BoFL: Bayesian Optimized Local Training Pace Control for Energy Efficient Federated Learning

Hongpeng Guo
UIUC
Urbana, Illinois
hg5@illinois.edu

Haotian Gu
UC Berkeley
Berkeley, California
haotian_gu@berkeley.edu

Zhe Yang
UIUC
Urbana, Illinois
zheyang3@illinois.edu

Xiaoyang Wang
UIUC
Urbana, Illinois
xw28@illinois.edu

Eun Kung Lee
IBM Research
Yorktown Heights, New York
eunkyung.lee@us.ibm.com

Nandhini Chandramoorthy
IBM Research
Yorktown Heights, New York
Nandhini.Chandramoorthy@ibm.com

Tamar Eilam
IBM Research
Yorktown Heights, New York
eilamt@us.ibm.com

Deming Chen
UIUC
Urbana, Illinois
dchen@illinois.edu

Klara Nahrstedt
UIUC
Urbana, Illinois
klara@illinois.edu

ABSTRACT

Federated learning (FL) is a machine learning paradigm that enables a cluster of decentralized edge devices to collaboratively train a shared machine learning model without exposing users' raw data. However, the intensive model training computation is energy-demanding and poses severe challenges to end devices' battery life. In this paper, we present BoFL, a training pace controller deployed on the edge devices that actuates the hardware operational frequencies over multiple configurations to achieve energy-efficient federated learning. BoFL operates in an *explore-then-exploit* manner within limited rounds of FL tasks. BoFL explores the large hardware frequency space strategically with a tailor-designed Bayesian optimization algorithm. BoFL first find a set of good operational configurations within few task training rounds, and then exploits these configurations in the remaining rounds to achieve minimized energy consumption for model training. Experiments on multiple real-world edge devices with different FL tasks suggest that BoFL can reduce energy consumption of model training by around 26%, and achieve near-optimal energy efficiency.

CCS CONCEPTS

• **Computing methodologies** → **Distributed computing methodologies**; **Learning paradigms**; • **Hardware** → **Power estimation and optimization**.

KEYWORDS

Federated learning, Energy efficient, Dynamic voltage and frequency scaling, Bayesian optimization

ACM Reference Format:
Hongpeng Guo, Haotian Gu, Zhe Yang, Xiaoyang Wang, Eun Kung Lee, Nandhini Chandramoorthy, Tamar Eilam, Deming Chen, and Klara Nahrstedt. 2022. BoFL: Bayesian Optimized Local Training Pace Control for Energy Efficient Federated Learning. In *23rd ACM/IFIP International Middleware Conference (Middleware' 22), November 7-November 11, 2022, Quebec QC, Canada.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3458305.3463381

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Middleware 22, November 7-November 11, 2022, Quebec QC, Canada

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-8434-6/21/09…$15.00
https://doi.org/10.1145/3458305.3463381

1 INTRODUCTION

Federated learning (FL) is a privacy-preserving machine learning architecture that performs collaborative model training with large amount of resource-constrained edge devices (e.g. IoT devices, smartphones, etc.) in a distributed way, while keeping all the raw data locally on each device [16, 37, 71]. With FL, diverse privacy-sensitive domains have been drastically improved by the advancement of AI power. Such domains include cancer diagnosis [35, 46], human action detection [25], surveillance video analytics [10, 30, 79], and clinical decision support for COVID-19 [21]. Google also deployed a large-scale federated learning system over millions of real-world devices to improve their keyboard query suggestion model [77].

In a typical FL system, as depicted in figure 1, all the edge devices are organized around a central server, which orchestrates the distributed model training in a round-by-round manner. Within each round, the central server first selects a group of participants from the large client pool to train the model. All selected devices will download a shared model from the server, and train it with their private-owned data. The participants are required to compute the gradient updates timely before a server-specified deadline, and then upload their local gradients back to the server. The server will aggregate the gradients from all the devices into a synchronized update to the global model, and initiate a new round of training accordingly.

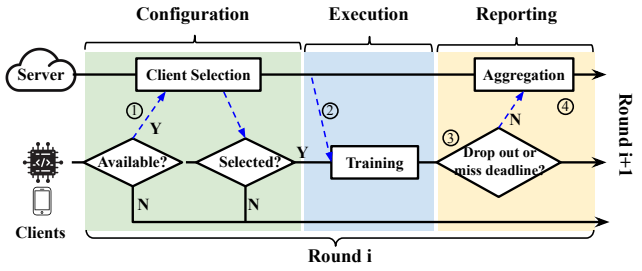


Figure 1: Standard Federated Learning Workflow. This figure is modified based on the FedScale paper [43].

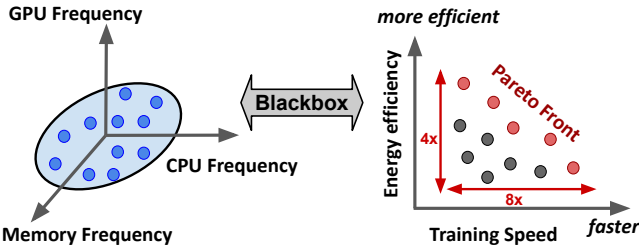


Figure 2: Complicated relation between hardware frequencies and model training performance (i.e. energy efficiency v.s. speed).

Despite the success of FL across various domains, it poses critical energy challenges to the edge devices, which are usually resource-constrained with limited battery power. For each edge device, the on-device model training involves intensive cooperation across multiple hardware processing units, i.e. CPU, GPU, and memory controller. This procedure is energy-consuming and usually burns out the device's battery in short time. Several works [40, 44, 68, 78] have been proposed to pursue better energy efficiency in FL, but most of them approach this problem from the server's perspective. For example, AutoFL [40] reduces the overall energy consumption by selecting a smaller group of participants in each round that are more likely to complete the model training before deadline. Although such design achieves global energy efficiency, it does not solve the energy challenge for individual devices. SmartPC [44] proposed an energy-aware training pace control solution based on CPU dynamic voltage and frequency scaling (DVFS) for edge devices (e.g., slow down CPU clock rate to achieve better energy efficiency). But this solution only works for CPU devices, and cannot be extended to modern edge devices that train neural networks jointly through GPU and CPU. Meanwhile, SmartPC modeled a linear relation between the training speed and CPU operational frequency. Such linear assumption fails to generalize to modern edge devices with multi-axes DVFS configurations, such as Nvidia Jetson AGX board [4], and the actual relation between performance and hardware configurations is highly-nonlinear (§2.2).

In this work, we present BoFL, a **system** deployed on each FL client locally to achieve energy-efficient training pace[1] control over multi-axes of DVFS configurations. As shown in figure 2, model training performance, i.e., training speed and energy efficiency, can be drastically affected by different operational frequencies of CPU, GPU and memory controller. A proper DVFS configuration may lead to 8× faster training speed and 4× less energy consumption. However, the 'configuration-to-performance' correspondence is highly non-linear and task-dependent (§2.2), thus, it is difficult to find and apply good configurations directly. Different from works such as [44] that builds explicit performance models on 1-D DVFS configurations, BoFL treats the multi-dimensional performance metric on multi-axes DVFS space as a blackbox function, and searches for the Pareto set with blackbox optimization tools. BoFL operates in a fully online manner within limited rounds of a FL task, and achieves near-optimal energy efficiency.

BoFL highlights three **challenges** to find the Pareto optimal configurations over multiple various-length training rounds and achieves optimal energy efficiency, as follows.

(**C1**) How to search for the set of Pareto optimal configurations in terms of energy efficiency and training speed *efficiently* in an *online* manner?

(**C2**) How to *balance* the effort between *exploring* the Pareto front and *exploiting* local optimal configurations within limited number of task rounds?

(**C3**) How to embed the *non-trivial* exploration algorithms with time-critical training tasks, while satisfying *multifaceted* system constraint and requirements?

To tackle these challenges, we design BoFL to operate in an *explore-then-exploit* manner. In the limited rounds of FL tasks, BoFL first explores the DVFS configuration space with a few trials, and then exploits the remaining rounds with the best configurations observed. Specially, BoFL strategically explores the large configuration space with multi-objective Bayesian optimization (MBO) framework, which searches for a set of Pareto trade-offs in the *energy-latency* 2-D performance space efficiently in just a few steps. The obtained Pareto optimal configurations can be exploited in later rounds adaptively with respect to the various deadlines (**C1**). To balance exploration and exploitation, we categorize the multi-round FL task into three phases: two short phases for exploration and one long phase for exploitation. We use hypervolume improvement indicator (§4) to determine the length of exploration phases, so that BoFL can construct near-optimal Pareto front within few rounds of exploration (**C2**). We run MBO calculation and the model training separately to avoid introducing extra latency overhead to the time-critical FL tasks. To cope with the uncertainty caused by exploring the whole configuration space (for example, BoFL may explore a straggler configuration and exacerbate the deadline challenge), we design a *safe exploration algorithm* to make sure every training deadline is being met (**C3**). Overall, this paper makes the following **contributions**:

(**1**) We depict the complex relation between hardware operational frequencies and Neural Network (NN) training performance through a comprehensive measurements with multiple network models over different devices (§2.2).

(**2**) We develop a blackbox optimization framework for the energy-aware training pace control problem, and propose a MBO based

[1] In this paper, *training pace* refers to the hardware processing speed, i.e, the operational frequencies of CPU, GPU, and memory controller, when training neural networks.

solution which achieves near-optimal energy efficiency in an online manner (§3).

(3) We tailor-design the MBO workflow to embed it into the time-critical FL tasks (§4), and implement the BoFL solution which achieves both smart exploration and efficient exploitation, despite multifaceted system constraints (§5).

