

Chapter

# Federated Learning Hyper-Parameter Tuning for Edge Computing

*Xueying Zhang, Lei Fu, Huanle Zhang and Xin Liu*

## Abstract

Edge computing is widely recognized as a crucial technology for the upcoming generation of communication networks and has garnered significant interest from both industry and academia. Compared to other offloading models like cloud computing, it provides faster data processing capabilities, enhanced security measures, and lower costs by leveraging the proximity of the edge servers to the end devices. This helps mitigate the privacy concerns associated with data transfer in edge computing, by reducing the distance between the data source and the server. Raw data in typical edge computing scenarios still need to be sent to the edge server, leading to data leakage and privacy breaches. Federated Learning (FL) is a distributed model training paradigm that preserves end devices' data privacy. Therefore, it is crucial to incorporate FL into edge computing to protect data privacy. However, the high training overhead of FL makes it impractical for edge computing. In this study, we propose to facilitate the integration of FL and edge computing by optimizing FL hyper-parameters, which can significantly reduce FL's training overhead and make it more affordable for edge computing.

**Keywords:** edge computing, federated learning, hyper-parameter tuning, system overhead, internet of things

## 1. Introduction

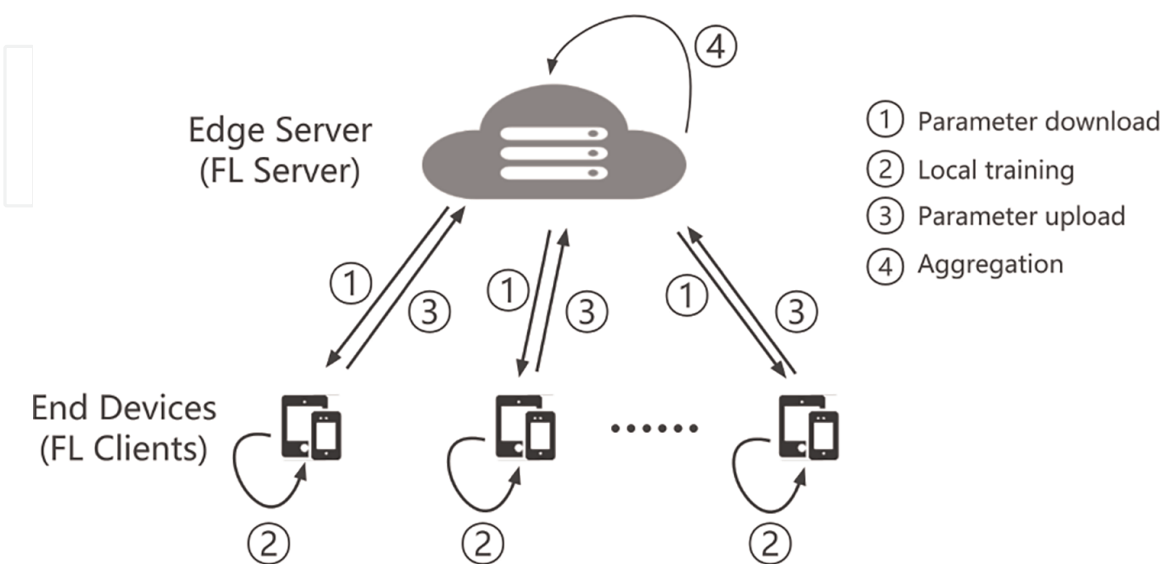
As machine learning (ML) and hardware manufacturing technologies continue to advance, training and deploying ML models have become increasingly ubiquitous in our daily lives, from smart-home voice assistants to widely deployed camera surveillance systems. Edge computing is becoming more and more popular due to its advantages, such as fast data processing and analysis, security, and low cost [1]. By placing the edge servers near to the end device, which is the fundamental principle of edge computing, the border of an edge computing system is constrained and manageable.

