EEFL: High-Speed Wireless Communications Inspired Energy Efficient Federated Learning over Mobile Devices

Rui Chen
rchen19@uh.edu
University of Houston
Houston, TX, USA

Qiyu Wan
wanzz061@gmail.com
University of Houston
Houston, TX, USA

Xinyue Zhang
xzhang48@kennesaw.edu
Kennesaw State University
Kennesaw, GA, USA

Xiaoqi Qin
xiaoqiqin@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Yanzhao Hou
houyanzhao@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Di Wang
di.wang@kaust.edu.sa
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia

Xin Fu
xfu8@central.uh.edu
University of Houston
Houston, TX, USA

Miao Pan
mpan2@uh.edu
University of Houston
Houston, TX, USA

ABSTRACT

Energy efficiency is essential for federated learning (FL) over mobile devices and its potential prosperous applications. Different from existing communication efficient FL research efforts, which regard communication energy consumption as the bottleneck, we have observed that with ever increasing wireless transmission speed (e.g., Wi-Fi 5 or 5G), the energy consumption of wireless communications for model updates in FL is significantly reduced and sometimes is smaller than that of local on-device training. Motivated by such observations, in this paper, we propose a high-speed wireless communications inspired energy efficient federated learning over mobile devices (EEFL), whose goal is to reduce the overall energy consumption (computing + communication). In particular, we design a novel energy-aware adaptive local update policy for mobile devices by jointly considering FL performance and energy saving of high-speed wireless transmissions. Furthermore, given the device's local update policy in each FL global round, we advance the dynamic voltage and frequency scaling (DVFS) strategy to minimize local training's energy consumption by keeping GPU and CPU working at appropriate frequencies without triggering thermal throttling. Extensive experimental results with various learning models, datasets, and wireless transmission environments demonstrate the proposed EEFL's superiority over the peer designs in terms of energy efficiency.

CCS CONCEPTS

• **Computing methodologies** → **Distributed artificial intelligence**; **Neural networks**.

KEYWORDS

Federated learning, Mobile devices, Energy efficiency, DVFS.

ACM Reference Format:
Rui Chen, Qiyu Wan, Xinyue Zhang, Xiaoqi Qin, Yanzhao Hou, Di Wang, Xin Fu, and Miao Pan. 2023. EEFL: High-Speed Wireless Communications Inspired Energy Efficient Federated Learning over Mobile Devices . In *The 21st Annual International Conference on Mobile Systems, Applications and Services (MobiSys '23), June 18–22, 2023, Helsinki, Finland.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3581791.3596865

1 INTRODUCTION

In recent years, we have witnessed the prosperity of federated learning (FL), whose disruptive idea is to train deep neural networks (DNNs) locally on multiple devices without data sharing, and aggregate global model on a FL server. With the development of hardware, more and more mobile devices (e.g., Google Pixel 4a, Galaxy Note20, iPad Pro, etc.) have on-device training capability, and FL over mobile devices fosters numerous promising applications in various domains such as cardiac event prediction in e-Healthcare, autonomous driving in smart transportation, physical hazard detection in smart home, smart farming and monitoring in agricultural and industrial IoT, etc. [29, 36, 53]. As we know, FL's training is energy hungry due to locally computing DNNs on big datasets and wirelessly communicating large-size model updates, while there have been very limited research efforts on energy efficient FL over mobile devices. Most existing works assume that mobile devices only participate in FL when they are plugged into wall-power, due to the iterative on-device training and model updates [9, 40, 48, 50, 55]. However, it would limit the practicality of FL in some mobile applications, resulting in longer model convergence time and degraded model accuracy [24, 30, 48]. Actually, energy efficiency is the key

Table 1: Comp. energy vs comm. energy in FL.

Model@Dataset (size)	Comm. (Tx. rate)		Comp. (Devices)
ResNet20@CIFAR10 (1.02MB)	0.6J (80Mbps)	3.8J (7Mbps)	9.4J when H=10 (NVIDIA XAVIER)
LSTM@CIFA10 (1.2MB)	0.63J (50Mbps)	4J (8Mbps)	4.8J when H=10 (NVIDIA TX2)
ResNet34@CIFAR10 (3.4MB)	1.7J (80Mbps)	19.6J (9Mbps)	74J when H=10 (NVIDIA TX2)
VGG16@CIFAR10 (50MB)	7.75J (80Mbps)	75J (8Mbps)	56J when H=10 (NVIDIA XAVIER)
VGG19@CIFAR10 (80MB)	4J (35Mbps)	317J (7Mbps)	80J when H=10 (NVIDIA XIAVER)

to enabling FL training anytime with good model quality and user experience and to deploying those intriguing FL applications on battery powered mobile devices in practice.

Instead of improving the energy efficiency of FL over mobile devices, most technicians and researchers regard wireless transmissions of local model updates as the bottleneck, and existing research efforts mainly focus on communication efficient FL designs. For example, Bernstein et al. in [4] quantized the model gradients into one bits, thus greatly reduce the communication costs. Rothchild et al. in [41] proposed structured and sketched updates to reduce the size of model updates. McMahan et al. [31] proposed FedAvg to reduce the communication frequency by allowing devices to perform multiple local training in one communication round. Haddadpour et al. in [17] developed an adaptive synchronization scheme to gradually increase the number of local updates as training proceeds, to further reduce the total number of communication rounds. However, reducing the number of communication rounds while incurring more local computing burden is not necessarily equivalent to reducing the total energy consumption of FL over mobile devices. Besides, ignoring the fast development of high-speed wireless transmissions recently, most works above rely on the assumption that wireless transmission rate is 4~8 Mbps [31], which is much slower than that of current Wi-Fi 5.

With the advance of wireless communication technologies (e.g., Wi-Fi 5, 5G or future 6G), the data transmission speed is ever increasing, e.g., the standardized 5G transmission rate is 1 Gbps on average [1] (real-world measured achievable uplink rate is around 32 Mbps), and the empirically tested Wi-Fi 5 transmission rates in our lab are within the range of 20-90 Mbps. Given high-speed wireless communications, we have two interesting observations: (i) The energy consumption of local model updates' wireless transmissions is significantly reduced compared to that of low-speed wireless communications if the transmission power of mobile devices is fixed, as shown in Table 1. (ii) Since both local training (i.e., computing) and model updates (i.e., communications) contribute to FL convergence, if the energy consumption of high-speed wireless communications is smaller than that of computing, we may need to "talk" (communication) more and "work" (computing) less, in order to reduce the overall energy consumption of FL over mobile devices.

Inspired by the observations above, in this paper, we propose an energy efficient federated learning design over mobile devices (EEFL), which considers the energy benefits of high-speed wireless communications during FL training, and aims to reduce the overall energy consumption of mobile devices in FL. Briefly, the proposed design includes two components: (i) an energy-aware adaptive local stochastic gradient descent (SGD) policy and (ii) a corresponding thermal-aware dynamic voltage and frequency scaling (DVFS) strategy for GPU and CPU. We basically answer two questions for the energy-aware adaptive local SGD policy. First, how to adaptively increase the number of local updates (i.e., H), which are aware of wireless transmission conditions and the energy saved by high-speed wireless communications. Second, when to halt increasing H aware of the energy consumption of FL over mobile devices. Given the device's local SGD policy in each FL global round, we leverage the predictable workloads of local training tasks to estimate the power, time and energy consumption, and further optimize the DVFS scheduling of GPU and CPU. The goal is to minimize local on-device training's energy consumption by adjusting the GPU and CPU to work at appropriate frequencies without triggering thermal throttling.

To demonstrate its effectiveness, we implemented and evaluated a full prototype of EEFL system consisting of FL aggregator (Lambda RTX 8000), mobile FL clients (NVIDIA Jetson TX2, Xavier, and Android smartphones) and wireless communications (Wi-Fi 5 and 5G) for model updates between them. We conducted extensive experiments with various FL learning models, datasets and wireless transmission rates. Our experimental results show that the proposed EEFL can provide smaller training losses and higher test accuracy with fewer communication rounds and much less energy consumption, compared with existing peer designs.

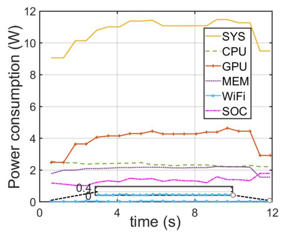
2 SYSTEM DESCRIPTION

2.1 FL over Mobile Devices

We consider a classical synchronized FL system consisting of one mobile edge server (e.g., base station or gNodeB) as the FL aggregator and N mobile devices as FL clients. Mobile devices attempt to jointly learn a global DNN model under the coordination of the mobile edge server. In particular, the mobile server first initializes and broadcasts a global model to the participating mobile devices. After receiving the global model, each mobile device trains its local model with its own training data examples for H local iterations in parallel. Once the local training is complete, the mobile devices will upload their model updates to the mobile edge server via wireless links. After that, the mobile edge server aggregates the local models to improve the global model. This procedure repeats until FL converges. Such iterative FL training process is very energy-consuming, with energy consumption mainly stemming from two parts: local on-device computing and wireless communications of the model update exchanges.

2.2 Computing Energy Consumption

The local computing energy consumption is linearly accumulated over local iterations, and the energy consumed per iteration can be calculated as the product of computing time and the mobile device's system power. For on-device computing, the major contributor to mobile device's system power is the computing runtime power. It is determined by voltage-frequency (VF) states and CPU/GPU utilization, which are related to specific learning tasks. The empirical local computing power consumption breakdown of FL over NVIDIA Jetson TX2 is shown in Fig. 1(a).


(a) Comp. power breakdown.