(4) We perform comprehensive experiments over multiple neural networks and devices to evaluate BoFL's effectiveness. Evaluation results suggest that BoFL can reduce more than 20% of energy consumption compared to a performant baseline. The energy overhead of BoFL is as low as 1.2% - 3.4% compared to an optimal oracle target (§6).

The rest of the paper is organized as follows. In §2, we present backgrounds about federated learning and Bayesian optimization. We also depict the complexity of NN training performance with different hardware frequency settings to motivate our solution. In §3, we present the BoFL system model and optimization framework. We present BoFL system workflow and design details in §4. §5 and §6 present the implementation and evaluations of BoFL system. We survey related literature in §7. Finally, §8 concludes the paper.

2 BACKGROUND AND MOTIVATION

2.1 Energy Efficient Federated Learning

Federated learning is a machine learning paradigm that enables collaborative model training from a large pool of edge devices with locally stored data. While a lot of works have been proposed to achieve fast model convergence [39, 45, 48, 56], protect client data privacy [9, 12, 57], defend against malicious attacks [13–15], and addressing source of data bias [31, 81], not many works have been presented to improve the energy efficiency of federated learning tasks.

SmartPC [44] and AutoFL [40] are two representative works that try to optimize the energy efficiency of federated learning clients. They both adopt a two-level energy optimization solution, where (1) the cloud server, from global level, strategically selects a small group of devices with high energy efficiency, and assigns them a well-designed training deadline[2]. (2) The selected devices, from local level, adapt their training paces (e.g. CPU frequencies) to minimize energy consumption and finish all training workloads before the assigned deadline. While this two-level solution can successfully reduce energy usage, SmartPC and AutoFL both oversimplify the complexity of local pace control on edge devices. They model the training speed as a linear dependent variable of CPU or GPU frequency, which is not accurate for modern edge devices where the clock rates of CPU, GPU and memory controller jointly influence the training performance in a highly nonlinear way (§2.2).

In BoFL, we focus on an efficient and effective local training pace control algorithm that jointly controls hardware frequencies over multiple axes to achieve energy-efficient federated learning. BoFL is deployed on each edge device locally, and can smartly find the best DVFS configurations for this hardware within a few training rounds. BoFL assumes a cloud server which assigns a training deadline for each training round. Any deadline assignment algorithm, either strategically designing round deadlines [40, 44, 68, 78] or using a static timeout value [3], as shown in the vanilla system design [16], can function well with BoFL.

[2]The training deadline is referred to as execution target in AutoFL [40]

2.2 Complicated correspondence between DVFS Configurations and Training Performance

Dynamic voltage and frequency scaling (DVFS) technique has been widely applied in modern computers for energy efficient purpose, especially for resource constrained mobile or edge devices. For example, Nvidia Jetson AGX, a newly released edge device for AI workloads, has a large discrete DVFS configuration space, i.e., CPU (0.4-2.3GHz), GPU (0.1-1.4GHz) and memory controller (0.2-2.7GHz), which leads to more than 2K unique combinations.

In federated learning applications, it is important to make good choices in the large configuration space to achieve high-performance model training. For example, a good frequency choice can increase the training speed by 8×, and energy efficiency by 4×. However, it is non-trivial to find the good configurations due to the complicated correspondence between the configuration and the model training performance. Specially, we observe the correspondence has **three-fold complexities** through a measurement experiment, which trains three representative neural networks (e.g. ViT, ResNet50 and LSTM) on two different devices (Nvidia Jetson AGX and Jetson TX2), as follows.

(1) Non-linearity. As shown in figure 3(a), when the CPU clock is set to 0.4 GHz, the training speed of the ViT model sees a diminishing improvement after 1.0 GHz GPU frequency. This is because the job does not benefit from faster GPU clocks, when the slow CPU becomes the bottleneck. The energy consumption curve in figure 3(b) presents an even higher complexity that it is neither linear nor monotonic. When the GPU frequency is low, i.e., 0.7 GHz, ViT achieves better energy efficiency with 0.4 GHz CPU clock than that of 2.2 GHz CPU setting. While the GPU is configured to high frequencies, i.e., 1.4 GHz, a slow CPU saves no more energy and slows down the training speed by half. In general, we observe that the training performance is influenced by the hardware frequencies in a *non-linear* way. Different configurations may have different bottlenecks over multiple axes, which leads to a complicated correspondence.

(2) NN-model dependence. As shown in figure 4(a), the execution latencies of the three neural networks show different patterns as the CPU frequency increases. The training speeds of ViT and ResNet50 almost remain the same, while LSTM reduces its execution latency by half when increasing CPU clock rate from 0.6 GHz to 1.7 GHz. For energy consumption as shown in figure 4(b), we can see ResNet50 exhibits a steadily increasing curve, while LSTM shows a consistently decreasing curve. In general, we observe that the relation between DVFS configurations and the training performance is *NN-model dependent*. Different network models may be influenced by the hardware configurations differently.

[3]There are two types of deadline definitions in the FL literature: (1) a training deadline before which the clients must finish the gradient calculation; and (2) a reporting deadline before which the server must receive the model updates from clients, which includes the model training delay and parameter uploading latency. In BoFL, we assume the first deadline model. For servers that only specify a reporting deadline, BoFL can be easily extended to work well with a network bandwidth measurement module that can infer its training deadlines from the reporting deadlines.

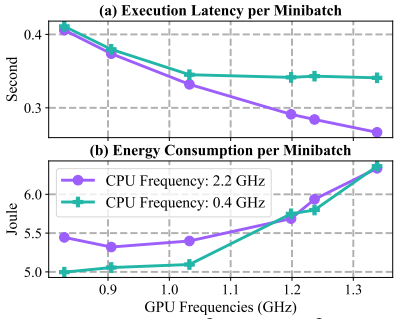


Figure 3: Training performance of ViT model with increasing GPU frequencies.

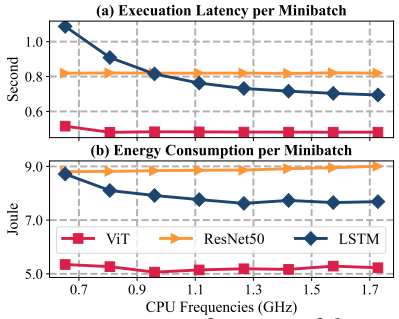


Figure 4: Training performance of three models with increasing CPU frequencies.

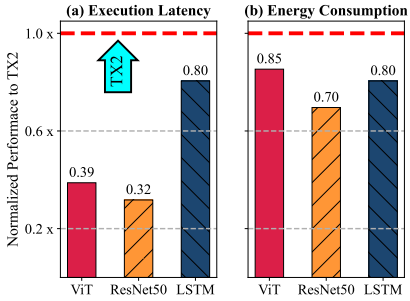


Figure 5: Normalized training performance on Jetson AGX compared to Jetson TX2.

(3) Hardware dependence. Figure 5 shows the normalized training performance of the three models on Jetson AGX compared to that of Jetson TX2 (unit performance as the red line shows). Both devices are configured with maximum operational frequencies. As a newer version with stronger hardware and updated architectures, the AGX board can significantly reduce the training time as well as the energy consumption, compared to TX2. However, the performance improvement does not apply uniformly to all three models. E.g., ResNet50 reduces its training time by 70% on AGX, while LSTM only achieves 20% execution latency reduction. The above measurements suggest that the correspondence between hardware configurations and the training performance is *hardware dependent*. E.g., it is non-trivial to estimate the performance curve of AGX board based on measurements from TX2.

In summary, the correspondence between DVFS configurations and neural network training performance is complicated and hard to be accurately modeled in an explicit way. This motivates us to model this correspondence as a blackbox function and search for the good configurations with Bayesian optimization. Our solution can search for good configuration points efficiently, and can be generally applied to any NN model on any hardware.

2.3 Multi-objective Bayesian Optimization for Blackbox Optimization Problems

The search for desirable DVFS configurations can be formulated as an optimization problem involving a multi-dimensional blackbox performance metric function (see §3.1 for details). A popular and sample-efficient solution for such optimization problem is multi-objective Bayesian optimization (MBO). MBO is an optimization framework that leverages a probabilistic surrogate model to solve optimization problems involving multiple blackbox objectives. The goal of MBO is to construct an approximated Pareto set (see §3.1) for those conflicting objectives, within a limited budget of function evaluations.

MBO sequentially selects new points to evaluate based on the surrogate model, and updates the model to incorporate new observations. To decide which point to evaluate next, MBO employs an *acquisition function* that specifies the utility of evaluating a new point based on the surrogate model's predictive distribution. A good choice of the acquisition function helps to balance the trade-off between exploration of unknown regions and exploitation of the current best-performing ones [26, 69].

MBO has been extensively studied in the literature [28, 82], and has enjoyed substantial successes in many applications, including environmental engineering [52], structural design [55] and high-energy physics [66].

We design BoFL with MBO embedded as the core algorithm for balancing exploration and exploitation in the Pareto search. The MBO workflow in BoFL as well as the choice of acquisition function is carefully designed to better fit the MBO module with other ingredients in the federated learning task. More details and reasoning of BoFL system design can be found in §4.