However, even with the shorter distance between the end device and the edge server, typical edge computing systems still suffer from a significant data privacy issue, as user data is frequently transmitted from the end device to the edge server for training a centralized ML model.

Federated Learning (FL) [2] is a method of model training that is distributed and has been utilized in various applications, including mobile keyboard and speech recognition for mobile devices and IoT. It is naturally suited for edge computing since data is kept on the end devices. **Figure 1** illustrates the combination of FL and edge computing in training a distributed model. First, the model parameters are transferred from the edge server to the end device. After that, the end device trains the model locally and then transfers the model parameters from the end device to the edge server. At the end of this iteration, the edge server aggregates the received model parameters and updates the model parameters. The above procedure will be repeated until the entire training process converges or reaches a predetermined number of epochs.

Unfortunately, FL training incurs significant system overhead, making it difficult for edge computing systems equipped with FL to operate without appropriate acceleration or optimization. Therefore, we propose the integration of FL hyper-parameter tuning in edge computing to reduce the system overhead of FL training and make it more feasible. The FL tuning algorithm should focus on optimizing the four essential system overheads:

- *Computation Time (CompT)*. It measures the time spent by an FL system in model training. When confronted with application scenarios that need a rapid reaction to environmental changes (e.g., when dealing with security issues), the overall model training period must be short.
- *Transmission Time (TransT)*. It represents how long an FL system spends in model parameter transmission between clients and servers. For applications in poor network environments, the transfer of the model should be as fast as possible.
- *Computation Load (CompL)*. It is the number of Floating-Point Operation (FLOP) that an FL system consumes. For low-profile devices, a large



**Figure 1.** An illustration of combining FL with edge computing. The model training process incorporates four steps: Model parameter download from the edge server to the end devices, local training on the end devices, model parameter upload from the end devices to the edge server, and model aggregation on the edge server.

computing load is beyond the reach of some low-profile devices (e.g., IoT nodes) with few computing resources.

- *Transmission Load (TransL)*. It is the total data size transmitted between the clients and the server. If the cost of data transfer is high (e.g., data transfer is expensive), the benefits of reducing the total amount of data transferred can be considerable.

Different application scenarios have distinct preferences for training parameters in terms of CompT, TransT, CompL, and TransL. For example, (1) detecting attacks and anomalies in computer networks, as shown in Ref. [3], requires quick adaptation to malicious traffic and is therefore time-sensitive (CompT and TransT); (2) smart home control systems for indoor environment automation [4], such as HVAC, have limited computation capabilities and therefore prioritize computation efficiency (CompT and CompL); (3) traffic monitoring systems for vehicles [5] rely on cellular communications and therefore emphasize communication efficiency (TransT and TransL); (4) precision agriculture based on IoT sensing [6] does not require urgent response but necessitates energy-efficient solutions, with emphasis on CompL and TransL; (5) healthcare systems, like fall detection for elderly individuals [7], require both quick response time and small energy consumption, and therefore prioritize all four training parameters (CompT, TransT, CompL, and TransL); and (6) human stampede detection/prevention systems, as discussed in [8], need efficient systems for time, computation, and communication.

In this chapter, we explore the problem of supporting FL in edge computing from the perspective of FL hyper-parameter tuning. FL hyper-parameters significantly affect the system overhead of FL training, and thus, optimizing FL hyper-parameters is greatly valuable for resource-constrained edge computing. We organize this chapter as follows. Section 2 provides related work on edge computing and FL hyper-parameter tuning. Section 3 explains the challenges of supporting FL in edge computing, and Section 4 presents some preliminary results. Last, Section 5 concludes this chapter.

## 2. Related work

In this section, we provide related work with regard to edge computing and FL hyper-parameter tuning.

### 2.1 Edge computing

With the fast expansion of IoT, more smart devices are connected to the Internet, producing significant amounts of data. The device-generated data causes bandwidth and latency problems when it is sent to a centralized data center or the cloud. Due to this, typical cloud computing models experience problems such as bandwidth usage, slow reaction times, insufficient security, and poor privacy. Moreover, the growing amount of data also puts more strain on servers and drives up operating costs.