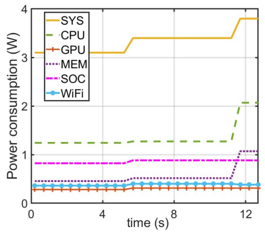
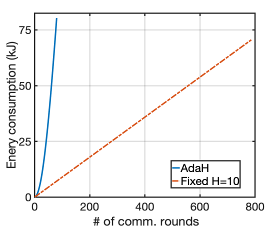

(b) Comm. power breakdown.

Figure 1: Power breakdown of FL over NVIDIA Jetson TX2s (VGG16@CIFAR10, Wi-Fi 5: 35Mbps).


(a) Energy consumption: AdaH [17] vs Fixed H.

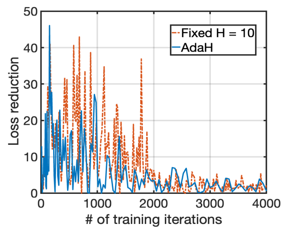
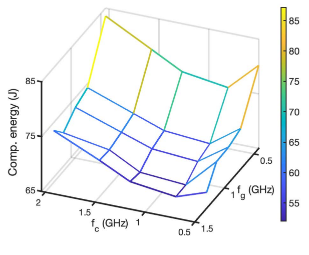

(b) Absolute training loss differentials of global models.

Figure 2: Empirical performance comparison: AdaH vs Fixed H (ResNet20@CIFAR10, Wi-Fi 5: 35Mbps).


(a) VGG16@CIFAR10, $H = 10$.

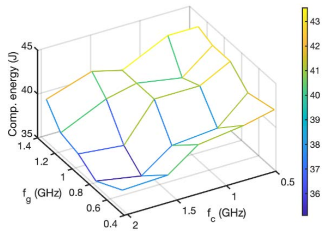

(b) LSTM@Shakespeare, $H = 10$.

Figure 3: The per iteration energy consumption of two learning tasks at different CPU and GPU frequencies.

2.3 Communication Energy Consumption

The overall communication energy consumed by each mobile device for wireless transmissions during the training process can be calculated as the product of the number of FL global rounds and the communication energy consumption per round. Here, the number of global rounds depends on FL convergence. Per round energy consumption, involving both uplink and downlink transmission, is mainly determined by transmission rate, the size of model updates to transmit, and device's system power consumption for wireless communications [42]. Note that for wireless communications, mobile device's system power consumption includes not only the transmission power consumed by antenna radiation (e.g., 23 dBm for sub-6GHz 5G [1]), but also power consumed by CPU, "idle" GPU, memory, RF amplification circuits, baseband signal processing, etc. [42]. The empirical communication power consumption decomposition of FL over NVIDIA Jetson TX2 is presented in Fig. 1(b).

3 OBSERVATION

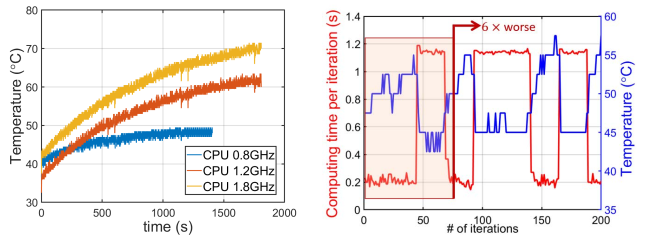
3.1 Hardware Limitation and Our Focus

In this paper, we aim to improve energy efficiency of FL over mobile devices by trading-off computing and communication energy consumption and reducing both. We envision that many possible recipes may help us approach to that goal, e.g., weight quantization [16, 56], model pruning [19], DVFS [23], help to reduce on-device training energy, and gradient compression [2, 8], adaptive local SGD policy [17], high-speed transmission, etc. help to reduce communication energy. Unfortunately, we cannot blend them all into our design due to two reasons: (i) Hardware limitation: the ARM based mobile devices, such as NVIDIA Jetson TX2, Xavier, and Android smartphones, that we have in the lab and use for experiments, cannot show the advantages of model pruning; TX2 doesn't support weight quantization, and Xavier cannot support training with weight quantization that is lower than 16-bit; and Xavier only accommodates Wi-Fi 5 for communications. (ii) Learning performance loss: either model compression or gradient compression brings learning performance loss to certain extent in terms of training loss and testing accuracy. Therefore, we will focus on adaptive local SGD policy and DVFS in this paper, and leave the testbed expansion and integration of compression techniques in future work as discussed in Sec. 7.

3.2 Inefficiency of Existing "AdaH" Policy

State-of-the-art adaptive local SGD policy [17] indicates that the number of global communication rounds in FL can be reduced by gradually increasing the number of local training iterations, which is referred as "AdaH" policy in this paper. Nevertheless, reducing the number of communication rounds while incurring more local computing burden is not necessary to reduce the total energy consumption of FL over mobile devices. We conducted empirical experiments to validate our projection. Here, we follow the AdaH policy in [17]: $H^r = H^0 \cdot (1 + \alpha r)$, where r is the global round index, H^r is the number of local training iterations in the r-th round, α is a coefficient, and H^0 is the initial value of the number of local iterations. We set $H^0 = 4$, and stop the FL training when its loss is lower than 0.3.

First, from the results in Fig. 2(a), we observe that compared with "Fixed H" policy, AdaH does reduce the communication rounds by around 85%, while AdaH consumes more energy than Fixed H in total. The potential reason is that current AdaH policy is *not aware of the energy reduced by high-speed wireless communications and its benefits* for FL training. Second, from the results in Figure 2(b), we observe that although the AdaH policy can help reduce the loss quickly with less variance at the early stages, its performance is similar to that of Fixed H policy at the late stages during FL training. That means always increasing H policy of AdaH brings very limited benefits to model convergence at late stages of FL training, while the energy consumption per round increases exponentially as shown in Fig. 2(a). Thus, *an energy-aware criterion to cease increasing the number of local training iterations* is in need.



(a) CPU temp.: (VGG16@CIFAR10). (b) Thermal throttling and delay.

Figure 4: Thermal issues of default DVFS.

3.3 Inefficiency of Default DVFS

Given a large H^r in the r-th round, processors (CPU & GPU) working at high frequencies help to speed up local training, but introduce thermal throttling issues. In many FL scenarios, certain devices are more frequently selected than the others[26]. It increases the likelihood of thermal-throttling for those devices. To avoid the overheating issue of devices, DVFS and thermal management [23] have been widely used to adjust the VF states of processors to reduce energy consumption. However, we observe that the default DVFS of mobile devices is learning task agnostic and has thermal issues.

First, we empirically measured energy consumption of Jetson TX2 with various local training tasks at different GPU and CPU frequencies. As results shown in Fig. 3, we observe that there exist optimal working frequencies for processors to minimize the energy consumption of different learning tasks (e.g., VGG16@CIFAR10, GPU: 1.88 GHz, CPU: 0.65 GHz and LSTM@Shakespeare, GPU: 0.82 GHz, CPU: 1.57 GHz), and different training tasks display different processor-related computing characteristics (e.g., VGG-19@CIFAR 10 is GPU intensive, and LSTM@Shakespeare is CPU intensive). Unfortunately, those characteristics cannot be captured by default DVFS governors of TX2/Xavier (i.e., *OnDemand*, *Interactive*, etc.). Among all default DVFS governors, *OnDemand* has the best performance in terms of energy efficiency, while it still consumes much more energy than the case of GPU and CPU working at the optimal frequencies, since *default DVFS governors are not learning task specific.*

Second, if processors work at high frequencies for local training, the temperature accumulates quickly. Once it triggers thermal throttling, the default DVFS drops the frequency of the processor to its minimum value. Such a frequency drop will significantly slow down the local training and incur much more energy consumption. As the example shown in Fig. 4, the computing time becomes 6 times slower when the CPU frequency drops from 1.5 GHz to 0.9 GHz. Thus, given H^r, for energy efficient local training, *a thermal-aware DVFS strategy to keep processors working at appropriate frequencies without violating thermal throttling* is in need.

4 EEFL DESIGN

4.1 EEFL Overview

To address the issues we observed above, we develop EEFL that includes: (i) an energy-aware AdaH policy, and (ii) a thermal-aware DVFS strategy. For the proposed energy-aware AdaH policy, we basically answer two questions: (1) How to develop an adaptive H^r policy aware of wireless transmission rates between mobile edge server and mobile devices? (2) How to design an energy-aware policy to stop increasing H^r? Given the updated H^r, the thermal-aware DVFS design allows mobile devices to customize their DVFS strategies specific to learning tasks, keep processors working at optimal frequencies without triggering thermal throttling and minimize the energy consumption of local on-device training. The sketch of the EEFL design is shown in Fig. 5.

4.2 Energy-Aware AdaH Policy

4.2.1 *Wireless communications aware AdaH.*

As explained in Sec. 3.2, to improve the energy efficiency of FL over mobile devices, it is not good enough to naively increase H over rounds, and the energy saved by high-speed wireless communications should be considered for the AdaH policy design. Since both local on-device training and local model updates contribute to FL convergence, our idea behind the wireless communications aware AdaH policy is intuitive: trade-off communications and computing, i.e., when the wireless transmission rates between mobile devices and mobile edge server are high, we "talk" more and "work" less; when the wireless transmission rates between mobile devices and mobile edge server are low, we "work" more and "talk" less.

Following that intuition, we ameliorate the AdaH policy in [17] and formulate the wireless communications aware AdaH policy in EEFL as follows.

$$H_i^r = \left\lceil H^0 + \sum_{j=1}^{r} \alpha(s_i^j) \bigtriangleup H \right\rceil, \tag{1}$$

where H_i^r is the number of local training iterations for mobile device i in the r-th round, H^0 is the initial value of the number of local iterations, $\triangle H \geq 0$ is the increment unit, and $\alpha(\cdot)$ is a non-negative decreasing function of wireless transmission rates. Here, we denote s_i^r as the average transmission data rate between mobile device i in the r-th round.