3 PROBLEM DEFINITION AND PRELIMINARIES

3.1 Problem formulation

Federated learning task: A federated learning task is initialized by the server and trained with a pool of edge devices in a round-by-round manner. At each round, the server selects a small group of participants from the device pool and trains the machine learning model for E epochs of the Stochastic Gradient Descent algorithm (SGD) with minibatch size of B on the selected devices using their private-owned data. The participants are required to finish the gradient calculation before a server specified training deadline, and then upload the model gradients back to the server where the gradients get averaged into a shared model. The deadline is calculated by the server as regards to the training data size and computation capabilities of the selected participants in this round. E.g., a training round using devices with stronger hardware or less training data may be assigned a shorter deadline by the server. The server usually forms different group of devices for each round to make sure the model is trained with heterogeneous data and thus not biased, which leads to various deadlines for different rounds.

In BoFL, a federated learning task can be formally defined as $(B, E, \mathbf{T}, N)$ from the perspective of a local device, where (1) B and E represent the aforementioned global parameters, minibatch size and training epoch number; (2) $\mathbf{T}$ is a vector, representing the training deadlines for the rounds when this device is selected to participate; and (3) N represents the number of minibatches of training data available on this device. E.g., a client with 1k images,

joining a federated classifier training task of minibatch size 10, has $N = 1k/10 = 100$ in this case.

DVFS configuration space: Our goal is to find the optimal or near-optimal device training pace for every round of the federated training that satisfies the corresponding training deadlines and minimizes the total energy consumption. As mentioned in §2, the training pace can be controlled by configuring the operational frequencies of the device's CPU, GPU and memory controller. We thus define the DVFS configuration space as $\mathbf{X} = \mathbf{F}_{CPU} \times \mathbf{F}_{GPU} \times \mathbf{F}_{MC}$, where the $\mathbf{F}_{(\cdot)}$s represent the discrete operational frequencies of the three hardware units (CPU, GPU and memory controller), respectively.

Energy optimization problem: For any given federated learning task on a specific device, the training speed and energy efficiency can be characterized as functions of the DVFS configurations. Formally, we define the two metrics as follows:

- $\mathcal{T}(\mathbf{x})$ is the execution **latency** to compute one minibatch of the training data under configuration $\mathbf{x}$;
- $\mathcal{E}(\mathbf{x})$ is the **energy** consumed on one minibatch of data when trained with configuration $\mathbf{x}$.

$\mathbf{x} \in \mathbf{X}$ is a three-element tuple encoding the operational frequencies of CPU, GPU and memory controller. In our system, we define the processing of a minibatch of data as a *job*. E.g., for an FL task that trains a ResNet50 model, a *job* refers to the process of feeding a minibatch of data (images) into the ResNet50 model and generating the gradient updates. We can always apply a different DVFS configuration for the next job, but no more than one configuration in the same job. We further define $W = E \times N$, which is the number of jobs in any single training round. The optimization problem can, thus, be formally presented as follows:

$$\begin{aligned} \min_{\mathbf{x}_{r,i} \in \mathbf{X}} \quad & \sum_{r=1}^{|\mathbf{T}|} \sum_{i=1}^{W} \mathcal{E}(\mathbf{x}_{r,i}) \\ \text{s.t.} \quad & \sum_{i=1}^{W} \mathcal{T}(\mathbf{x}_{r,i}) \leq \mathbf{T}_r \quad \forall\, 1 \leq r \leq |\mathbf{T}| \end{aligned} \tag{1}$$

The variable $\mathbf{x}_{r,i}$ is the DVFS configuration applied in the r-th training round on the i-th job. The optimization goal of BoFL is to minimize the overall energy consumption during the $|\mathbf{T}|$ rounds of training by carefully selecting DVFS configurations for every job, while satisfying the unique training deadline for each round.

3.2 Solution Sketch with MBO Searched Pareto Set

While the whole configuration space is large, the optimizers of Eqn. (1) can be chosen from a small **Pareto set**, defined as the set of Pareto optimal points in $\mathbf{X}$, in terms of the two metric functions, $\mathcal{E}$ and $\mathcal{T}$. More formally, let $\mathcal{M}(\mathbf{x}) = (\mathcal{E}(\mathbf{x}), \mathcal{T}(\mathbf{x}))$ be a two-objective function defined on $\mathbf{X}$, a point $\mathbf{x}_1 \in \mathbf{X}$ is Pareto dominated by another point $\mathbf{x}_2 \in \mathbf{X}$, iff $\mathcal{E}(\mathbf{x}_1) \geq \mathcal{E}(\mathbf{x}_2)$ and $\mathcal{T}(\mathbf{x}_1) \geq \mathcal{T}(\mathbf{x}_2)$, and either $\mathcal{E}(\mathbf{x}_1) > \mathcal{E}(\mathbf{x}_2)$ or $\mathcal{T}(\mathbf{x}_1) > \mathcal{T}(\mathbf{x}_2)$. We denote this by $\mathcal{M}(\mathbf{x}_1) \prec \mathcal{M}(\mathbf{x}_2)$. A point is Pareto optimal if it is not Pareto dominated by any other point. We use $\mathbf{P} \subseteq \mathbf{X}$ to denote the set of Pareto optimal points for function $\mathcal{M}$. The images of the Pareto optimal points $\mathbf{P}_f := \mathcal{M}(\mathbf{P})$ are called the Pareto front.

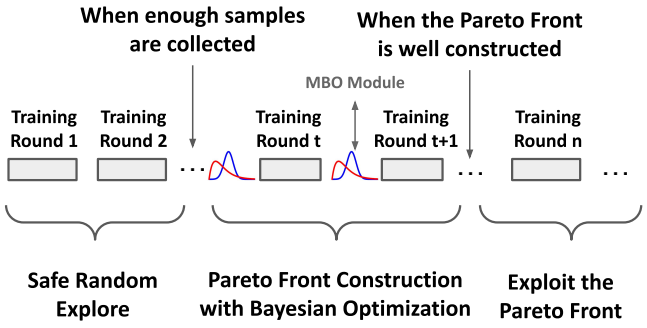


Figure 6: BoFL Workflow

It is intuitive that the variable space in Eqn. (1) can be reduced from $\mathbf{X}$ to $\mathbf{P}$ without affecting the solution: any DVFS configuration outside the Pareto front could be replaced by its dominant point in $\mathbf{P}$ to further minimize the energy objective without causing extra delay. Once the Pareto set $\mathbf{P}$ is given, it is straightforward to reformulate Eqn. (1) as an integer linear programming problem, which can be solved efficiently.

However, searching for the Pareto set $\mathbf{P}$ is a challenging task since the two metric $\mathcal{E}$ and $\mathcal{T}$ are blackbox functions that are expensive to evaluate. Due to the computational cost for configuration evaluation and the deadline requirement for training tasks, it is crucial to obtain an approximated Pareto set within limited rounds. In BoFL, we use multi-objective Bayesian optimization (MBO) as a sample-efficient method to search for the Pareto optimal points with just a few trials, and get near-optimal solution for Eqn. (1).

4 BOFL SYSTEM DESIGN

4.1 Overview

As shown in figure 6, BoFL operates in **three** phases spanning all the training rounds as follows:

(1) **Safe random exploration** samples and tries candidates from the configuration space uniformly to collect the first group of observations for MBO model initialization. It will continue for one or a few rounds at the beginning of the FL task. We design a *safe exploration algorithm* to make sure the training deadlines are being satisfied, and the observed metrics (e.g., latency and energy consumption) are being accurate.

(2) **Pareto front construction** tries on the candidates suggested by MBO. This phase will continue for several rounds (e.g., 3 to 5 rounds) to search for the Pareto optimal configurations. The MBO update is separated from the model training, using the configuration and reporting time window as shown in figure 1, to minimize system overhead generated by co-locating training and MBO calculation at the same time. To accelerate Pareto searching, the MBO algorithm will propose *multiple* candidates (in batched form) to be explored for the next round.

(3) **Exploitation** is a long phase, which usually takes more than 90% of the time in a FL task. In this phase, we solve Eqn. (1) with the approximated Pareto optimal points constructed from the second phase. The energy consumption is thus minimized with the sweet spot configurations.

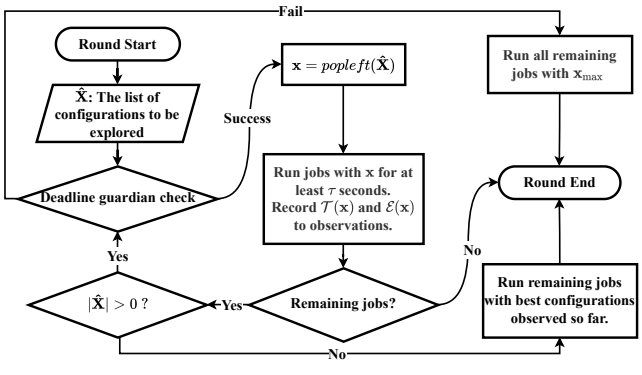


Figure 7: Safe exploration algorithm

In the rest of this section, we present the design details of the three operational phases in §4.2, §4.3 and §4.4, respectively.

4.2 Safe Random Exploration

The goal of the random exploration phase is to collect some starting points, uniformly from the space, for the MBO model to estimate the objective functions. As we sample the starting points randomly over the whole space, it is inevitable to select some bad DVFS configurations that introduce longer delays than we expect, which exacerbates the challenge to meet every training deadline in this phase. We design a safe exploration algorithm, as depicted in figure 7, to ensure the training deadlines met. We present the detailed design decisions as follows.

Sample selection: We sample a small group (e.g., 1% of the whole space) of starting points, uniformly distributed over $\mathbf{X}$, using a quasi-random number generator. This uniform exploration strategy helps the Bayesian model avoid making wrong assumptions over the sample space.