Edge computing solutions have evolved as a result of the fact that traditional cloud computing is no longer able to serve the diversified data processing demands of today's intelligent society. In simplest terms, edge computing is a network technology that analyzes data collected from an endpoint directly in a local device or network close to

where the data is generated, without sending the data to a cloud-based data processing facility. Its core idea is to make computing closer to the source of the data [9].

Edge computing has several advantages. (1) **Low latency**: Since edge computing is closer to the data source, data storage and computational operations may be performed in the edge computing node, reducing the intermediate data transmission process. Therefore, service providers can process user requests in real time and allow users to experience low-latency services. (2) **Low bandwidth**: In edge computing, as the data to be processed do not need to be uploaded to a cloud computing centre, it does not need to use too much network bandwidth, therefore reducing the network bandwidth load and significantly reducing the energy consumption of intelligent devices at the edge of the network. (3) **Privacy**: Since the edge nodes are only responsible for tasks within their own scope and do not need to upload data to the cloud, network transmission concerns are avoided. Even if one of the edge nodes suffers a data breach as a result of a network attack, the other edge nodes will not be affected. Edge computing significantly secures data.

However, although edge computing protects user data privacy better than traditional cloud computing, it is inevitable that users will upload some or all of their personal information to edge servers, such as cloud data centers or edge data centers. These core infrastructures may be managed by the same third-party suppliers, such as mobile network operators, that may not be trusted. Data is exposed to data security issues such as data leakage and data loss during transmission. Also, personal private data may be used illegally by application providers. Thus, the security of outsourcing data is still a fundamental problem of edge computing data security [10].

## 2.2 FL hyper-parameter tuning

The area of Hyper-Parameter Optimization (HPO) has received a lot of attention [11]. The hyper-parameters of machine learning models are optimized using a variety of classical HPO techniques, such as Bayesian Optimization (BO) [12], successive halving [13], and hyperband [14]. These cannot, however, be directly applied to FL due to FL's unique hyper-parameters and different training paradigms. For example, FL has specific client-side and server-side aggregation methods that need to be optimized, and the data remains on end devices rather than being centralized on a server.

Work	Description	Single trial	System
FTS [15]	Optimize client models	✗	✗
Zhiyuan et al. [16]	PSO-based optimization	✗	✗
DP-FTS-DE [17]	Trade-off privacy and utility	✓	✗
Auto-FedRL [18]	Improve model accuracy	✓	✗
[19]	Improve training robustness	✓	✗
FedEx [20]	NAS-based framework	✗	✗
FLoRA [21]	NAS-based framework	✓	✗
FedTune [22]	A simple framework	✓	✓

**Table 1.**

*Related work on FL hyper-parameter optimization. We tag if (1) the work can run in an online and single trail manner and (2) the work targets system overheads of FL training.*



Designing HPO algorithms for FL is an emerging area of research. In the past studies, several methods have touched the field of FL HPO. **Table 1** provides an overview of various notable methods, indicating whether they can operate in a single-trial and online manner, and whether they address system overhead concerns in FL training.