A simple interpretation of the upgraded AdaH policy defined in Eqn. (1) is: during the r-th round, if the wireless communications between mobile device i and mobile edge server is fast (i.e., large s_i^r), the increment of H_i^r will be small, and device i will compute less and communicate more; otherwise, if the wireless communications between i and mobile edge server is slow (i.e., small s_i^r), the increment of H_i^r will be large, and device i will compute more and communicate less. In this way, the AdaH policy in Eqn. (1) benefits not only from increasing H^r as illustrated in [17] but also from awareness of wireless transmission rates and computing-communication trade-off, which eventually helps to improve the energy efficiency of FL over mobile devices.

4.2.2 *An energy-aware criterion to stop H_i^r's increasing.*

Following the second observation in Sec. 3.2, the benefits of increasing H_i^r in Eqn.(1) diminish in the late stages of FL training, when it is close to convergence. However, on-device computing energy consumption increases exponentially. An energy-aware criterion to stop increasing H_i^r is needed.

Here, we propose a simple yet effective "STOP" metric by evaluating the ratio of H_i^{r-1}'s contributions to global training and its corresponding computing energy cost:

$$\Lambda_i^r = |\ell^{r-2} - \ell^{r-1}| / E_{i,cp}^{r-1}, \tag{2}$$

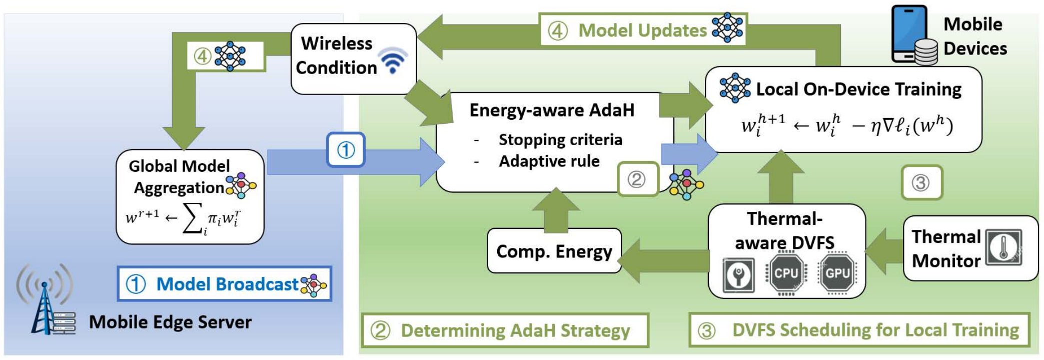


Figure 5: The design sketch of energy efficient federated learning over mobile devices (EEFL).

where ℓ^{r-1} and ℓ^{r-2} are the training loss of global model in the $(r-1)$-th round and the $(r-2)$-th round, respectively, and $E_{i,cp}^{r-1}$ denotes the computing energy of mobile device i in the $(r-1)$-th round. Here, we employ model performance analysis to decide whether H_i^r should be increased or not. If increasing H_i^{r-1} in the $(r-1)$-th round contributes a lot to global training by increasing limited computing energy consumption, i.e., Λ_i^r is large, we will continue increasing H_i^r. If increasing H_i^{r-1} in the $(r-1)$-th round has limited contributions to global training by increasing a lot of computing energy consumption, i.e., Λ_i^r is small, we will stop increasing H_i^r, and set $H_i^r = H_i^{r-1}$.

Generally speaking, Λ_i measures how much training loss is reduced per Joule for device i. At the beginning stages of FL training, Λ_i is large and device i would increase H_i, since it helps to reduce the global model loss. As FL training proceeds, Λ_i becomes small as the change of global model loss becomes small, and the computing energy consumption increases. Once Λ_i^r for the r-th round is smaller than a certain threshold Λ_{th}, it suggests by increasing H_i^r, device i may have limited contributions to the global model for potential huge energy consumption. At this point, device i should stop increasing H_i^r and continue local training with $H_i^r = H_i^{r-1}$.

4.3 Thermal-Aware DVFS Strategy

Given the updated H^r policy, we then propose a thermal-aware DVFS design that can be configured by the mobile device itself. To address the inefficiency of default DVFS mentioned in Sec. 3.3, we first build up on-device training task (DNN@dataset) specific computing energy modeling of processors (CPU and GPU), estimate model parameters for practical usage and validate them with empirical studies. Then, based on the computing energy modeling of processors, we formulate an optimization problem to jointly adjust the VF states of CPU and GPU to minimize the energy consumption of local on-device training without triggering thermal throttling. Note that since each mobile device determines its DVFS scheduling independently, we omit the mobile device index notation (i.e., the subscript i) and describe the DVFS scheduling without differentiating them in this subsection.

4.3.1 *Learning Task Specific Computing Energy Modeling.*

The CPU/GPU in most mobile devices can work across a set of discrete VF states, e.g., for CPU, $\mathcal{F}_c = \{f_{c,j}\}_{j=1}^J$ and for GPU, $\mathcal{F}_g = \{f_{g,k}\}_{k=1}^K$. To obtain the computing energy over H^r iterations, we first model the processors' computing runtime power and computing time for one single iteration. Then, we employ those models to calculate processors' computing energy for one iteration by letting processors work at $\boldsymbol{f}_{j,k} \in \mathcal{F}_c \times \mathcal{F}_g$, where $\boldsymbol{f}_{j,k} = (f_{c,j}, f_{g,k})$ and the total design space is $C = |\mathcal{F}_c \times \mathcal{F}_g|$. Then, we sequentially add up the processors' computing energy consumption from the 1st iteration to the H^r-th iteration (i.e., processors working at a sequence of frequency states) to estimate the energy consumption of processors for performing H^r local iterations.

Computing power model: For a local training iteration, the average computing power of CPU and GPU at state $\boldsymbol{f}_{j,k}$ can be modeled as [57],

$$P_c(f_{c,j}) = (V_{c,j})^2 f_{c,j} \tau_c, \text{ and } P_g(f_{g,k}) = (V_{g,k})^2 f_{g,k} \tau_g, \quad (3)$$

where $V_{c,j}$ and $V_{g,k}$ are the voltage values of CPU and GPU, respectively, τ_c and τ_g indicate the average resource utilization ratios of CPU and GPU, respectively. $\boldsymbol{V}_{j,k} = (V_{c,j}, V_{g,k})$ is jointly determined by $\boldsymbol{f}_{j,k}$ and hardware configurations [10]. Here, the computing characteristics of a learning tasks can be reflected by τ_c and τ_g. Given $\boldsymbol{f}_{j,k}$, the average power consumption within one local iteration is computed as

$$P(\boldsymbol{f}_{j,k}) = P_c(f_{c,j}) + P_g(f_{g,k}) + P_0, \quad (4)$$

where P_0 denotes the the summation of power consumption unrelated to the CPU and GPU VF scaling.

Computing time estimation: Since CPU's and GPU's workloads to train a specific learning task are almost evenly distributed across iterations, it helps us find a simple yet effective way to estimate the computing time at different VF states. Assume that the computing time $D_{cp}(\boldsymbol{f}_{j,k})$ is inversely proportional to the frequencies of CPU and GPU $\boldsymbol{f}_{j,k}$ [11]. Since the CPU and GPU workloads almost remain unchanged across iterations, we can leverage the computing time measured at the maximum frequency setting $(f_{c,\max}, f_{g,\max})$ and estimate the computing time for one local iteration at $\boldsymbol{f}_{j,k}$ [11]:

$$\hat{D}_{cp}(\boldsymbol{f}_{j,k}) = (1-\rho) \cdot D_{\min} \frac{f_{c,\max}}{f_{c,j}} + \rho \cdot D_{\min} \frac{f_{g,\max}}{f_{g,k}}, \quad (5)$$

where ρ is the ratio of computing time on GPU, $(1-\rho)$ is the ratio of computing time on CPU, and $D_{\min}$ is the minimum time when the mobile device operating at the maximum frequency of both CPU and GPU. The value of ρ varies over different learning tasks,

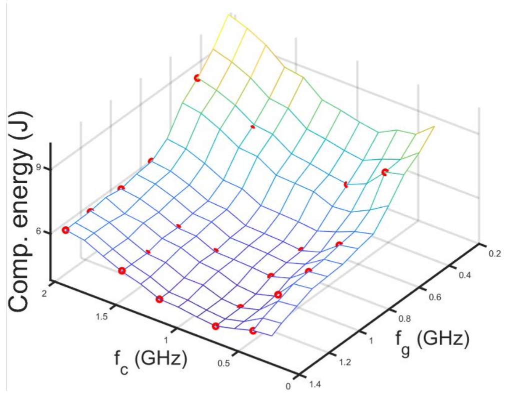


Figure 6: Validation of parameter estimation and computing energy model (VGG@CIFAR10, NVIDIA Jetson TX2); the estimated parameters are $\hat{\rho} = 0.94$, $\hat{\tau_g} = 0.742$, $\hat{\tau_c} = 0.236$, $\hat{P_0} = 246\,mW$, and $D_{\min} = 0.58$.

and once the learning task is fixed, ρ is fixed, which implies that ρ captures the computing characteristics of the specific learning task.

Computing energy model: The computing energy is the product of the average computing power and the computing time in h-th iteration at r-th round, which is

$$E_{cp}^{r,h}(\boldsymbol{f}_{j,k}) = D_{cp}^{r,h}(\boldsymbol{f}_{j,k}) \cdot P^{r,h}(\boldsymbol{f}_{j,k}). \tag{6}$$

The total computing energy at communication round r is the sum of the energy consumed for H^r local iterations, i.e., $E_{cp}^r = \sum_h E_{cp}^{r,h}(\boldsymbol{f}_{j,k})$.