Instead of trying the starting points from the very beginning, we manually choose $\mathbf{x}_{\max}$ as the first DVFS configuration to be applied in the FL task, where

$$\mathbf{x}_{\max} = (\max(\mathbf{F}_{CPU}), \max(\mathbf{F}_{GPU}), \max(\mathbf{F}_{MC}))$$

refers to the DVFS configuration with the highest operational frequencies on all three processing units (e.g., CPU, GPU and the memory controller). $\mathbf{x}_{\max}$ is a configuration with maximum processing capability where the training can be quickly finished. After $\mathcal{T}(\mathbf{x}_{\max})$ is observed, $\mathbf{x}_{\max}$ could be used as a guardian configuration, so that we can always speed up our pace to $\mathbf{x}_{\max}$ in the middle of any exploration round to catch the training deadline before it is too late.

Workload assignment: In each exploration round, we can try multiple DVFS configurations through the W training jobs. It is important to assign a balanced workload to each configuration. A transient workload (e.g., trying the configuration for only one job) will lead to the execution being finished before the hardware voltage gets stable, and will generate large energy measurement error. Contrarily, a heavy workload prolongs exploration phase, and squeezes the exploitation phase.

In practice, we define τ as a reference measurement duration (e.g., 5s). BoFL will keep assigning a new job to configuration $\mathbf{x}$ until it has been explored for at least τ seconds. When the current job finishes and the configuration has been measured for more than τ seconds, BoFL will switch to explore the next candidate point.

Deadline guardian strategy: As we randomly sample configurations to explore in the first phase, we will inevitably meet some bad points that delay our training progress, and exacerbates the challenge to catch the training deadlines. In BoFL, we make sure the deadline requirements never violated by using the known guardian configuration $\mathbf{x}_{\max}$, as mentioned in §4.2. $\mathbf{x}_{\max}$ is tested first so that $\mathcal{T}(\mathbf{x}_{\max})$ is known before any exploration point is tried. Before exploring an unknown configuration $\mathbf{x}$, we run a quick *deadline guardian check* first to see if the remaining jobs could still be finished with $\mathbf{x}_{\max}$, even if the τ seconds of exploration on $\mathbf{x}$ fails to finish any job. Formally, let W_{remain} and T_{remain} represent the number of remaining jobs and the remaining time before deadline when configuration $\mathbf{x}$ is about to be explored, the deadline guardian check criterion could be expressed as follows,

$$T_{\text{remain}} - \tau \geq W_{\text{remain}} \times \mathcal{T}(\mathbf{x}_{\max}) \tag{2}$$

Configuration $\mathbf{x}$ would be explored only if Eqn. (2) satisfies. In case Eqn. (2) fails, configuration exploration will terminate in this round, and all the remaining jobs will be executed under configuration $\mathbf{x}_{\max}$. The random exploration phase will continue for several rounds until all the uniformly sampled starting points are explored.

Last round exploitation: In the last round of the random exploration phase, we may finish exploring all the starting points early, leaving some jobs not executed. While we can play safe to apply $\mathbf{x}_{\max}$ on the remaining jobs, this method may consume more energy than needed, as $\mathcal{E}(\mathbf{x}_{\max})$ could be high. In BoFL, we apply an exploitation strategy to finish the last random exploration round with observed configurations. We calculate the best profile of configurations from the observed starting points to minimize energy consumption, while satisfying the deadline requirement. More details of the exploitation algorithm will be presented in §4.4.

4.3 Pareto Front Construction

The Pareto front construction phase is the main module that we apply MBO algorithms to search for Pareto optimal configurations. As shown in figure 6, this phase continues for several rounds with the following two components.

(**1**) A MBO module that runs between two consecutive training rounds. It updates the function estimation with the observations from the previous rounds, and provides *a batch of suggested configurations* to explore for the next round.
(**2**) A FL task round in which the suggested configurations are explored. It ensures to satisfy the corresponding deadline with the safe exploration algorithm as presented in §4.2.

We present the reasoning and design decisions of the Pareto front construction phase as follows.

Separation of MBO computation and model training: The MBO calculation involved in Pareto searching is nontrivial, and usually takes several seconds to complete. Running the MBO module and training the learning task simultaneously can result in unexpected

running-time overhead, and increase the risk of missing deadline. Alternatively, in BoFL, the MBO module is executed only during the configuration and reporting time window as shown in figure 1, while the module will idle during the training time. Such separation will effectively minimize the running-time influence of the MBO module on the training task.

MBO prior function: Without the loss of generality, in this project, two objective functions $\mathcal{T}$ and $\mathcal{E}$ are modeled by two independent Gaussian processes [65], each of which has a prior distribution with mean function $m(\mathbf{x}) = 0$ and kernel function $k(\mathbf{x}, \mathbf{x}') = C_{5/2}(\|\mathbf{x} - \mathbf{x}'\|_2)$. Here $C_{5/2}(\|\mathbf{x} - \mathbf{x}'\|_2)$ is the widely-used Matérn-5/2 kernel function [65] that can capture a large variety of function properties.

Pareto front approximation with hypervolume improvement: Recall that the goal of our MBO algorithm is to identify a finite approximated Pareto front. To measure the quality of an approximated Pareto front, hypervolume indicator (HV) is the most commonly used metric in MBO. It quantifies the hypervolume of the region in the performance space that is dominated by the approximated Pareto front and bounded from below by a reference point. Mathematically, given an approximated Pareto front

$$\mathbf{P}' = \{\mathbf{p}_i = \mathcal{M}(\mathbf{x}_i) : \mathbf{x}_i \in \mathbf{X}, i = 1, \ldots, n\} \tag{3}$$

and a reference point $\mathbf{r} \in \mathbb{R}^2$, the hypervolume indicator $\mathrm{HV}(\mathbf{P}', \mathbf{r})$ is defined as

$$\mathrm{HV}(\mathbf{P}', \mathbf{r}) = \int_{\mathbb{R}^2} \mathbb{1}_{\mathrm{H}(\mathbf{P}', \mathbf{r})}(\mathbf{z}) d\mathbf{z}, \tag{4}$$

where $\mathrm{H}(\mathbf{P}', \mathbf{r}) := \cup_{i=i}^{n} \{\mathbf{z} \in \mathbb{R}^2 : \mathbf{r} \preceq \mathbf{z} \preceq \mathbf{p}_i\}$ is the region dominated by $\mathbf{P}'$ and bounded from above by $\mathbf{r}$. The higher the $\mathrm{HV}(\mathbf{P}', \mathbf{r})$ is, the better the $\mathbf{P}'$ approximates the true Pareto front $\mathbf{P}$. The reference point can be selected as the combination of the worst performances, for $\mathcal{T}$ and $\mathcal{E}$, we observed in phase 1, i.e., $\mathbf{r} = (\max(\mathcal{E}(\mathbf{x})), \max(\mathcal{T}(\mathbf{x}'))), \forall \mathbf{x}, \mathbf{x}' \in \widehat{\mathbf{X}}$, where $\widehat{\mathbf{X}}$ is the set of starting points explored in the random exploration phase.

To determine how much the hypervolume would increase if a set of new points $\mathbf{Q} = \{\mathbf{q}_j = \mathcal{M}(\mathbf{x}'_j) : \mathbf{x}'_j \in \mathbf{X}, j = 1, \ldots, m\}$ is added to the current Pareto front approximation $\mathbf{P}'$, we define the hypervolume improvement (HVI) of $\mathbf{Q}$ with respect to $\mathbf{P}'$ as

$$\mathrm{HVI}(\mathbf{Q}; \mathbf{P}', \mathbf{r}) = \mathrm{HV}(\mathbf{Q} \cup \mathbf{P}', \mathbf{r}) - \mathrm{HV}(\mathbf{P}', \mathbf{r}). \tag{5}$$

EHVI acquisition function: In general, the performance metric $\mathcal{M}(\mathbf{x})$ at any unobserved point $\mathbf{x} \in \mathbf{X}$ is unknown in the blackbox optimization. However, the GP surrogate model in the Bayesian framework provides a posterior distribution $\mathbb{P}(\mathcal{M}(\mathbf{x})|\mathbf{D})$ for any unobserved point $\mathbf{x} \in \mathbf{X}$, where $\mathbf{D}$ is the set of all historical observations. This allows one to define and compute the expected hypervolume improvement (EHVI) acquisition function, conditioned on historical observations $\mathbf{D}$, current approximated Pareto front $\mathbf{P}'$, and reference point $\mathbf{r}$:

$$\alpha_{\mathrm{EHVI}}(\mathbf{x}|\mathbf{D}, \mathbf{P}', \mathbf{r}) = \mathbb{E}_{\mathcal{M}(\mathbf{x}) \sim \mathbb{P}(\cdot|\mathbf{D})}[\mathrm{HVI}(\{\mathcal{M}(\mathbf{x})\}; \mathbf{P}', \mathbf{r})]. \tag{6}$$

The algorithm will select the point with maximal EHVI to evaluate in the next iteration. In practice, the 2-D EHVI value can be computed efficiently calculated in $O(|\mathbf{D}| \log(|\mathbf{D}|))$ time complexity [76].

Batch Selection Strategy: The classical formulation of EHVI (6) proposes only a single point to evaluate. However, practically, the time scale of each round in the Pareto front construction phase in BoFL allows the system to explore multiple configurations within one round. Therefore, the MBO algorithm is required to propose a batch of configurations to evaluate in the next round.

Note that the definition of EHVI (6) can be extended to a batch setting, by simply replacing the input point $\mathbf{x}$ by a batch of points, and the expectation is taken over the posterior distribution on the entire batch. However, finding the optimal batch based on such acquisition function will suffer from the high computational cost, especially when the batch size is large [20].