For instance, BO has been combined with FL to strengthen client privacy [17] and enhance various client models [15]; Zhiyuan et al. utilized particle swarm optimization (PSO) to expedite the exploration process of FL hyper-parameters [16]. However, this approach lacked support for single-trial and system overhead. Multiple methods utilize reinforcement learning to fine-tune FL hyper-parameters [18, 19], but this leads to additional intricacy and reduced versatility. FedEx is a comprehensive framework that utilizes weight-sharing neural architecture search (NAS) techniques to optimize the round-to-accuracy of FL. This approach enhances the baseline by a few percentage points [20]. FLoRA chooses global hyper-parameters by identifying the ones that exhibit high performance in local clients [21]. Although a benchmark suite for optimizing federated hyper-parameters has been created [23], its efficacy has not been evaluated yet. FedTune suggests a basic framework for tuning FL hyper-parameters based on the specific requirements of an application [22]. There are two reasons why the current approaches are not applicable to the problem of federated learning in edge computing. Firstly, the measures such as CompT (in seconds), TransL (in seconds), CompL (in FLOPs), and TransL (in bytes) are not directly comparable, and incorporating various system factors in optimizing HPO is challenging. Secondly, hyper-parameter tuning must occur simultaneously with FL training, and there is no possibility of revisiting the model as the training continues until the final model accuracy is reached. Otherwise, this would lead to a substantial rise in the system's overhead.

### 3. Challenges of supporting FL in edge computing

In edge computing, a multitude of end devices with varying hardware and data are connected through an edge server, resulting in heterogeneous end devices. This heterogeneity in both system and data poses several challenges when integrating federated learning with edge computing.

#### 3.1 System heterogeneity

The end devices typically possess a range of distinct hardware, which can vary in their capabilities regarding computation, communication, energy, and other factors.

- *Computation Capability.* Due to the increasing demand for gaming and AI applications, many end devices are now equipped with AI accelerators, such as GPU, NPU, or CUDA cores. Nevertheless, tests on popular end devices reveal that their running times for AI models can vary by a factor of tens or more [24]. This difference in time is even more pronounced when the AI models cannot fit into the memory of the AI accelerators, or if the AI model operators are not compatible with the end devices [25].
- *Communication Capability.* In federated learning, the speed of transmission plays a crucial role since it involves multiple rounds of model parameter transmission between the end devices and the edge server. However, since end devices can be equipped with different transmission standards (LTE vs. WiFi), be situated in varying

locations (indoor vs. outdoor), and encounter different wireless channel conditions (congested vs. clear), their transmission speeds can differ greatly. By analyzing hundreds of end devices in a real-world FL deployment [26], it has been observed that there is a substantial order-of-magnitude difference in the network bandwidth [27].

- *Other Factors.* Apart from computation and communication capabilities, the availability and capability of end devices are influenced by many other factors. For instance, when the battery of an end device is low, its computation and communication capabilities are reduced to conserve power. Furthermore, end devices running heavy applications in the background can substantially limit the available computing resources.

### 3.2 Statistical heterogeneity

End devices in edge computing possess distinct characteristics in terms of their data properties, including massively distributed data, unbalanced data, and non-Independent and Identically Distributed (IID) data [2].

- *Massively Distributed Data:* The number of end devices in edge computing is typically much larger than the average number of data points per end device. For instance, in the Google keyboard query suggestion project [26], there are millions of smartphones involved, but an individual user typically generates only dozens of queries per day.
- *Unbalanced Data:* The local data size on end devices varies significantly due to different usage patterns. For instance, the Reddit comment dataset [28] demonstrates that 70% of users contribute to the first quarter of the normalized number of comments, whereas 10% of users generate three times more comments than the average user [27].
- *Non-IID Data:* The data on each end device is not a representative sample of the overall distribution of data, as it does not follow the Independent and Identically Distributed (IID) property. This non-IID property is commonly found in real-world scenarios [29], and it significantly impacts the training of FL models due to the presence of attribute and label skew [30].
- Traditionally, edge computing research has concentrated on examining a limited number of end devices and a basic edge server. Nonetheless, in order to facilitate Federated Learning, a comprehensive understanding of numerous end devices and their interdependent effects on the overall machine learning training process is necessary. Consequently, developing an edge computing system that is compatible with FL is more difficult than creating a cross-device FL system.