Thermal model: With the computing energy modeling above, we can model how the processors' temperature changes when performing one local training iteration $h \in \{1, 2, \cdots, H^r\}$ at the state $\boldsymbol{f}_{j,k}$. Here, we consider the following discretized version of a lumped RC thermal model [27]:

$$T^{r,h+1} = T^{r,h} + \frac{D_{cp}^{r,h}(\boldsymbol{f}_{j,k})}{R_{th}C_{th}}(R_{th}P^{r,h}(\boldsymbol{f}_{j,k}) - T^{r,h}), \tag{7}$$

where $T^{r,h}$ is the temperature at the beginning of local iteration h, and R_{th} and C_{th} are the device-specific thermal resistance and capacitance, respectively, e.g., default R_{th} and C_{th} of the NVIDIA Jetson TX2 [3, 38] are 2 $°C/W$ and 0.9 $J/°C$, respectively.

4.3.2 Parameter Estimation and Model Validation.

For the computing energy model in Eqn. (6), we still have five unknown parameters, i.e., $D_{\min}$, ρ, τ_c, τ_g, and P_0. $D_{\min}$ can directly be measured by setting the CPU and GPU to work at their maximum frequencies. To estimate the rest of parameters, we can randomly pick two different states, $\boldsymbol{f}_{j,k}$ and $\boldsymbol{f}_{p,q}$ ($\boldsymbol{f}_{j,k}, \boldsymbol{f}_{p,q} \in \mathcal{F}_c \times \mathcal{F}_g$), conduct a small number of local training iterations ($\leq$ 5), and record the power consumption and computing time. Based on the recorded information, ρ can be estimated as $\hat{\rho} = (\frac{D(f_k)}{D_{\min}} - \frac{f_{c,\max}}{f_{c,j}})/(\frac{f_{g,\max}}{f_{g,k}} - \frac{f_{c,\max}}{f_{c,j}})$. We can also estimate τ_g by solving a three-variable linear equation in Eqn. (3) and Eqn. (4) and have

$$\hat{\tau_g} = \frac{\left(P(\boldsymbol{f}_{j,k}) + P(\boldsymbol{f}_{\max}) - \frac{P(\boldsymbol{f}_{j,k})+P(\boldsymbol{f}_{p,q})}{(V_{c,j})^2 f_{c,j}+(V_{c,p})^2 f_{c,p}}\right)}{\left((V_{g,k})^2 f_{g,k} + (V_{c,\max})^2 f_{c,\max} - \frac{(V_{g,k})^2 f_{g,k}+(V_{c,q})^2 f_{c,q}}{(V_{c,j})^2 f_{c,j}+(V_{c,\max})^2 f_{c,\max}}\right)}.$$

Similarly, we can estimate τ_c and P_0. Then, we can replace the unknown parameters in Eqns. (3), (4) and (5) with the estimated values of those parameters.

We have conducted experiments to validate the proposed parameter estimation methods and computing energy model. Briefly, we have measured the power consumption of Jetson TX2 using built-in sensors, recorded its computing time in Python and calculated the real-world energy consumption. We compare it with the estimated energy in Eqn.(6), and show the results in Fig. 6. The surface is the estimated energy, and red dots are the energy consumption measurements at different values of $\boldsymbol{f}_{j,k}$. We have also done similar comparisons using different learning tasks, and the results show that the estimation error ratio is small ($\leq$5%).

4.3.3 Thermal-Aware DVFS Scheduling.

With the fulfilled computing energy and thermal models above, to address the thermal issues of default DVFS recognized in Sec. 3.3, we develop a thermal-aware DVFS scheduling for mobile devices. The goal is to minimize the on-device training energy consumption without violating the delay and thermal constraints during every FL global round. The thermal-aware DVFS optimization is as follows.

$$\min_{f_{c,j}, f_{g,k}} \quad E_{cp}^r = \sum_{h=1}^{H^r} E_{cp}^{r,h}(\boldsymbol{f}_{j,k}) \tag{8a}$$

$$\text{s.t.,} \quad T^{r,h+1} = T^{r,h} + \frac{D_{cp}^{r,h}(\boldsymbol{f}_{j,k})}{R_{th}C_{th}}(R_{th}P^{r,h}(\boldsymbol{f}_{j,k}) - T^{r,h}), \tag{8b}$$

$$T^{r,h} \leq T_{\text{th}}, \quad \forall h \in \{1, \cdots, H^r\} \tag{8c}$$

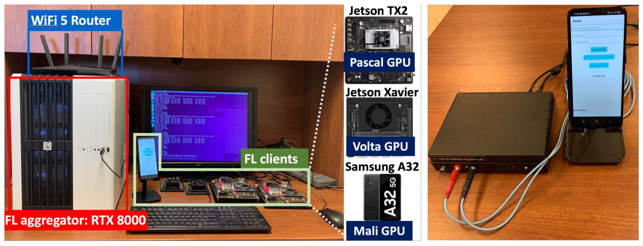
$$\sum_{h=1}^{H^r} D_{cp}^{r,h}(\boldsymbol{f}_{j,k}) \leq D_{\text{th}}, \tag{8d}$$

where $T^{r,h}$ denotes the temperature of local iteration h in r-th round. Constraint (8c) ensures zero thermal throttling [23], i.e., the temperature will not exceed the threshold T_{th} to avoid triggering thermal throttling. Constraint (8d) guarantees mobile devices complete their local training within a pre-set deadline D_{th}, which is determined by mobile edge server.

The DVFS scheduling in (8) can be mapped to the multi-choice knapsack problem (MCKP). The thermal and delay constraints can be regarded as the capacities of the knapsack, and the goal is to fill in items (i.e., CPU and GPU frequencies), so that the total costs (i.e., energy) of the objects in the knapsack is minimized. We exploit dynamic programming [12] to solve this thermal-aware DVFS optimization. The solution has a complexity of $O(CH^r T_{th} D_{th})$, and the solution searching space can be further reduced by discarding those VF states whose steady-state temperature $T_s(\boldsymbol{f}_{j,k}) = R_{th}P^{r,h}(\boldsymbol{f}_{j,k})$ is higher than T_{th}.

5 IMPLEMENTATION & EVALUATION SETUP

In this section, we first describe the implementation of the EEFL system (Sec. 5.1), then introduce two FL tasks, four local DNN



(a) (b)

Figure 7: EEFL testbed in the lab: (a) FL testbed configuration; (b) Samsung A32's energy consumption measured by the Monsoon Power Monitor [35].

models and three datasets used for experiments (Sec. 5.2), and last present several peer FL designs employed for performance comparison (Sec. 5.3).

5.1 EEFL Prototype

The EEFL is implemented on testbed. The testbed, as illustrated in Fig 7(a), consists of an FL aggregator and 15 FL clients. On FL aggregator side, we use a NVIDIA RTX 8000 with 48GB memory. On FL client side, we consider three types of mobile devices: (1) NVIDIA Jetson TX2 with Jetpack 4.4, which has an NVIDIA Pascal GPU, 6-core ARM CPU, 8GB DDR4 memory; (2) NVIDIA Jetson Xavier with Jetpack 4.6, which has a Volta GPU and 8-core ARM CPU, 8GB DDR4 memory; and (3) Samsung A32 smartphone with a Mali-G53 MP3 GPU and Octa-core ARM CPU, 8GB RAM. For the communications between FL aggregator and FL clients, we follow the WebSocket [15] communication protocol and employ Wi-Fi 5 for wireless transmissions. The command line tools *nmcli* and wondershaper are used for reporting network status and controlling wireless transmission rates.

EEFL is implemented by building on top of FLOWER [5]. In total, we changed and added more than 800 lines of codes to FLOWER in both the FL aggregator and the FL client sides, in order to support the proposed energy-aware adaptive local SGD policy and thermal-aware DVFS scheduling. In particular, EEFL allows the mobile devices to customize the local SGD policy and DVFS scheduling. On the client side, given the newly-received global model and their computing energy consumption from the last round, the FL client can calculate its Λ_i. To estimate s_i^r, we let Jetson devices apply network analysis toolbox, bmon [39], and android smartphones utilize Network Monitor toollbox in the Android kernal. At the r-th communication round, they can determine the adaptive H^r policy based on the estimated s_i^r. Here, we set α function as a decreasing linear function[1]. We select the value of stopping threshold Λ among $\{10^{-4}, 5\times10^{-4}, 10^{-3}\}$ for the best testing accuracy. Then, to optimize local computing energy, the FL clients dynamically adjust CPU-GPU VF states by setting *Userspace* governor in Linux. Note that the MCKP problem is solved and coded in Python, and implemented at the client side. Given the estimated transmission rate, FL clients can solve MCKP problem while they wait for the new global model. The computational time of MCKP is less than the waiting time (see details in Sec. 6.5) and the solutions can be stored in a local look-up table. Hence, the overall computational overhead is very small.

To monitor and measure the energy consumption, Jetson devices are programmed with Python 3.6 to monitor its computing and Wi-Fi power consumption with the INA3221 module, temperatures, and frequencies of CPU and GPU with built-in sensors, which are accessible through sysfs [34] in the Linux kernel. For Android smartphones, we employ the Monsoon Power Monitor [35] to measure their power consumption during the FL process. The temperature and CPU/GPU frequencies can also be collected via sysfs in the Android kernel. We set the peak temperature $T_{th} = 60\,^{\circ}C$ to protect devices from overheating damages [27].

5.2 Models and Datasets

Our experiments consist of three FL tasks: image classification, next character/word prediction, and human activity recognition (HAR).

