020.9145118.

[47] Qunsong Zeng, Yuqing Du, Kaibin Huang, and Kin K. Leung. Energy-efficient resource management for federated edge learning with cpu-gpu heterogeneous computing. *IEEE Transactions on Wireless Communications*, 20(12):7947–7962, 2021. doi: 10.1109/TWC.2021.3088910.

[48] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[49] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10174–10183, June 2022.

[50] Peiying Zhang, Chao Wang, Chunxiao Jiang, and Zhu Han. Deep reinforcement learning assisted federated learning algorithm for data management of iiot. *IEEE Transactions on Industrial Informatics*, 17(12):8475–8484, 2021.

[51] Yuchen Zhao, Payam Barnaghi, and Hamed Haddadi. Multimodal federated learning on iot data. In *2022 IEEE/ACM Seventh International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 43–54. IEEE, 2022.

[52] Yuhao Zhu, Anand Samajdar, Matthew Mattina, and Paul Whatmough. Euphrates: Algorithm-soc co-design for low-power mobile continuous vision. *arXiv preprint arXiv:1803.11232*, 2018.

[53] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12878–12889. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/zhu21b.html.