## 4. FL hyper-parameter tuning for edge computing

At present, there is no *de facto* method for incorporating FL into edge computing. We propose the use of automated tuning of FL hyper-parameters as a means to decrease the system overhead associated with FL training. The possibilities of adjusting FL hyper-parameters to minimize the system overhead of FL training are

becoming more apparent. In this section, we use our preliminary work, called FedTune [22], to clarify the potential value of FL hyper-parameter tuning for edge computing. FedTune takes into account the application’s prioritization for CompT, TransT, CompL, and TransL, which are represented by  $\alpha$ ,  $\beta$ ,  $\gamma$ , and  $\delta$ , respectively. We have  $\alpha + \beta + \gamma + \delta = 1$ . For instance, if we take  $\alpha = 0.6$ ,  $\beta = 0.2$ ,  $\gamma = 0.1$ , and  $\delta = 0.1$ , it means that the application gives the highest priority to CompT, some importance to TransT, and least importance to CompL and TransL. For two sets of FL hyper-parameters  $S_1$  and  $S_2$ , FedTune defines the comparison function  $I(S_1, S_2)$  as

$$I(S_1, S_2) = \alpha \times \frac{t_2 - t_1}{t_1} + \beta \times \frac{q_2 - q_1}{q_1} + \gamma \times \frac{z_2 - z_1}{z_1} + \delta \times \frac{v_2 - v_1}{v_1} \quad (1)$$

where  $t_1$  and  $t_2$  are CompT for  $S_1$  and  $S_2$  when achieving the same model accuracy. Similarly,  $q_1$  and  $q_2$  denote TransT,  $z_1$  and  $z_2$  represent CompL, and  $v_1$  and  $v_2$  indicate TransL for  $S_1$  and  $S_2$ , respectively. If  $I(S_1, S_2) < 0$ , then  $S_2$  is better than  $S_1$ . A set of hyper-parameters is better than another set if the weighted improvement of some training aspects (e.g., CompT and CompL) is higher than the weighted degradation, if any, of the remaining training aspects (e.g., TransT and TransL). The weights assigned to each aspect are determined by the application’s training preferences on CompT, TransT, CompL, and TransL.

FedTune utilizes an iterative algorithm to update the hyper-parameters for the next round (refer to [22] for more details). This process is triggered only when the model accuracy has improved by a minimum amount of  $\epsilon$ . After normalizing the current overheads, FedTune computes the comparison function between the previous hyper-parameters  $S_{prv}$  and the current hyper-parameters  $S_{cur}$ . It then updates the hyper-parameters and resumes the training process. Due to its lightweight nature, FedTune has a minimal computational burden on a standard edge computing system.

The results obtained by FedTune are promising. The performance of FedTune for various datasets when FedAvg is employed is illustrated in **Table 2**. For the speech-to-command dataset and EMNIST dataset, the learning rate is set to 0.01, while for the Cifar-100 dataset, it is set to 0.1, all with a momentum of 0.9. The standard deviation is presented in parentheses. The results demonstrate that FedTune consistently enhances the overall performance for all three datasets. Specifically, by averaging 15 combinations of training preferences, FedTune reduces the system overhead of the speech-to-command dataset by 22.48% compared to the baseline. We have observed that FedTune is more beneficial for FL training when the convergence of the training process takes more training rounds. The performance of FedTune with various aggregation methods is presented in **Table 3** for the ResNet-10 model using the speech-to-command dataset. A learning rate of 0.1,  $\beta_1$  of 0, and  $\tau$  of 1e-3 were used for FedAdagrad. As shown, FedTune can improve the performance of the system when using different aggregation methods. Specifically, when using FedAdagrad, FedTune reduces the system overhead by 26.75%.

Dataset	Speech-command	EMNIST	Cifar-100
Data Feature	Voice	Handwriting	Image
ML Model	ResNet-10	2-layer MLP	ResNet-10
Performance	+22.48% (17.97%)	+8.48% (5.51%)	+9.33% (5.47%)

**Table 2.** Performance of FedTune for diverse datasets when FedAvg aggregation method is applied.

Aggregator	FedAvg	FedNova	FedAdagrad
Performance	+22.48% (17.97%)	+23.53% (6.64%)	<b>+26.75%</b> (6.10%)

**Table 3.**

Performance of FedTune for diverse aggregation algorithms. Speech-to-command dataset and ResNet-10 are used in this experiment.