For image classification FL task, we use two datasets: (1) MNIST dataset [13] with total 60,000 training images. We train with 2-layer CNN. (2) CIFAR10 dataset [25] with 50,000 training images. We train VGG16 [46] and ResNet20 [18] to classify the image. We denote λ as the non-i.i.d levels, where $\lambda = 1$ indicates that data on each device only belong to one label, $\lambda = 0.75$ indicates that 75% of the data belong to one label and the remaining 25% data belong to other labels, $\lambda = 0.25$ indicates that 25% of the data belong to one label and the remaining 75% data belong to other labels, and $\lambda = 0$ represents i.i.d data setting.

For the next character prediction FL task, we use Shakespeare dataset [6], which is built from *The Complete Works of William Shakespeare*. Specifically, the Shakespeare dataset contains 845,231 samples separated into 139 users. The Shakespeare dataset is naturally unbalanced and non-i.i.d. For the next word prediction FL task, we use *Reddit*, which contains redditposts with 32,680 samples [6]. We use two-layer LSTM [20] to train the next character/word prediction task.

For HAR task, we use UCI-HAR dataset that contains six types of activity data from 30 users with 10,299 samples. We use 2-layer CNN in [45] to train it.

5.3 Peer FL Designs for Comparison

We compare the EEFL with the following peer FL designs under different FL learning tasks, DNN models and datasets.

FedAvg [31]: FL aggregator considers a fixed local SGD policy and the FL clients utilize default DVFS governor for energy and thermal management.

LUPA [17]: FL aggregator considers a adaptive local SGD policy during FL training and the adaptive rule is defined as $H^r = H^0 \cdot (1 + \alpha r)$. Similar to the *FedAvg*, the FL clients utilize default DVFS governor for energy and thermal management.

E-AdaH: FL aggregator applies the proposed energy-aware local SGD policy, and the adaptive rules are illustrated in Sec. 4.2. The FL clients utilize default DVFS governor for energy and thermal management.

FEDL [14]: FL aggregator considers a fixed local SGD policy during FL training and optimizes the operating CPU frequencies of FL clients to minimize the energy consumption under delay constraints without considering thermal effects.

[1]More detailed discussions are provided in Appendix B.

SparFL [28]: FL aggregator uses the fixed local SGD policy and gradient sparsification to reduce energy consumption. The optimal gradient sparsification and H are derived based on FL convergence bounds.

6 EVALUATION RESULTS & ANALYSIS

In this section, we present the experimental evaluation of EEFL and performance comparison with its peer FL designs, aiming to demonstrate EEFL's effectiveness and superiority.

6.1 FL Performance & Energy Efficiency

6.1.1 Performance comparison with FedAvg, LUPA & E-AdaH. Figure 8 shows the comparison of different FL schemes in terms of FL performance and corresponding energy consumption, i.e., Figs. 8(a)-8(d): training loss vs energy consumption, and Figs. 8(e)-8(h): testing accuracy vs energy consumption, when FL converges. It is obvious that both *EEFL* and *E-AdaH* have better performance than *LUPA* and *FedAvg*. Taking training loss for example, when FL converges, *EEFL* saves around 51%∼72% energy for the same training loss across different FL tasks, compared with *LUPA*, and reduces 28%∼58% compared with *FedAvg*. Similarly, *E-AdaH* reduces 32%∼71% energy consumption for the same training loss, compared with *LUPA* and reduces 18%∼52%, compared with *FedAvg*. The same trends can be observed for testing accuracy in Fig. 8(e)-8(h). It demonstrates the effectiveness of the proposed energy-aware AdaH policy. To effectively reduce energy consumption, both *EEFL* and *E-AdaH* are aware of the energy saving contributions of high-speed wireless communications, and understand the importance of "FL performance improvement per Joule" for mobile devices.

Besides, the proposed *EEFL* outperforms *E-AdaH*. Taking training loss for example, *EEFL* and *E-AdaH* have similar performance for ResNet20@CIFAR10 case (*EEFL*: 36,649 J vs *E-AdaH*: 36,777 J) in Fig. 8(b). For complex model in image tasks (e.g., VGG@CIFAR10), *EEFL* can reduce 28% energy consumption compared with *E-AdaH*. For the language task, the proposed *EEFL* can save 8%∼24% energy than *E-AdaH*. The reasons for *EEFL*'s superiority are: (1) the learning specific DVFS in *EEFL* is more energy efficient than the default DVFS in *E-AdaH*, and (2) for a complex task to train a large model (e.g., VGG16@CIFAR10), the thermal-aware DVFS scheduling of *EEFL* can avoid performance degradation due to CPU's and GPU's overheating and frequency dropping.

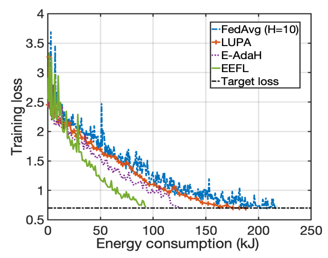
6.1.2 Performance comparison with FEDL & SparFL.
Furthermore, we compare *EEFL* with two most related prior designs, *FEDL* and *SparFL*, both of which aim to minimize the energy consumption of FL over mobile devices as well. The results are shown in Table 2. Compared with *FEDL*, the proposed *EEFL* can save around 40%∼50% energy while increasing ∼1.2% testing accuracy on average for image tasks; reduce energy consumption by 18%∼33% and improves testing accuracy by ∼0.1% testing accuracy on average for the language task; and save energy consumption by 37% and improves testing accuracy by 0.13% for the HAR task. The reason behind is that *FEDL* only optimizes the CPU working frequencies, but ignores the GPU frequencies and thermal issues. Besides, the fixed local SGD policy in *FEDL* is not aware of the computing-communication energy trade-off, or its potential benefits of improving the energy efficiency of mobile devices during FL training. Compared to *SparFL*, we find that the proposed *EEFL* can lower energy consumption by 23%∼28.1% while improving testing accuracy ∼ 0.2% on average for the image tasks; and it saves 13%∼31% energy with 0.05%∼0.9% testing accuracy improvement for the language task; and it reduces energy consumption by 31% and improves testing accuracy by 1.2% for the HAR task. The potential reason is three-folder: although gradient sparsification helps to reduce the size of local model updates and thus reduce the energy consumption, (1) the theoretical FL converge bound used in [28] may be not that tight, (2) the fixed local SGD policy may slow down the FL convergence in practice, and (3) no thermal-aware DVFS is used in *SparFL*.

6.2 Impacts of Wireless Transmission Rates

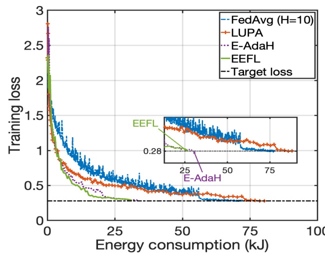
First, we exploit wondershaper [21] to set up Wi-Fi 5 transmission rates. Here, we only consider two types of mobile devices: Jetson Xavier and Jetson TX2. Fig. 9 presents the training energy with varying wireless transmission rates. The results indicate that the FL training energy consumption decreases as the wireless transmission rate increases. Moreover, under all those transmission rates, the proposed *EEFL* scheme outperforms the other FL designs due to its awareness of the computing-communication energy trade-off. For example, for "Wi-Fi 56 Mbps", *EEFL* reduces 55%, 61% and 21% energy consumption, compared with *LUPA*, *FedAvg* and *E-AdaH* in VGG16@CIFAR10, respectively. We also find that different transmission rates yield different workload allocation between "working" and "talking" to achieve FL convergence at the least energy consumption in *EEFL*. From the monitored H value of Jetson TX2 over *EEFL* rounds in Figs. 9(b) and 9(d), we find that when transmission rate is low, mobile devices "work" more (i.e., large H) and "talk" less; when transmission rate is high, mobile devices "work" less (i.e., small H) and "talk" more. For example, in VGG16@CIFAR10 case, when Wi-Fi speed is 35 Mbps, H stops increasing at around 65 for "Wi-Fi 35 Mbps"; by contrast H stops increasing at around 23 for "Wi-Fi 80 Mbps".

Then, we compare EEFL with other schemes under more complex wireless transmission scenarios, which have three types of mobile devices (the aforementioned two types of Jetson devices and Android smartphones) under the following two settings: (1) Jetson devices are connected via Wi-Fi 5, and Android smartphones are connected via 5G networks in an indoor environment (i.e., "Mix-Indoor"), and (2) Jetson devices are connected via Wi-Fi 5, and Android smartphones are connected via 5G networks in an outdoor environment (i.e., "Mix-Outdoor"). We measure and record the logs of uplink transmission rates of Wi-Fi 5 and 5G, as shown in Fig. 9(e). Here, we conduct the experiments of 15 FL clients. There are 5 Xaivers and 5 TX2s communicating with the server via Wi-Fi 5, and 5 Android smartphones via 5G.

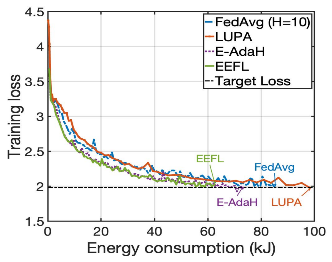
Obviously, Android smartphones have higher 5G transmission rates outdoors than indoors, while other devices are experiencing higher Wi-Fi 5 transmission rate indoors than outdoors. Correspondingly, for CNN@MNIST, as shown in Fig. 9(g), we can observe that the Android smartphones "work" more and the Jetson devices "work" less in the outdoor/indoor scenario. Figure 9 shows the total energy consumption of different FL schemes under "Mix-Indoor" and "Mix-Outdoor" scenarios. The proposed *EEFL* scheme



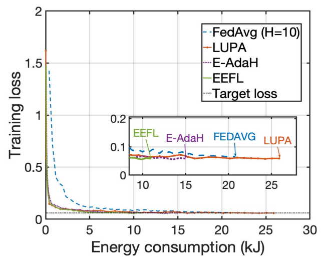
(a) VGG16@CIFAR10.