In BoFL, inspired by [20, 27], we adopt a batch selection strategy to select K points for the next batch in a *sequential greedy fashion* with the following three steps:

(**1**) Choose the next point $\mathbf{x}$ to be explored based on the acquisition function (6);
(**2**) Fantasize the observation on $\mathbf{x}$ from our surrogate model, i.e. $\widehat{\mathcal{M}}(\mathbf{x}) = \mathbb{E}[\mathcal{M}(\mathbf{x})|\mathbf{D}]$, and update posterior estimation accordingly;
(**3**) Repeat step (1) and (2) until K configurations are selected.

Our sequential greedy batching strategy can be easily implemented and scales well for large batch sizes.

MBO batching size selection: In every run of the MBO model, we generate a batch of $K = T_{avg}/\tau$ suggestions, which roughly estimates the average number of explorations in each round. T_{avg} here refers to the average deadline length that we have observed in the safe random exploration phase. In practice, we can also set an upper threshold for the MBO batch size (e.g. 10 suggestions) to avoid the MBO calculation running too long, and affecting the proceeding training rounds.

MBO stopping condition: The Pareto construction phase will continue until the following stopping condition is satisfied: when at least a certain number of configurations (e.g. 3% of the whole space) are explored and the EHVI value increase is less than a threshold (e.g., 1%). This stopping criterion ensures that the MBO module has explored enough configurations before stopping and does not struggle too much for small improvements.

Training round execution details: After the MBO module generates a batch of K suggestion points, they will be explored in the proceeding training round. As each round has its unique deadline which is unknown beforehand, BoFL may not have enough time to explore all the K configurations when the deadline length is short. Meanwhile, it is also possible that there are jobs remained after the K configurations are all explored.

In the Pareto front construction phase, we still follow the *safe exploration algorithm* (figure 7) to explore the Bayesian suggestion points. With deadline guardian checking, we can drop extra suggestions to make sure the deadlines are caught. In case there are remaining jobs, we exploit the best profile of configurations we have observed to minimize energy consumption. We will present the exploitation details in §4.4.

	Jetson AGX	Jetson TX2
CPU	8-core ARM v8.2	2-core Nvidia Denver2 + 4-core ARM Cortex-A57
Frequencies	**0.42GHz → 2.26GHz (25 steps)**	**0.34GHz → 2.03GHz (12 Steps)**
GPU	512-core Volta GPU	256-core Pascal GPU
Frequencies	**0.11GHz → 1.38GHz (14 steps)**	**0.11GHz → 1.30GHz (13 steps)**
Memory	32GB 256-bit LPDDR4x	8GB 128-bit LPDDR4
Frequencies	**0.20GHz → 2.13GHz (6 steps)**	**0.41GHz → 1.87GHz (6 steps)**

Table 1: BoFL Testbed Hardware Specifications

4.4 Exploitation

After the Pareto construction phase, we have explored sufficiently many DVFS configurations, from which an approximated Pareto set $\mathbf{P}'$ can be selected. In the exploitation phase, we use $\mathbf{P}'$ as a surrogate of the actual Pareto set $\mathbf{P}$ to solve for the energy minimization problem in Eqn. (1).

Although we reduced the input space from $\mathbf{X}$ to $\mathbf{P}'$, the optimization problem of Eqn. (1) is still nontrivial to solve. In general, the Pareto front curve induced by $\mathbf{P}'$ is not convex, and consequently, the minimizers of Eqn. (1) could be a combination of multiple configurations. In BoFL, we solve the Integer Linear Programming (ILP) problem with *branch-and-bound algorithm* [58], which estimates the lower and upper bounds of the search space regions efficiently, and is widely applied in many discrete optimization problems.

As mentioned in §4.2 and §4.3, the exploitation algorithm is also applied in the exploration phase (7) when the candidates are fully explored, but the jobs are not finished. In such cases, we build Pareto front based on existing observations, and solve for the minimizers on the remaining jobs before the training deadline.

5 IMPLEMENTATION

5.1 Hardware Testbed

We implement BoFL on two devices, Nvidia Jetson AGX [4] and Nvidia Jetson TX2 [5]. BoFL controls the CPU, GPU and memory controller frequencies on these testbeds whose specifications are shown in Table 1. E.g., The CPU of Jetson AGX has a clock frequency range of 25 discrete steps from 0.42GHz to 2.26GHz, and the GPU of Jetson TX2 has 13-stepped operational frequencies ranging from 0.11GHz to 1.30GHz. Overall, the DVFS configuration spaces of Jetson AGX and Jetson TX2 have 2100 and 936 unique configurations, respectively.

5.2 Software Implementation

Figure 8 depicts an overview of BoFL's software implementation. We implement BoFL with `Python` in around 2K lines of code. It has five main modules as follows.

FL task executor ❶*:* The FL task executor takes a deep learning model (e.g., ResNet50) and executes the training loops with its local data. It follows the training parameters, e.g., minibatch size and number of epochs, specified by each FL task. We implement this

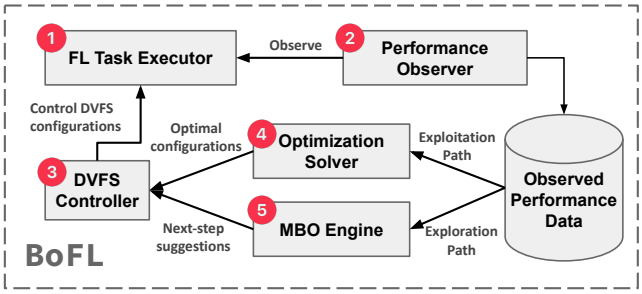


Figure 8: Architecture of BoFL's Implementation

module based on `Pytorch-1.10` [6], and it could be easily generalized to other machine learning platforms, such as `Tensorflow` [8]and `MXNet` [1].

Performance observer ❷*:* We implement the performance observer to read the execution latency $\mathcal{T}(\mathbf{x})$ and energy consumption $\mathcal{E}(\mathbf{x})$ when any DVFS configuration $\mathbf{x}$ is applied. We use CUDA event recording APIs[4] to accurately measure the execution latency. We read the energy consumption with the built-in INA3221 power sensor [3], which can be easily accessed through the `sysfs` in Linux kernel.

DVFS controller ❸*:* The DVFS controller implements the main workflow in §4 and directly actuates the hardware frequencies. During exploration, it takes the Bayesian suggestions as the next-step exploration configurations. During exploitation, it actuates the operational frequencies according to the optimization results (solution for the ILP as shown in Eqn. 1). We modify the hardware frequencies by directly writing into the corresponding `sysfs` kernel files[5].

Optimization solver ❹*:* The optimization solver solves the ILP problem, Eqn.(1), with observed Pareto optimal configurations. We build this module with `Gurobi` optimization engine [2], which implements the *branch-and-bound algorithm* and solves Eqn. (1) efficiently, i.e., within 20ms.

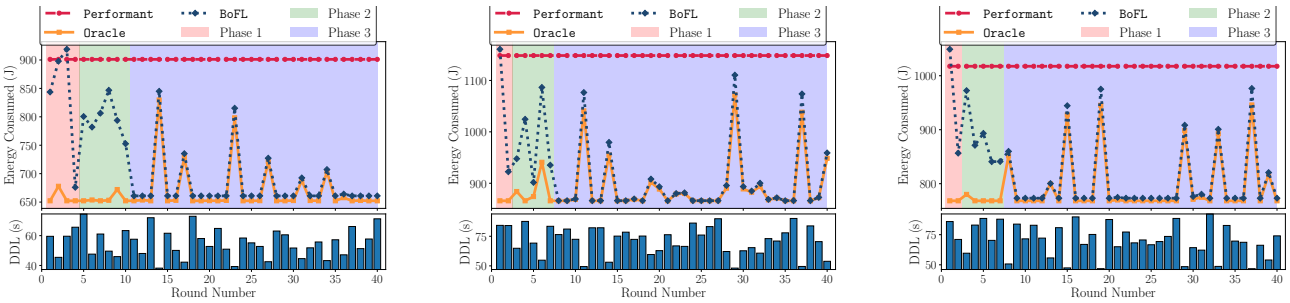
MBO engine ❺*:* Our MBO engine is built on top of `Trieste` [7] which is a Bayesian optimization library implemented in `Python`. `Trieste` implements the standard BO priors, as well as a wide range of acquisition functions and batching rules, including the *EHVI* function and *sequential greedy* rule adopted by BoFL, as discussed in §4. The MBO engine is triggered before each training round in the Pareto construction phase to update the posterior estimation of $\mathcal{T}(\cdot)$ and $\mathcal{E}(\cdot)$, and generates the next-step suggestions for the DVFS controller, ❸, to achieve efficient exploration.