## 5. Conclusion

Artificial intelligence is becoming increasingly important for enhancing people's quality of life and boosting productivity. Artificial intelligence is becoming increasingly important for enhancing people's quality of life and boosting productivity. The integration of edge computing with Federated Learning (FL) can help to tackle the data privacy issue. However, federated learning involves a significant amount of training overhead, which can be a challenge for resource-limited end devices. We propose a solution to reduce the system overhead of FL and make it more affordable to edge computing by automatically adjusting FL hyper-parameters. Our preliminary work has demonstrated promising results, with up to 26% reduction in system overhead. This suggests that FL hyper-parameter tuning is an effective approach for edge computing. However, further research is needed to fully support FL in edge computing, and more applications are required to drive the growth of the edge computing ecosystem.

## Acknowledgements

The work was partially supported by NSF through grants USDA/NIFA 2020-67021-32855, IIS-1838207, CNS 1901218, OIA-2134901. It was also partially supported by the Jiangsu Funding Program for Excellent Postdoctoral Talent (No. 2022ZB804).

## Author details

Xueying Zhang<sup>1</sup>, Lei Fu<sup>2</sup>, Huanle Zhang<sup>1\*</sup> and Xin Liu<sup>3</sup>


1 Shandong University, Qiangdao, China

2 Bank of Jiangsu and Fudan University, Shanghai, China

3 University of California, Davis, USA

\*Address all correspondence to: dtczhang@sdu.edu.cn

## IntechOpen

© 2023 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 



## References

- [1] Cao K, Liu Y, Meng G, Sun Q. An overview on edge computing research. *IEEE Access*. 2020;**8**:85714-85728
- [2] Brendan McMahan H, Moore DRE, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. New York, USA: PMLR; 2017. pp. 1-10
- [3] Haji SH, Ameen SY. Attack and anomaly detection in IoT networks using machine learning techniques: A review. *Asian Journal of Research in Computer Science (AJRCOS)*. 2021;**9**(2):30-46
- [4] Mekuria DN, Sernani P, Falcionelli N, Dragoni AF. Smart home reasoning systems: A systematic literature review. *Journal of Ambient Intelligence and Humanized Computing*. 2021;**12**: 4485-4502
- [5] Won M. Intelligent traffic monitoring Systems for Vehicle Classification: A survey. *IEEE Access*. 2020;**8**: 73340-73358
- [6] Sharma A, Jain A, Gupta P, Chowdary V. Machine learning applications for precision agriculture: A comprehensive review. *IEEE Access*. 2020;**9**:4843-4873
- [7] Hassan MM, Gumaei A, Aloï G, Fortino G, Zhou M. A smartphone-enabled fall detection framework for elderly people in connected home healthcare. *IEEE Access*. 2019;**33**:58-63
- [8] Maria Moitinho de Almeida and Johan von Schreeb. A smartphone-enabled fall detection framework for elderly people in connected home healthcare. *Prehospital and Disaster Medicine*. 2018; **34**:82-88
- [9] Satyanarayanan M. The emergence of edge computing. *Computer*. 2017;**50**(1): 30-39
- [10] Zhang J, Chen B, Zhao Y, Cheng X, Feng H. Data security and privacy-preserving in edge computing paradigm: Survey and open issues. *IEEE Access*. 2018;**6**:18209-18237
- [11] Yang L, Shami A. On Hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*. 2020; **415**:295-316
- [12] Snoek J, Larochelle H, Adams RP. Practical Bayesian optimization of machine learning algorithms. In: *International Conference on Neural Information Processing Systems (NIPS)*. Red Hook, NY, USA: Curran Associates Inc.; 2012
- [13] Karnin Z, Koren T, Somekh O. Almost optimal exploration in multi-armed bandits. In: *International Conference on Machine Learning (ICML)*. New York, USA: PMLR; 2013. pp. 1238-1246
- [14] Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A. Hyperband: A novel bandit-based approach to Hyperparameter optimization. *Journal of Machine Learning Research (JMLR)*. 2017;**18**:1-52
- [15] Dai Z, Low BKH, Jaillet P. Federated bayesian optimization via Thompson sampling. In: *Conference on Neural Information Processing Systems (NeurIPS)*. Red Hook, NY, USA: Curran Associates, Inc.; 2020
- [16] Li Z, Li H, Zhang M. Hyperparameter tuning of federated learning based on particle swarm optimization.