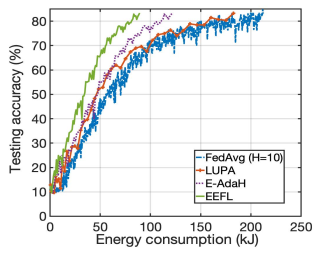
(b) ResNet20@CIFAR10



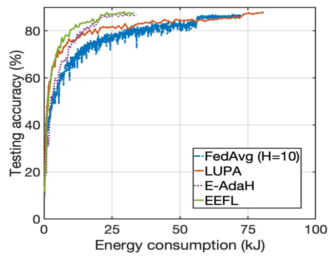
(c) LSTM@Shakespeare



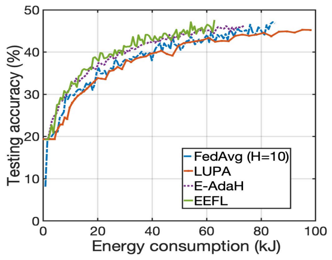
(d) CNN@MNIST.



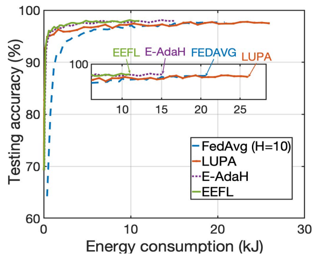
(e) VGG16@CIFAR10.



(f) ResNet20@CIFAR10



(g) LSTM@Shakespeare



(h) CNN@MNIST.

Figure 8: Learning performance and energy consumption under different FL tasks, local DNN models and datasets.

Table 2: Energy efficiency (EE) and accuracy on six datasets with the real-world user data.

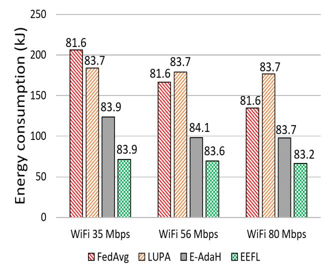
Task	CV						NLP				HAR	
Dataset	CNN@MNIST		ResNet20@CIFAR10		VGG16@CIFAR10		LSTM@Shakespeare		LSTM@Reddit		CNN@HAR	
Methods	EE	Acc.	EE	Acc.	EE	Acc.	EE	Acc.	EE	Acc.	EE	Acc.
FedAvg	1x	98.37	1x	89.41	1x	81.64	1x	47.14	1x	12.75	1x	82.15
LUPA	0.81x	**98.60**	0.90x	**90.12**	1.12x	83.77	0.80x	47.01	0.48x	12.80	0.82x	82.28
FEDL	1.17x	98.33	1.15x	87.32	1.23x	83.71	1.14x	46.83	1.21x	12.76	1.14x	82.13
SparFL	1.31x	98.30	1.76x	89.25	1.66x	81.48	1.29x	46.46	1.28x	12.71	1.25x	81.18
E-AdaH	1.40x	98.41	2.12x	89.82	1.67x	**83.92**	1.21x	**47.34**	1.80x	12.82	1.50x	**82.32**
EEFL	**1.79x**	98.41	**2.31x**	89.67	**2.36x**	83.91	**1.64x**	47.22	**1.89x**	**12.84**	**1.82x**	82.31

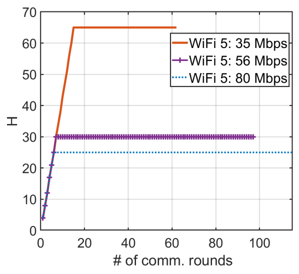
can always obtain high model accuracy with the minimum training energy consumption.

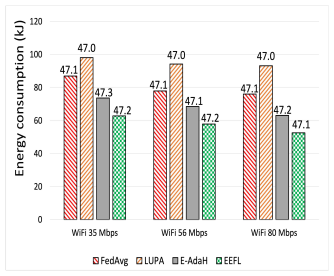
The findings above are consistent with our computing-communication energy trade-off analysis and design philosophy of energy-aware AdaH policy in *EEFL*.

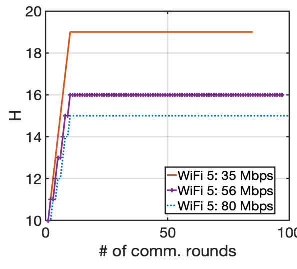
6.3 Advantages of Thermal-Aware DVFS

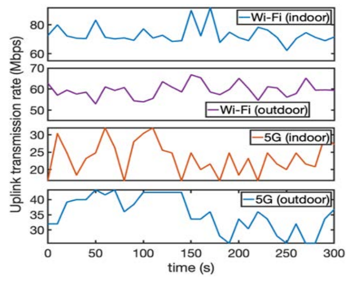
In Fig. 10(a), we provide the CPU temperature profiles of Jetson TX2 over time, when performing local training with VGG16@CIFAR10. We zoom in the temperature turning point from 145 s to 215 s, and compare the CPU frequency changes between default DVFS and the thermal-aware DVFS of the proposed *EEFL* in Fig. 10(b). From the results, we observe that after reaching peak temperature $60°C$, the CPU temperature of TX2 with default DVFS quickly decreases. The reasons are (i) the cooling system for processors in TX2 automatically starts to protect processors from overheating damages [27], and (ii) the default DVFS governor drops the CPU frequency to a very low level (from 2.05 GHz to 800 MHz). As we know, low frequencies will significantly slow down the training process, which may result in more energy consumption. Besides, we find that the CPU frequency managed by the default DVFS has large fluctuations, when the temperature is high. By contrast, the proposed thermal-aware DVFS scheduling ensures that the CPU temperature is well regulated, so that the peak temperature in *EEFL* will not exceed the temperature threshold. For example, when the CPU temperature approaches to $59°C$ at around 145 s, the thermal-aware DVFS will gradually lower the CPU frequencies from 1.42 GHz to 1.136 GHz. We observe that compared with default DVFS, the thermal-aware DVFS ensures not only the frequency dropping is small, but also the frequency adjustment process is very smooth. In addition, the thermal-aware DVFS is learning task specific, so that its smooth frequency adjustment will maintain processors working around their optimal VF states for a given local training task. Similar analyses apply to GPU temperature and frequency adjustment as well. All those advantages of the thermal-aware DVFS help it save more energy than default DVFS, and achieve good energy efficiency performance, which is close to the estimated ideal case without considering thermal throttling, as shown in Fig. 10(c). Similar frequency adjustment and temperature regularization can also been observed on the Android smartphones when the on-device training task is CNN@MNIST, as shown in Fig. 11. Figure 11(a) shows the temperature changes during the entire training process, and Fig. 11(b) displays the zoomed-in temperature turning point from 1180 s to 1270 s.

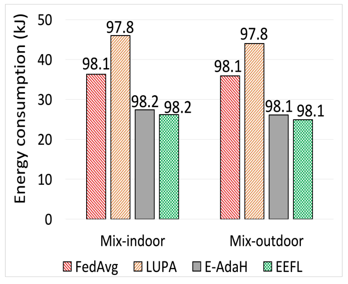

(a) Energy consumption (VGG16@CIFAR10).


(b) AdaH policy in EEFL (VGG16@CIFAR10).


(c) Energy consumption (LSTM@Shakespeare).


(d) AdaH policy in EEFL (LSTM@Shakespeare).


(e) Uplink Tx. rate traces of Wi-Fi 5 and 5G.


(f) Energy consumption (CNN@MNIST).

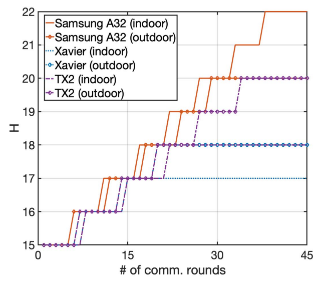
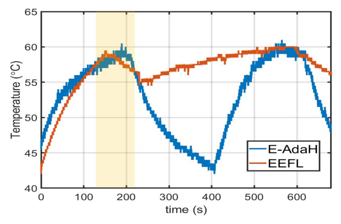
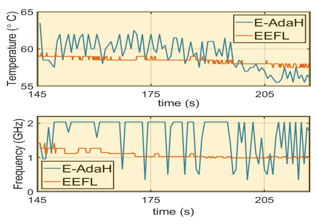

(g) AdaH policy in EEFL (CNN@MNIST).

Figure 9: Performance of FL designs under different wireless Tx. rates. For (a), (c), and (e), the numbers above the bars are the model accuracy of different methods.


(a) The temperature profile of CPU: *E-AdaH* vs *EEFL*.


(b) Zoomed in CPU temperature and frequency changes in Fig. 10(a).

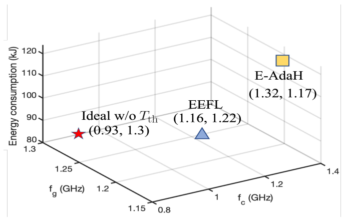
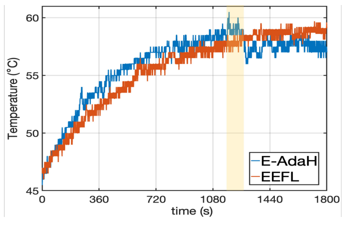
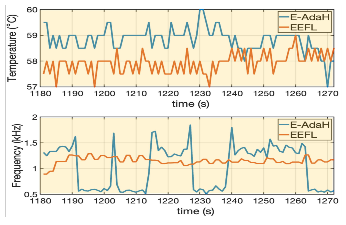

(c) Training energy: (f_c, f_g) is the averaged CPU & GPU frequencies.