6 EVALUATION

6.1 Methodology

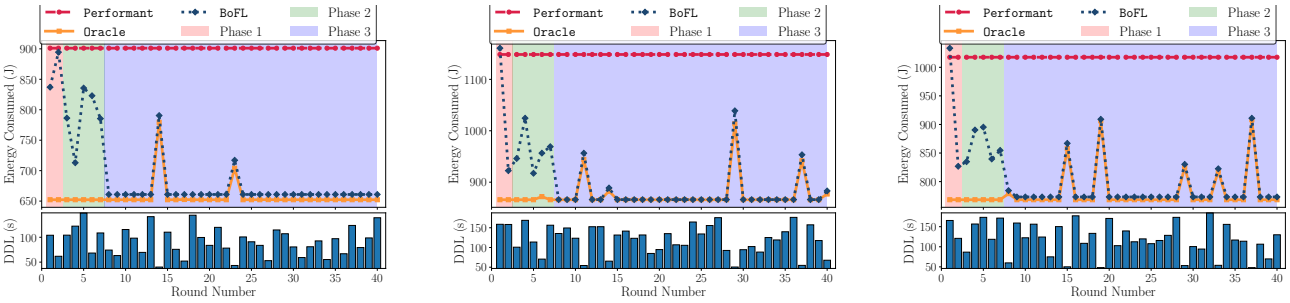
Datasets & Neural Network Models. We evaluate BoFL with three different federated learning tasks spanning both computer vision (CV) and natural language processing (NLP) applications. The three tasks cover three major types of neural network models, i.e., CNN, RNN and Transformer, as follows:

[4] `torch.cuda.Event()` and `torch.cuda.synchronize()`.
[5] e.g., writing into `"/sys/devices/*/devfreq/*/min(max)_freq"` to modify the corresponding GPU frequencies.



(a) CIFAR10-ViT (b) ImageNet-ResNet50 (c) IMDB-LSTM

Figure 9: Energy consumption for the first 40 rounds of FL training on AGX testbed with $T_{max}/T_{max} = 2$



(a) CIFAR10-ViT (b) ImageNet-ResNet50 (c) IMDB-LSTM

Figure 10: Energy consumption for the first 40 rounds of FL training on AGX testbed with $T_{max}/T_{max} = 4$

(1) **CIFAR10-ViT** trains Vision-Transformer model [23] (ViT) for image classification on the CIFAR10 dataset [42]. The CIFAR10 dataset contains $32p \times 32p$ color images of 10 different classes. This dataset is widely applied in image classification tasks for its lightweight.

(2) **ImageNet-ResNet50** trains ResNet50 model [32] on ImageNet dataset [22] for image classification. Compared to CIFAR10, ImageNet contains image data of more diverse classes and higher resolutions. When training with ImageNet, the images are usually cropped to $224p \times 224p$ for normalized and convenient data loading.

(3) **IMDB-LSTM** trains LSTM-RNN model [33] for text semantic analysis with IMDB movie review dataset [51]. This dataset contains more than $50K$ movie reviews with ground truth binary semantic labels, i.e., positive or negative. It is a widely used dataset for NLP model training.

		CIFAR10-ViT	ImageNet-ResNet50	IMDB-LSTM
B		32	8	8
E		5	2	4
N	AGX	40	90	40
	TX2	15	30	20
$\|\mathbf{T}\|$		100		
$\mathbf{T}_{min}$	AGX	37.2s	46.9s	46.1s
	TX2	36.0s	49.2s	55.6s
$\mathbf{T}_{max}/\mathbf{T}_{min}$		{2.0, 2.5, 3.0, 3.5, 4.0}		

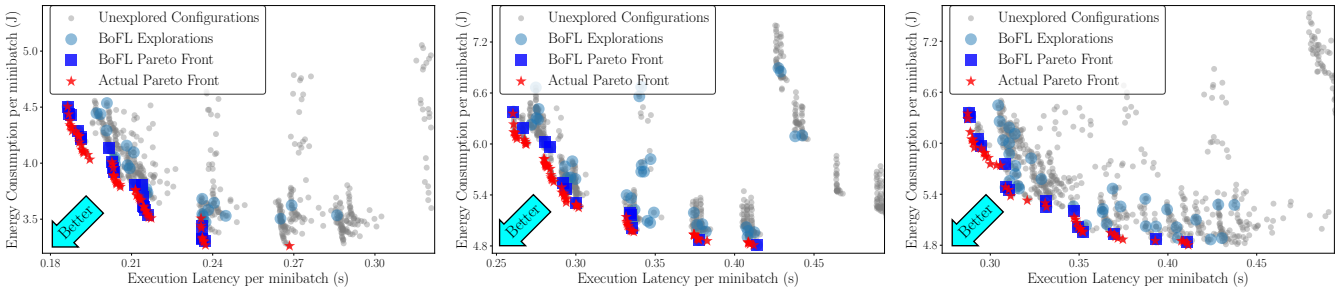
Table 2: Federated Learning Task Specifications

Experiment Setup & FL Task Specifications. To evaluate the performance of BoFL, we train the above three FL tasks on the two testbeds, i.e., Jetson AGX and Jetson TX2, each for 100 rounds. Table 2 presents the detailed specifications of the three FL tasks. For each task, we first specify its global parameters B and E, representing the minibatch size and number of training epochs in each round. We then load part of the datasets into the two devices as their private training data. As mentioned in §3, N refers to the number of minibatches. For example, we load 40 batches of data, each containing 32 images into the AGX device for task *CIFAR10-ViT*. Finally, We sample 100 deadlines uniformly from the range $[\mathbf{T}_{min}, \mathbf{T}_{max}]$. $\mathbf{T}_{min}$ is the execution latency to finish one round of training when the device is configured with maximum operational frequencies, e.g., $\mathbf{T}_{min} = \mathcal{T}(\mathbf{x}_{max}) \times W$. The $\mathbf{T}_{min}$ values in Table 2 are experiment measurements on the two testbeds when $\mathbf{x}_{max}$ is applied. A FL task can be finished on a device in time only if the assigned deadline is no less than $\mathbf{T}_{min}$. $\mathbf{T}_{max}$ is the deadline sampling upper bound. To evaluate the performance sensitivity of BoFL with different range of deadlines, we select a wide spectrum of $\mathbf{T}_{max}$ ranging from $2.0 \times \mathbf{T}_{min}$ to $4 \times \mathbf{T}_{min}$.

Comparison Targets. To evaluate the effectiveness of BoFL, we compare it with two other designs as follows:

(1) **Performant**. The Performant design is the default DVFS configuration for real-time tasks. It turns all the hardware units into maximum operational frequencies, i.e., $\mathbf{x}_{max}$, to maintain stable performance, and make sure the deadlines will not miss. We compare BoFL with the Performant design to show that our algorithm can significantly reduce the energy consumption.

(2) **Oracle**. In the Oracle design, we profile $\mathcal{T}$ and $\mathcal{E}$ over the whole configuration space offline, and only run exploitation over the FL training rounds to achieve optimal energy usage. Note that Oracle can not be achieved in practice as it requires



(a) CIFAR10-ViT (b) ImageNet-ResNet50 (c) IMDB-LSTM

Figure 11: A comparison between BoFL searched Pareto fronts and the actual Pareto fronts on AGX testbed.

long-lasting offline profiling. We compare BoFL with the Oracle design to show that we achieve near-optimal energy efficiency with little regret.

6.2 BoFL Energy Efficiency

We evaluate the energy efficiency of BoFL as shown in in figure 9 and 10. We plot the energy consumption of BoFL, as well as the two baselines, for the first 40 training rounds. The deadlines for each round are presented together with the energy usage. We also highlight the three algorithm phases of BoFL to better illustrate the *exploration and exploitation* trade off in our solution.

Figure 9a depicts our experiment results for *CIFAR10-ViT* task measured on the AGX testbed. As the figure shows, BoFL can reduce the overall energy consumption substantially comparing to the Performant baseline, and achieves pretty close energy usage comparing to the Oracle target in the exploitation phase. BoFL takes the first 10 training rounds (phase 1 & 2) to explore the configuration space, and runs exploitation in all the remaining rounds. BoFL outperforms Performant consistently over the whole training process, except one round in phase 1 when BoFL inevitably meet some bad configurations during random exploration. Comparing to the Oracle design, it is clear that the two energy curves, of BoFL and Oracle, almost coincide in the exploitation phase. The major energy overhead comes from phase 1 and 2 when BoFL focuses on the exploration but not energy optimization. Note that in practice, a FL model may take 500 $\sim$ 10000 rounds to converge [37]. An exploration phase of around 10 rounds generates negligible overhead comparing to that of the dominantly long exploitation phase.

Overall, BoFL reduces energy consumption by 22.3% compared with Performant and generates 3.48% energy overhead compared to Oracle for the experiment as shown in figure 9a. Similar results can be observed from all remaining plots in figure 9 and 10. Comparing figure 9 and figure 10, it is clear the longer deadlines reduce the spikes in the energy curves, as it provides more space to pace down for energy optimization. We will present more results to showcase how deadline length influences the performance of BoFL in §6.4.

6.3 BoFL Pareto Construction

In figure 11, we present the Pareto front constructed by BoFL (as shown in blue squares), as well as the actual Pareto front derived from offline profiling (as shown in red stars). It is clear that BoFL can successfully find a close approximation to the actual Pareto front over all three tasks. We further plot all remaining BoFL explorations in blue circles, and the unexplored configurations in gray dots. As the figure shows, BoFL makes most of its explorations around the Pareto front while seldom trials in the less performant area, because our solution can smartly search the configuration space with Bayesian optimized suggestions and skip a lot of sub-optimal points. Note that we only plot a small part of the whole configuration space around the Pareto front for presentation clarity. There are much more sub-optimal and unexplored points than those as shown in figure 11. In our experiments, the Pareto front can be efficiently constructed after exploring just 3% of the whole configuration space.

