This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

Training Latency Minimization for Model-Splitting Allowed Federated Edge Learning

Yao Wen, Guopeng Zhang, Kezhi Wang, and Kun Yang

Abstract—To alleviate the shortage of computing power faced by clients in training deep neural networks (DNNs) using federated learning (FL), we leverage the *edge computing* and *split learning* to propose a model-splitting allowed FL (SFL) framework, with the aim to minimize the training latency without loss of test accuracy. Under the *synchronized global update* setting, the latency to complete a round of global training is determined by the maximum latency for the clients to complete a local training session. Therefore, the training latency minimization problem (TLMP) is modelled as a minimizing-maximum problem. To solve this mixed integer nonlinear programming problem, we first propose a *regression method* to fit the quantitative-relationship between the *cut-layer* and other parameters of an AI-model, and thus, transform the TLMP into a continuous problem. Considering that the two subproblems involved in the TLMP, namely, the *cut-layer selection problem* for the clients and the *computing resource allocation problem* for the parameter-server are relative independence, an alternate-optimization-based algorithm with polynomial time complexity is developed to obtain a high-quality solution to the TLMP. Extensive experiments are performed on a popular DNN-model *EfficientNetV2* using dataset MNIST, and the results verify the validity and improved performance of the proposed SFL framework.

Index Terms—Federated learning, split learning, edge computing, computing task offloading, resource allocation.

1 Introduction

The latest Artificial Intelligence (AI) products leverage advanced machine learning (ML) techniques, ranging from face detection [1] and speech recognition [2] on mobile devices to virtual assistants in autonomous systems [3]. Large-scale data is critical for training high-performance AI models, such as decision trees, support vector machines (SVMs), and deep neural networks (DNNs). However, centralized training approaches require clients to transfer private data to servers, risking user privacy and security [4]. Federated learning (FL) [5], proposed by Google, enables multiple clients to train a shared AI model while keeping their data local. The vanilla FL algorithm, FedAvg, shown in Fig. 1, first distributes an AI model to clients. Each client trains the model locally, uploads it to the server for aggregation, and this process iterates until a certain test accuracy is achieved.

However, FedAvg is better suited for training lightweight models where communication costs exceed computational costs [5]. With the rise of 5G and the increasing scale of deep AI models (e.g., DNNs), FL's application landscape has shifted. Firstly, 5G's ultra-reliable and low-latency communication technologies have reduced communication costs in FL [6]. Secondly, despite the rapid growth in AI model complexity, the computing power of mobile devices has not kept pace [7]. These trends highlight the need for new techniques to address the challenges posed by large-scale models.

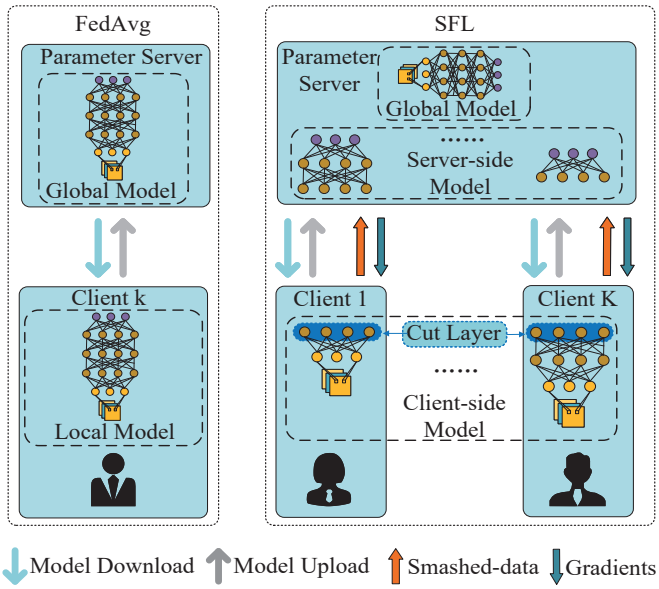


Fig. 1. The frameworks of FedAvg and SFL.

Split learning (SL) [8] supported by edge computing offers a solution. As shown in Fig. 1, split federated learning (SFL) divides the model into two parts: the client-side model and the server-side model. The client initiates forward propagation (FP) on the client-side model, generating "smashed-data," which it sends to the server. The server uses this data to continue FP on the server-side model and then performs backward propagation (BP), sending gradients back to the client to complete BP on the client-side model.

This paper was partly funded by Natural Science Foundation of China (No. 62132004), Jiangsu Major Project on Basic Researches (Grant No.: BK20243059) and Gusu Innovation Project for People (Grant No.: ZXL2024360).