Figure 10: Performance comparison of default DVFS and thermal-aware DVFS on Jeston TX2 (VGG16@CIFAR10).


(a) The temperature profile of CPU: *E-AdaH* vs *EEFL*.


(b) Zoomed in CPU temperature and frequency changes in Fig. 11(a).

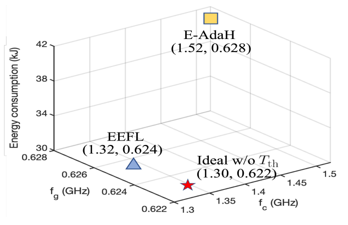

(c) Training energy: (f_c, f_g) is the averaged CPU & GPU frequencies.

Figure 11: Performance comparison of default DVFS and thermal-aware DVFS on Samsung A32 (CNN@MNIST).

6.4 Impacts of Data Heterogeneity

We further evaluate the performance of *EEFL* with different data distributions in the context of skew class distribution and unbalanced number of training data samples. As shown in Fig. 12, we find that training with non-i.i.d data incurs more energy than that with i.i.d data. The results also indicate that the adaptive local SGD policy helps to improve the model performance and save the training energy consumption in both i.i.d and non-i.i.d cases. Furthermore, since the proposed *EEFL* adopts both energy-aware AdaH policy and thermal-aware DVFS scheduling, it has the best performance in terms of energy efficiency for FL over mobile devices.

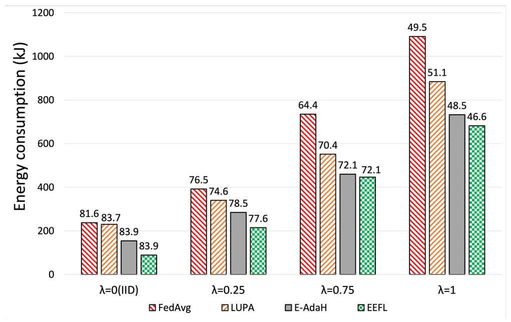


Figure 12: Energy consumption under data heterogeneity (VGG16@CIFAR10). The numbers above the bars represent the testing accuracy.

6.5 Overhead

The average computing time of solving the MCKP formulated in Eqn. (8) per device is 780 ms, which is equal to 0.17%, 0.8%, 3% and 3.6% of the training time for VGG16, Resnet, LSTM and CNN. The average energy consumption of solving the MCKP is 2.25J. This overhead is totally acceptable, given EEFL's huge margin in energy efficiency improvement.

7 DISCUSSION & FUTURE WORK

We plan to integrate some potential extensions into EEFL in the future. Note that all extensions below are compatible with EEFL and not against any analyses/results in this paper.

Integration with model compression techniques. As explained in Sec. 3.1, although we know model compression techniques (e.g., model pruning, quantization, and sparficication) help to reduce computing workloads and energy, they are not included in current EEFL design due to the learning performance loss and hardware limitation. We plan to expand our EEFL testbed with FPGA based Xilinx Zynq [49] and hardware and software co-designs to support and show the actual advantages of model compression. By integrating those compression techniques, it's worth investigating how compression parameters affect the computing-communication energy trade-off, AdaH policy, thermal-aware DVFS, etc.

Integration with more heterogeneous mobile devices. Besides Xilinx Zynq, we would like to enrich the testbed with more mobile devices of various types, e.g., Jetson Nano, Raspberry Pi 4, tablets supporting on-device training, etc. We will study how to customize their local SGD policy and DVFS. Considering heterogeneous computing and communication capabilities among devices, we also plan to upgrade EEFL design by jointly optimizing the energy consumption of individual mobile device, fairness and learning performance, and evaluate it under the updated EEFL testbed.

Integration with richer wireless environments. We also plan to evaluate the EEFL's performance under more wireless environments, such as mobile devices are moving, wireless network contention/congestion situations, Wi-Fi 6 transmission speed, etc. Moreover, We plan to concatenate current mobile device development kits (e.g., Jetson TX2) with USRPs (e.g., USRP B210 and VERT2450 Antenna) and employ open-source GNU Radio software to integrate different communication approaches into EEFL platform.

Integration with privacy preserving scheme. EEFL will not cause extra privacy leakage and is compatible with most privacy preserving FL approaches, including PPFL [32]. For example, using SGX to execute layer-by-layer training in PPFL may result in more energy consumption, but the integration of PPFL will not affect the effectiveness and trends of energy-efficient analysis in the EEFL framework.

Integration with FL user selection/participation. Most existing FL user selection schemes rely on either local update's contributions or delay. EEFL helps to add another dimension, energy efficiency, to the selection criteria. Besides, based on energy consumption estimation in EEFL, devices can choose to participate in this round FL training or not.

8 RELATED WORK

Communication efficient FL. Numerous techniques and algorithms have been proposed to improve the communication efficiency of FL. For example, Alistarh et al. in [2] exploited stochastic quantization to reduce the size of local model updates for communication efficiency. Stich et al in [47]) employed randomized sparsification to achieve the same purpose. In particular, Haddadpour et al. in [17] and Jiang et al. in [22] suggested that the number of global communication rounds can be reduced by gradually increasing the number of local training iterations, referred as "AdaH" policy in this paper, and AdaH policy help to improve model generalization by capturing the inherent nature of the learning process (e.g., the model variance among the devices [22]). Most prior works only focused on reducing the communication costs while ignoring the energy of local on-device computing and the advance of wireless technologies.

DVFS for mobile devices. Most DVFS scheduling efforts for mobile devices is to meet the delay requirements of some applications (video streaming, online game, etc.) without violating thermal throttling. For example, Choi et al. in [11] designed an overlaid DVFS scheme by adjusting maximum CPU frequency and minimum GPU frequency based on frame load prediction to improve graphics pipeline in gaming applications. Kim et al. in [23] proposed a deep reinforcement learning based DVFS to improve energy efficiency and manage the temperature of mobile devices, considering the application performances (i.e., frame per second in video applications). However, those DVFS designs are not learning task specific, and cannot be hammered directly into processors' management for on-device training.

Energy efficient FL over mobile devices. To reduce FL's energy consumption, research efforts have been made on network optimization [51], device scheduling [37] and resource utilization optimization [7, 28, 33, 43, 44, 52, 54]. For example, Mo et al. in [33] designed the computing and communication resources allocation to minimize the energy consumption but targeted CPU models for mobile devices only. Zeng et al. [54] proposed to partition the computing workload between CPU-GPU to improve the computing energy efficiency. Chen et al. [7] and Li et al. [28] considered gradient compression and fixed local SGD policy to reduce the communication energy and determine heterogeneity-aware gradient compression strategies. Unfortunately, those designs depend on a

Table 3: Comparison of latency speedup on FedAvg, LUPA and EEFL

	CNN@MNIST	ResNet@CIFAR10	VGG@CIFAR10	LSTM@Shakespeare	LSTM@Reddit	CNN@HAR
FedAvg	1x	1x	1x	1x	1x	1x
LUPA	0.35x	0.72x	1.21x	0.33x	0.21x	0.98x
EEFL	**1.45x**	**2.49x**	**1.69x**	**1.33x**	**1.78x**	**1.43x**

fixed number of local iterations and lack consideration of thermal throttling.

9 CONCLUSION

In this paper, we developed an energy efficient FL (EEFL) design, whose goal is to reduce FL's total energy consumption (computing + communication) over mobile devices. We observed (i) the huge amount of energy saved by high-speed wireless communications during the FL training, (ii) the inefficiency of existing adaptive local SGD policy, and (iii) the inefficiency of default DVFS. To address those inefficiencies, EEFL presents a novel energy-aware adaptive local SGD policy considering wireless transmission conditions and the computing-communication energy trade-off. Given the upgraded adaptive local SGD policy, EEFL further customizes DVFS to learning tasks, keeps processors (GPU and CPU) working at appropriate frequencies without triggering thermal throttling, and schedules DVFS to minimize the on-device training's energy consumption. Extensive experimental results have demonstrated the effectiveness of the proposed EEFL, and its energy efficiency superiority over peer designs under various FL tasks, DNN models, datasets and wireless transmission rates.

ACKNOWLEDGEMENT

We thank anonymous reviewers and shepherd for their valuable suggestions and feedback. The work of R. Chen and M. Pan was supported in part by NSF under Grant CNS-2107057. The work of Q. Wan and X. Fu was partially supported by NSF under Grants CCF-2130688, CCF-1900904, and CNS-210705. The work of X. Qin was supported by the Young Elite Scientists Sponsorship Program by CAST (Grant No. 2021QNRC001). The work of Y. Hou is partly supported by the National Key R&D Program of China under Grant 2019YFE0114000. The work of D. Wang was supported in part by BAS/1/1689-01-01, URF/1/4663-01-01, FCC/1/1976-49-01 of KAUST, and a funding of the SDAIA-KAUST AI center.

A LATENCY COMPARISON BETWEEN ADAH & FIX H POLICY

Here, we compare the latency of FedAvg, LUPA and EEFL on different learning tasks. The results are shown in Table 3. Specifically, for ResNet20@CIFAR10, the latency of FedAvg, LUPA and EEFL is 760,030 s, 844,377 s and 304,394 s, respectively, when the training loss reaches 0.28. EEFL is 2.49 times faster than FedAvg and 2.77 times faster than LUPA. Similar trends can be observed in other learning tasks.