We further present a walkthrough example of how BoFL explores the space and searches for the Pareto optimal configurations, as shown in table 3. For the *CIFAR10-ViT* task, BoFL first randomly searches the whole configuration space for 4 rounds and explores 21 different configurations. After that, BoFL switches into the Pareto construction phase and starts taking exploration suggestions from the MBO module. The second phase continues for 6 rounds and explores 49 points before the ending criteria (§4.3) is satisfied. During the whole exploration process, 70 configurations are explored, in which 20 of them constitute the Pareto front, i.e., the blue squares as shown in figure 11a. It can be easily observed that most of Pareto

	CIFAR10-ViT		ImageNet-ResNet50		IMDB-LSTM	
Round	# Exp	# Pareto	# Exp	# Pareto	# Exp	# Pareto
1	9	1	17	2	16	1
2	2	0	4	0	5	0
3	8	1	10	2	5	0
4	2	0	10	0	10	3
5	10	1	10	5	10	4
6	6	4	7	2	10	3
7	10	2	10	2	10	3
8	7	2				
9	8	6				
10	8	3				
Total	70	20	68	13	66	14

Table 3: The number of Explorations and searched Pareto points for each round in the first two phases. Red numbers are for the safe random exploration phase. Blue numbers are for the Pareto construction phase. E.g., In the first round of CIFAR10-ViT task, BoFL is in the random safe exploration phase. It explores 9 configurations with one of them being in the ultimate Pareto front.

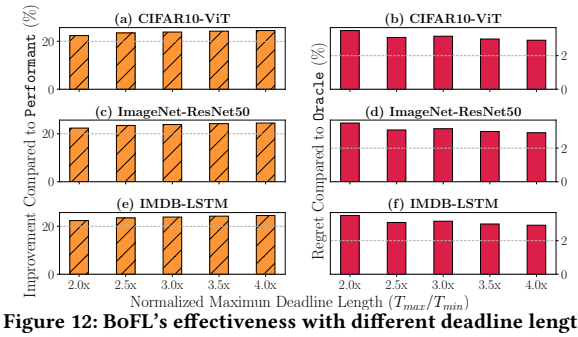
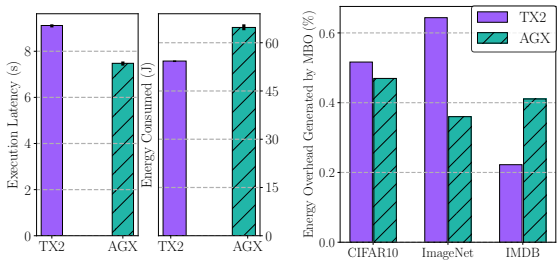


Figure 12: BoFL's effectiveness with different deadline length.



(a) MBO overhead per round. (b) Overall energy overhead.
Figure 13: Overhead of the MBO module

front points, i.e. 18 out of 20, are searched in the second phase, because the Bayesian optimization algorithm focuses on exploring more promising regions and can search for Pareto front more efficiently compared to random sampling. Similar patterns can also be observed in the other two tasks, as shown in table 3.

6.4 Sensitivity to Deadline Length

We evaluate how deadline length influences BoFL's effectiveness as follows. We run 6 set of experiments with different ranges of deadline length, i.e., $\mathbf{T}_{max}/\ \mathbf{T}_{min} \in [2, 4]$, as shown in table 2, measuring the following two metrics to evaluate BoFL's performance:

(1) **Improvement compared to Performant**: A metric represents how much energy can be reduced by BoFL compared to Performant, i.e., $1 - \frac{\text{BoFL energy usgae}}{\text{Performant energy usgae}}$.
(2) **Regret compared to Oracle**. A metric represents how much energy overhead is generated by BoFL compared to Oracle, i.e., $\frac{\text{BoFL energy usgae}}{\text{Oracle energy usgae}} - 1$.

The evaluation results are presented in figure 12. As the figure shows, BoFL's improvement compared to Performant steadily increases as the FL tasks are assigned with longer deadlines. As the deadlines are getting longer, BoFL has larger optimization space for pace control, it can train the model slower to benefit from less energy consumption, which explains the increasing curve. As the deadlines are getting longer, BoFL's strategy will gradually converge to choosing the most energy-efficient configuration, i.e., the bottom point in the Pareto front as shown in figure 11, which makes the overall energy consumption stable. Overall, BoFL can reduce energy consumption by 20.3% $\sim$ 25.9% compared to the Performant baseline.

As the deadline increases, BoFL's regret compared to Oracle steadily decreases. When the deadlines are short, BoFL tends to spend more time in on exploration, which generates more energy regret. As shown in figure 9a and figure 10a for *CIFAR10-ViT* task, BoFL explores 10 rounds before exploitation when $\frac{\mathbf{T}_{max}}{\mathbf{T}_{min}} = 2$, while only explores 6 rounds when $\frac{\mathbf{T}_{max}}{\mathbf{T}_{min}} = 4$. The reason under the hood is that longer deadlines allow BoFL to explore more configurations in a single round, which leads to a well constructed Pareto within fewer rounds. Overall, BoFL generates energy consumption regret by 1.2% $\sim$ 3.4% compared to the Oracle target.

6.5 MBO Module Overhead

In BoFL, we use MBO to smartly generate the next-step exploration points. However, calculating these suggestions does not come for free. In the rest of this section, we evaluate the overhead of the MBO module in terms of two metrics: (1) MBO calculation latency, and (2) MBO energy consumption. We present the evaluation results in figure 13.

As shown in figure 13a, the MBO module may take 6 $\sim$ 9 seconds to update its estimation and generate next-step suggestions. Note that the MBO calculation happens outside the time critical model training and introduces zero overhead to the FL workloads. In practice, the Bayesian optimization calculation can be co-located when the client is sending or receiving models from the server, which usually takes 10 $\sim$ 20 of seconds[6], and is long enough for the MBO calculation. For weak devices that takes longer time to update their Bayesian models, we can always speed up the MBO calculation by reducing its suggestion batch size.

As figure 13a depicts, one round of MBO calculation usually consumes 50 $\sim$ 70 Joule energy, which is much smaller than that of model training, which usually takes 600 $\sim$ 1200 Joule, as shown in figure 9 and 10. As MBO only happens a few times during the Pareto construction phase, the overall energy overhead generated by MBO is as small as 0.4% $\sim$ 0.7%, as presented in figure 13b.

7 RELATED WORK

Federated learning. Federated learning brings collaborative intelligence into multiple domains, including health care [18, 62, 75], smart transportation [49, 67], localization service [19, 80], recommendation system [24, 64] and beyond. Significant work has been proposed to improve the performance of FL in multiple perspectives, which contains but is not limited to model optimization [47, 56, 74], privacy preserving[12, 61], and model personalization [53, 73]. In this paper, we present BoFL to improve FL from an energy efficient perspective.

Dynamic voltage frequency scaling. Dynamic voltage frequency scaling (DVFS) is a widely applied technique for managing power and performance of processors, such as CPU and GPU. A large amount of research has been shown to achieve better energy efficiency with the benefit of DVFS [29, 34, 59, 60, 70]. In this paper, we design a

[6]E.g., sending and receiving ResNet50 model may take $51.2Mb/5Mbps = 10.2s$ for model transmission under 4G LTE network.

model training pace controller based on multi-axes DVFS to minimize the energy consumption of edge devices during federated learning tasks.

Bayesian optimization and its applications. Bayesian optimization (BO) is a sample-efficient optimization framework for many black-box functions that are expensive to evaluate. BO has been applied across multiple fields, i.e., A/B Testing [17, 41], robotics motion planing [50, 54], drug discovery [63], cherry picking cloud configurations [11], and more recently as an important ingredient for neural architecture search [36, 38, 72], which automates the process of neural network design. CherryPick [11] is the first work applying BO on system configuration selection for cloud clusters. It applies single-object BO to minimize the overall operational cost. In this work, we apply BO to efficiently search the vast DVFS design space for energy-aware federated learning. Compared to CherryPick, our problem is more challenging due to the multi-dimensional optimization targets, i.e., a joint minimization of latency and energy consumption.

8 CONCLUSION

In this work, we present BoFL, a local training pace controller for edge devices to achieve energy efficient federated learning. Experiments on multiple edge devices over multiple neural network models show that BoFL can reduce energy consumption of model training by more than 20% compared to the Performant baseline, and achieve close to optimal energy efficient compared to the Oracle target with as low as 1.2% - 3.4% energy regret.

ACKNOWLEDGMENT

This work was supported by IBM-Illinois Discovery Accelerator Institute.