In: IEEE International Conference on Cloud Computing and Intelligent Systems (CCIS). Xi'an, China: IEEE; 2021

[17] Dai Z, Low BKH, Jaillet P. Differentially private federated bayesian optimization with distributed exploration. In: Conference on Neural Information Processing Systems (NeurIPS). Red Hook, NY, USA: Curran Associates, Inc.; 2021

[18] Guo P, Dong Y, Hatamizadeh A, Xu A, Xu Z, Li W, et al. Auto-FedRL: Federated Hyperparameter Optimization for Multi-Institutional Medical Image Segmentation. Cham: Springer; 2022. pp. 1-18

[19] Mostafa H. Robust federated learning through representation matching and adaptive Hyperparameters. arXiv. 2019;1:1-11

[20] Khodak M, Renbo T, Li T, Li L, Balcan M-F, Smith V, et al. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. In: Conference on Neural Information Processing Systems (NeurIPS). Red Hook, NY, USA: Curran Associates, Inc.; 2021

[21] Zhou Y, Ram P, Salonidis T, Baracaldo N, Samulowitz H, Ludwig H. FLoRA: Single-shot hyper-parameter optimization for federated learning. arXiv. 2021;1:1-11

[22] Zhang H, Zhang M, Liu X, Mohapatra P, DeLucia M. Fedtune: Automatic tuning of federated learning hyper-parameters from system perspective. In: IEEE Military Communications Conference (MILCOM). Rockville, MD, USA: IEEE; 2022

[23] Zhen WANG, Kuang W, Zhang C, Ding B, Li Y. FedHPO-B: A benchmark

suite for federated Hyperparameter optimization. arXiv. 2022;1:1-27

[24] Ignatov A, Timofte R, Kulik A, Yang S, Wang K, Baum F, et al. Ai benchmark: All about deep learning on smartphones in 2019. In: IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). Seoul, Korea (South): IEEE; 2019

[25] Zhang H, Han B, Mohapatra P. Toward mobile 3d vision. In: IEEE International Conference on Computer Communications and Networks (ICCCN). Honolulu, HI, USA: IEEE; 2020

[26] Yang T, Andrew G, Eichner H, Sun H, Li W, Kong N, et al. Applied federated learning: Improving google keyboard query suggestions. arXiv. 2018; 1:1-9

[27] Lai F, Zhu X, Madhyastha HV, Chowdhury M. Oort: Efficient federated learning via guided participant selection. In: USENIX Symposium on Operating Systems Design and Implementation (OSDI). USA: USENIX Association; 2021

[28] Reddit comment dataset. Available from: <https://files.pushshift.io/reddit/comments/> [Accessed: October 2022]

[29] He C, Li S, So J, Zeng X, Zhang M, Wang H, et al. Fedml: A research library and benchmark for federated machine learning. In: Conference on Neural Information Processing Systems (NeurIPS). Red Hook, NY, USA: Curran Associates, Inc.; 2020

[30] Zhu H, Jinjin X, Liu S, Jin Y. Federated learning on non-iid data: A survey. *Neurocomputing*. 2021;465: 371-390