Since EEFL considers the trade-off between computing and communication energy consumption, it allows devices to "work" less and "talk" more when the transmission rate is high. The similar trade-off also exists in computing and communication latency. For example, in ResNet20 @CIFAR10, the communication latency is 90ms under 100 Mbps (478 ms under 20 Mbps) and the computing latency of one local iteration of Jetson Xavier is 86ms. Hence, EEFL encourages devices to "talk" more under high-speed transmission, outperforming FedAvg and LUPA in terms of latency efficiency.

B IMPACTS OF DIFFERENT α FUNCTIONS

We also evaluate EEFL using other decreasing functions in addition to the linear decreasing function, including an exponentially decreasing function and a rational function. As shown in Table 4, we can see that EEFL integrated with other decreasing functions shows general effectiveness. Different types of α functions can be utilized adaptively for various learning tasks to achieve the desired performance. They can be selected based on a small dataset training before the large-scale training. We leave how to determine a global optimal linear function for a given model and task for future work.

Table 4: Testing accuracy vs type of α function

	VGG@CIFAR10	LSTM@Shakespeare	CNN@MNIST
Linear	83.8	47.3	98.4
Exponential	83.1	47.2	98.1
Rational	83.5	47.3	98.2

REFERENCES

[1] 3GPP. 2019. *Technical Specification Group Services and System Aspects; Release 15 Description; Summary of Rel-15 Work Items* . Technical Specification (TS) 21.915. 3rd Generation Partnership Project (3GPP). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3389 Version 2.0.0.

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*. NIPS, Long Beach, CA, 1709–1720.

[3] ASM. 2021. Aluminum 6063-T5 ASM Material Data Sheet. http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MA6063T5.

[4] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. 2018. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*. JMLR.org, Vienna, Austria, 560–569.

[5] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro Porto Buarque de Gusmao, and Nicholas D Lane. 2021. FLOWER: A friendly federated learning framework. *arXiv preprint arXiv:2007.14390* (2021).

[6] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[7] Rui Chen, Liang Li, Kaiping Xue, Chi Zhang, Miao Pan, and Yuguang Fang. 2022. Energy Efficient Federated Learning over Heterogeneous Mobile Devices via Joint Design of Weight Quantization and Wireless Transmission. *IEEE Transactions on Mobile Computing* (2022), 1–13.

[8] Rui Chen, Dian Shi, Xiaoqi Qin, Dongjie Liu, Miao Pan, and Shuguang Cui. 2023. Service Delay Minimization for Federated Learning over Mobile Devices. *IEEE Journal on Selected Areas in Communications* 41, 4 (2023), 990–1006.

[9] Yitao Chen, Saman Biookaghazadeh, and Ming Zhao. 2019. Exploring the capabilities of mobile devices in supporting deep learning. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. Association for Computing Machinery, NY, Arlington, Virginia, 127–138.

[10] Minki Cho, William Song, Sudhakar Yalamanchili, and Saibal Mukhopadhyay. 2012. Thermal system identification (TSI): A methodology for post-silicon characterization and prediction of the transient thermal field in multicore chips. In *2012 28th Annual IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM)*. IEEE, San Jose, California, 118–124.

[11] Yonghun Choi, Seonghoon Park, and Hojung Cha. 2019. Graphics-aware power governing for mobile devices. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. Association for Computing Machinery, Seoul, South Korea, 469–481.

[12] Eric V Denardo. 2012. *Dynamic programming: models and applications*. Courier Corporation.

[13] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[14] Canh T Dinh, Nguyen H Tran, Minh NH Nguyen, Choong Seon Hong, Wei Bao, Albert Y Zomaya, and Vincent Gramoli. 2020. Federated learning over wireless networks: Convergence analysis and resource allocation. *IEEE/ACM Transactions on Networking* 29, 1 (2020), 398–409.

[15] Ian Fette and Alexey Melnikov. 2021. The websocket protocol. https://www.hjp.at/doc/rfc/rfc6455.html. RFC 6455, IETF. Accessed April 4, 2021.

[16] Yonggan Fu, Han Guo, Meng Li, Xin Yang, Yining Ding, Vikas Chandra, and Yingyan Lin. 2021. CPT: Efficient Deep Neural Network Training via Cyclic Precision. In *International Conference on Learning Representations*. Virtual.

[17] Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck R Cadambe. 2019. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. In *Neural Information Processing Systems*. NIPS, Vancouver, Canada.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* (2015).

[19] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. Venice, Italy, 1389–1397.

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

[21] Bert hubert. 2021. The Wonder Shaper. https://lartc.org/wondershaper/.

[22] Peng Jiang and Gagan Agrawal. 2020. Adaptive Periodic Averaging: A Practical Approach to Reducing Communication in Distributed Learning. *arXiv preprint arXiv:2007.06134* (2020).

[23] Seyeon Kim, Kyungmin Bin, Sangtae Ha, Kyunghan Lee, and Song Chong. 2021. zTT: learning-based DVFS with zero thermal throttling for mobile devices. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. Association for Computing Machinery, Virtual Conference, 41–53.

[24] Young Geun Kim and Carole-Jean Wu. 2021. AutoFL: Enabling Heterogeneity-Aware Energy Efficient Federated Learning. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. Association for Computing Machinery, Virtual Conference, 183–198.

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

[26] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient Federated Learning via Guided Participant Selection. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*. USENIX Association, Virtual Conference, 19–35.

[27] Youngmoon Lee, Hoon Sung Chwa, Kang G Shin, and Shige Wang. 2018. Thermal-aware resource management for embedded real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 11 (2018), 2857–2868.

[28] Liang Li, Dian Shi, Ronghui Hou, Hui Li, Miao Pan, and Zhu Han. 2021. To Talk or to Work: Flexible Communication Compression for Energy Efficient Federated Learning over Heterogeneous Mobile Edge Devices. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Virtual Conference, 1–10.

[29] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[30] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 2031–2063.

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

[32] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. 2021. PPFL: privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th annual international conference on mobile systems, applications, and services*. Virtual Conference, 94–108.

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

[34] Patrick Mochel. 2021. The sysfs Filesystem. https://www.kernel.org/doc/ols/2005/ols2005v1-pages-321-334.pdf.

[35] Monsoon-solutions. 2019. High Voltage Power Monitor. http://www.msoon.com/LabEquipment/PowerMonitor/.

[36] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. 2021. Federated Learning for Internet of Things: A Comprehensive Survey. *arXiv preprint arXiv:2104.07914* (2021).

[37] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *IEEE International Conference on Communications (ICC)*. IEEE, Shanghai, China.

[38] Nvidia. 2021. Jetson TX2 Module. https://developer.nvidia.com/embedded/jetson-tx2.

[39] Mahesh Pawar, Anjana Panday, Ratish Agrawal, and Sachin Goyal. 2019. Developing a Big-Data-Based Model to Study and Analyze Network Traffic. In *Big Data and Knowledge Sharing in Virtual Organizations*. IGI Global, 198–223.

[40] Xinchi Qiu, Titouan Parcollet, Daniel J Beutel, Taner Topal, Akhil Mathur, and Nicholas D Lane. 2020. Can federated learning save the planet? *arXiv preprint arXiv:2010.06537* (2020).

[41] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. 2020. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*. PMLR, 8253–8265.

[42] Pablo Serrano, Andres Garcia-Saavedra, Giuseppe Bianchi, Albert Banchs, and Arturo Azcorra. 2014. Per-frame energy consumption in 802.11 devices and its implication on modeling and design. *IEEE/ACM Transactions on networking* 23, 4 (2014), 1243–1256.

[43] Dian Shi, Liang Li, Rui Chen, Pavana Prakash, Miao Pan, and Yuguang Fang. 2021. Towards Energy Efficient Federated Learning over 5G+ Mobile Devices. *arXiv preprint arXiv:2101.04866* (2021).

[44] Wenqi Shi, Sheng Zhou, Zhisheng Niu, Miao Jiang, and Lu Geng. 2020. Joint device scheduling and resource allocation for latency constrained wireless federated learning. *IEEE Transactions on Wireless Communications* (2020).

[45] Jaemin Shin, Yuanchun Li, Yunxin Liu, and Sung-Ju Lee. 2022. FedBalancer: data and pace control for efficient federated learning on heterogeneous clients. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*. 436–449.

[46] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[47] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. 2018. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*. Montréal, CANADA, 4447–4458.

[48] Cong Wang, Yuanyuan Yang, and Pengzhan Zhou. 2020. Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity. *IEEE Transactions on Parallel and Distributed Systems* 32, 2 (2020), 394–410.

[49] Xilinx. 2021. Zynq UltraScale+ MPSoC. https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html.

[50] Zichen Xu, Li Li, and Wenting Zou. 2019. Exploring federated learning on battery-powered devices. In *Proceedings of the ACM Turing Celebration Conference-China*. ACM, China, 1–6.

[51] Kai Yang, Tao Jiang, Yuanming Shi, and Zhi Ding. 2020. Federated learning via over-the-air computation. *IEEE Transactions on Wireless Communications* 19, 3 (2020), 2022–2035.

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

[53] Dongdong Ye, Rong Yu, Miao Pan, and Zhu Han. 2020. Federated Learning in Vehicular Edge Computing: A Selective Model Aggregation Approach. *IEEE Access* 8 (2020), 23920–23935.

[54] Qunsong Zeng, Yuqing Du, Kaibin Huang, and Kin K Leung. 2021. Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing. *IEEE Transactions on Wireless Communications* 20, 12 (2021), 7947 – 7962.

[55] Yufeng Zhan, Peng Li, and Song Guo. 2020. Experience-driven computational resource allocation of federated learning by deep reinforcement learning. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, New Orleans, LA, 234–243.

[56] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. 2018. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*. ECCV, Munich, Germany, 365–382.

[57] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. 2010. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. IEEE, Scottsdale, AZ, 105–114.