REFERENCES

[1] [n.d.]. Apache MXNET. https://https://mxnet.apache.org/versions/1.9.0/. Accessed: 2022-05-08.
[2] [n.d.]. Gurobi-The fastest solver. https://www.gurobi.com/. Accessed: 2022-05-08.
[3] [n.d.]. INA3221 Texas Instruments. https://www.ti.com/product/INA3221. Accessed: 2022-05-08.
[4] [n.d.]. Jetson AGX Xavier Developer Kit. https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit. Accessed: 2022-05-08.
[5] [n.d.]. Jetson TX2 Module. https://developer.nvidia.com/embedded/jetson-tx2. Accessed: 2022-05-08.
[6] [n.d.]. Pytorch. https://pytorch.org. Accessed: 2022-05-08.
[7] [n.d.]. Secondmind-Trieste Document. https://secondmind-labs.github.io/trieste/index.html. Accessed: 2022-05-08.
[8] [n.d.]. Tensorflow. https://www.tensorflow.org. Accessed: 2022-05-08.
[9] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.
[10] Abdullatif Albaseer, Bekir Sait Ciftler, Mohamed Abdallah, and Ala Al-Fuqaha. 2020. Exploiting unlabeled data in smart cities using federated edge learning. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 1666–1671.
[11] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. 2017. {CherryPick}: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 469–482.
[12] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. 2018. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984* (2018).
[13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 387–402.
[14] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).
[15] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems* 30 (2017).
[16] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems* 1 (2019), 374–388.
[17] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. *Advances in neural information processing systems* 24 (2011).
[18] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. 2020. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems* 35, 4 (2020), 83–93.
[19] Bekir Sait Ciftler, Abdullatif Albaseer, Noureddine Lasla, and Mohamed Abdallah. 2020. Federated learning for localization: A privacy-preserving crowdsourcing method. *arXiv preprint arXiv:2001.01911* (2020).
[20] Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. 2021. Multi-objective bayesian optimization over high-dimensional search spaces. *arXiv preprint arXiv:2109.10964* (2021).
[21] Ittai Dayan, Holger R Roth, Aoxiao Zhong, Ahmed Harouni, Amilcare Gentili, Anas Z Abidin, Andrew Liu, Anthony Beardsworth Costa, Bradford J Wood, Chien-Sung Tsai, et al. 2021. Federated learning for predicting clinical outcomes in patients with COVID-19. *Nature medicine* 27, 10 (2021), 1735–1743.
[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
[24] Sijing Duan, Deyu Zhang, Yanbo Wang, Lingxiang Li, and Yaoxue Zhang. 2019. JointRec: A deep-learning-based joint cloud video recommendation framework for mobile IoT. *IEEE Internet of Things Journal* 7, 3 (2019), 1655–1666.
[25] Jie Feng, Can Rong, Funing Sun, Diansheng Guo, and Yong Li. 2020. PMF: A privacy-preserving human mobility prediction framework via federated learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 1 (2020), 1–21.
[26] Peter I Frazier. 2018. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*. Informs, 255–278.
[27] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. 2010. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*. Springer, 131–162.
[28] Stewart Greenhill, Santu Rana, Sunil Gupta, Pratibha Vellanki, and Svetha Venkatesh. 2020. Bayesian optimization for adaptive experimental design: A review. *IEEE access* 8 (2020), 13937–13948.
[29] João Guerreiro, Aleksandar Ilic, Nuno Roma, and Pedro Tomás. 2019. DVFS-aware application classification to improve GPGPUs energy efficiency. *Parallel Comput.* 83 (2019), 93–117.
[30] Hongpeng Guo, Shuochao Yao, Zhe Yang, Qian Zhou, and Klara Nahrstedt. 2021. CrossRoI: cross-camera region of interest optimization for efficient real time video analytics at scale. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 186–199.
[31] Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. 2018. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*. PMLR, 1929–1938.
[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*. Springer, 630–645.
[33] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[34] Qing Jiao, Mian Lu, Huynh Phung Huynh, and Tulika Mitra. 2015. Improving GPGPU energy-efficiency through concurrent kernel execution and DVFS. In *2015 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 1–11.
[35] Amelia Jiménez-Sánchez, Mickael Tardy, Miguel A González Ballester, Diana Mateus, and Gemma Piella. 2021. Memory-aware curriculum federated learning for breast cancer classification. *arXiv preprint arXiv:2107.02504* (2021).
[36] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1946–1956.
[37] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[38] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. 2018. Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems* 31 (2018).

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

[40] 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*. 183–198.

[41] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M Henne. 2009. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery* 18, 1 (2009), 140–181.

[42] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. [n.d.]. CIFAR-10 (Canadian Institute for Advanced Research). ([n. d.]). http://www.cs.toronto.edu/~kriz/cifar.html

[43] Fan Lai, Yinwei Dai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. FedScale: Benchmarking model and system performance of federated learning. In *Proceedings of the First Workshop on Systems Challenges in Reliable and Secure Federated Learning*. 1–3.

[44] Li Li, Haoyi Xiong, Zhishan Guo, Jun Wang, and Cheng-Zhong Xu. 2019. SmartPC: Hierarchical Pace Control in Real-Time Federated Learning System. In *2019 IEEE Real-Time Systems Symposium (RTSS)*. 406–418.

[45] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* 2 (2020), 429–450.

[46] Wenqi Li, Fausto Milletarì, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M Jorge Cardoso, et al. 2019. Privacy-preserving federated brain tumour segmentation. In *International workshop on machine learning in medical imaging*. Springer, 133–141.

[47] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[48] Xiang Li, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. Communication efficient decentralized training with multiple local updates. *arXiv preprint arXiv:1910.09126* 5 (2019).

[49] Yi Liu, JQ James, Jiawen Kang, Dusit Niyato, and Shuyu Zhang. 2020. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet of Things Journal* 7, 8 (2020), 7751–7763.

[50] Daniel J Lizotte, Tao Wang, Michael H Bowling, Dale Schuurmans, et al. 2007. Automatic Gait Optimization With Gaussian Process Regression.. In *IJCAI*, Vol. 7. 944–949.

[51] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 142–150.

[52] Derek C Manheim and Russell L Detwiler. 2019. Accurate and reliable estimation of kinetic parameters for environmental engineering applications: A global, multi objective, Bayesian optimization approach. *MethodsX* 6 (2019), 1398–1414.

[53] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619* (2020).

[54] Ruben Martinez-Cantin, Nando De Freitas, Eric Brochu, José Castellanos, and Arnaud Doucet. 2009. A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots* 27, 2 (2009), 93–103.

[55] Alexandre Mathern, Olof Skogby Steinholtz, Anders Sjöberg, Magnus Önnheim, Kristine Ek, Rasmus Rempling, Emil Gustavsson, and Mats Jirstrand. 2021. Multi-objective constrained Bayesian optimization for structural design. *Structural and Multidisciplinary Optimization* 63, 2 (2021), 689–701.

[56] 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*. PMLR, 1273–1282.

[57] H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. 2018. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210* (2018).

[58] David R Morrison, Sheldon H Jacobson, Jason J Sauppe, and Edward C Sewell. 2016. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization* 19 (2016), 79–102.

[59] Seyed Morteza Nabavinejad, Hassan Hafez-Kolahi, and Sherief Reda. 2019. Coordinated DVFS and precision control for deep neural networks. *IEEE Computer Architecture Letters* 18, 2 (2019), 136–140.

[60] Seyed Morteza Nabavinejad, Sherief Reda, and Masoumeh Ebrahimi. 2022. Coordinated Batching and DVFS for DNN Inference on GPU Accelerators. *IEEE Transactions on Parallel and Distributed Systems* (2022).

[61] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 739–753.

[62] Stephen R Pfohl, Andrew M Dai, and Katherine Heller. 2019. Federated and differentially private learning for electronic health records. *arXiv preprint arXiv:1911.05861* (2019).

[63] Edward O Pyzer-Knapp. 2018. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development* 62, 6 (2018), 2–1.

[64] Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2020. Privacy-preserving news recommendation model learning. *arXiv preprint arXiv:2003.09592* (2020).

[65] Carl Edward Rasmussen. 2003. Gaussian Processes in Machine Learning. In *Advanced Lectures on Machine Learning*. Springer, 63–71.

[66] Ryan Roussel, Adi Hanuka, and Auralee Edelen. 2021. Multiobjective Bayesian optimization for online accelerator tuning. *Physical Review Accelerators and Beams* 24, 6 (2021), 062801.

[67] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Mérouane Debbah. 2019. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications* 68, 2 (2019), 1146–1159.

[68] Jaemin Shin, Yuanchun Li, Yunxin Liu, and Sung-Ju Lee. 2022. Sample Selection with Deadline Control for Efficient Federated Learning on Heterogeneous Clients. *arXiv preprint arXiv:2201.01601* (2022).

[69] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, Vol. 25. Curran Associates, Inc.

[70] Zhenheng Tang, Yuxin Wang, Qiang Wang, and Xiaowen Chu. 2019. The impact of GPU DVFS on the energy and performance of deep learning: An empirical study. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*. 315–325.

[71] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. 2021. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917* (2021).

[72] Colin White, Willie Neiswanger, and Yash Savani. 2019. Bananas: Bayesian optimization with neural architectures for neural architecture search. *arXiv preprint arXiv:1910.11858* 1, 2 (2019), 4.

[73] Qiong Wu, Kaiwen He, and Xu Chen. 2020. Personalized federated learning for intelligent IoT applications: A cloud-edge based framework. *IEEE Open Journal of the Computer Society* 1 (2020), 35–44.

[74] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).

[75] Zhaoping Xiong, Ziqiang Cheng, Chi Xu, Xinyuan Lin, Xiaohong Liu, Dingyan Wang, Xiaomin Luo, Yong Zhang, Nan Qiao, Mingyue Zheng, et al. 2020. Facing small and biased data dilemma in drug discovery with federated learning. *BioRxiv* (2020).

[76] Kaifeng Yang, Michael Emmerich, André Deutz, and Thomas Bäck. 2019. Multi-objective Bayesian global optimization using expected hypervolume improvement gradient. *Swarm and evolutionary computation* 44 (2019), 945–956.

[77] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).

[78] 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* 20, 3 (2020), 1935–1949.

[79] Zhe Yang, Klara Nahrstedt, Hongpeng Guo, and Qian Zhou. 2021. Deeprt: A soft real time scheduler for computer vision applications on the edge. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 271–284.

[80] Feng Yin, Zhidi Lin, Qinglei Kong, Yue Xu, Deshi Li, Sergios Theodoridis, and Shuguang Robert Cui. 2020. FedLoc: Federated learning framework for data-driven cooperative localization and location data processing. *IEEE Open Journal of Signal Processing* 1 (2020), 187–215.

[81] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. 2017. Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics*. PMLR, 962–970.

[82] Marcela Zuluaga, Guillaume Sergent, Andreas Krause, and Markus Püschel. 2013. Active learning for multi-objective optimization. In *International Conference on Machine Learning*. PMLR, 462–470.