- *Yao Wen and Guopeng Zhang are with the School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China. E-mail: ywen@cumt.edu.cn; gpzhang@cumt.edu.cn.*
- *Kezhi Wang is with the Department of Computer Science, Brunel University London, Middlesex UB8 3PH, U.K. E-mail: kezhi.wang@brunel.ac.uk.*
- *Kun Yang is with the State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, 210008, China, and School of Intelligent Software and Engineering, Nanjing University (Suzhou Campus), Suzhou, 215163, China. E-mail: kyang@ieee.org.*

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

By deploying high-performance computing servers, the PS can process multiple server-side models in parallel, reducing client computational burdens. However, unlike FedAvg, clients in SFL cannot independently perform FP and BP, resulting in multiple rounds of data transmission during model training. Although this suits the increased network bandwidth and model size, it introduces new challenges that must be addressed to optimize SFL.

Challenge 1: Cut-layer selection for clients. In SL, the "cut-layer" is the last layer of the client-side model [9], which impacts both communication and computational load. However, the relationship between the cut-layer and model parameters is not well-defined, creating a large solution space and high computational complexity for selecting the optimal cut-layer.

Challenge 2: Computing resource allocation for the PS. In SFL, the PS trains multiple server-side models from different clients. Due to client heterogeneity in computing power and datasets, the PS must efficiently allocate its limited resources to enhance training efficiency.

This paper uses edge computing and split learning to improve SFL training efficiency, aiming to minimize training latency using synchronized global model updates (SGMU) [10] while maintaining test accuracy. The training latency minimization problem (TLMP) is modeled as one that reduces the maximum latency among clients' local training sessions. To address *Challenge 1*, we use *EfficientNetV2* [11] as an example to propose a regression method that quantifies the relationship between the cut-layer and the resulting communication and computational loads. This transforms the original mixed-integer nonlinear programming (MINLP) problem into a continuous one. We then decouple the problem into two subproblems: cut-layer selection (to address *Challenge 1*) and computing resource allocation (to address *Challenge 2*). An alternate-optimization-based algorithm is proposed to find a high-quality solution to the TLMP. Experiments on the MNIST and CIFAR-10 datasets [12] validate the proposed method. In summary, the main contributions of this paper are as follows:

1) Edge computing and split learning techniques are integrated to improve SFL training efficiency, formulating the TLMP and minimizing training latency.
2) A regression method is introduced to relate the cut-layer selection to the communication and computational loads on clients, transforming the MINLP problem into a continuous one.
3) The TLMP is decoupled into two subproblems, and an alternate-optimization-based algorithm with polynomial time complexity is proposed to solve it. Extensive experiments validate the method.

The paper is organized as follows: Sec. 2 reviews related works, Sec. 3 details the SFL training procedure, Sec. 4 formulates the TLMP as a min-max problem, and Sec. 5 presents the algorithm to solve it. Experiment results are provided in Sec. 6, and conclusions are drawn in Sec. 7.

2 Related Works

Federated Learning: Federated Learning (FL) has been widely applied as a distributed machine learning method to protect privacy. The Parameter Server architecture [13] enables large-scale model training by aggregating updates from numerous computation nodes. Federated optimization [14] enhances communication efficiency by minimizing rounds while keeping data local. Client selection algorithms [15] address the straggler issue, and fairness between clients is discussed in [16]. Non-IID data biases are countered in [17], and training under resource constraints is accelerated via task offloading [18]. In [19], the authors studied the non-convex resource allocation problem of FL over wireless networks. The Federated Dropout (FedDrop) technique [20] reduces resource load, overfitting, and communication overhead by pruning models using heterogeneous dropout rates. Unlike FedDrop, the proposed SFL framework separates the model into client-side front-end layers and server-side back-end layers, reducing the computational burden on resource-limited devices.

Split Learning: To address the bottleneck caused by limited client resources, Split Learning (SL) [21] splits large AI models into parts, enabling separate training. A parallel SL method [22] prevents overfitting by managing the order and size of segmented models. SL has been applied in collaborative DNN computation between mobile devices and cloud servers [23], where optimal resource scheduling is formulated as a shortest path problem. The multi-split algorithm in [24] optimizes DNN submodel assignment across computational nodes. SL has also been used to reduce energy costs for edge devices under varying wireless channels [25]. In [26], SL tackles challenges in training GANs in non-IID scenarios. Hybrid Split Federated Learning (HSFL) combines FL and SL for wireless networks, analyzing convergence with non-IID data [27].

Collaborative Training of AI Models: In collaborative training frameworks like FL, differences in computing power and workload between clients and edge servers create bottlenecks. To mitigate the straggler issue, [28] sets a timeout threshold to discard delayed models, reducing overall latency. Dynamic allocation of computing and communication resources to clients [29] enables task offloading, but estimating client training time remains a challenge. The optimal regression model [9] predicts client training time, while [16] uses Lyapunov optimization to address online scheduling. A DRL-based offloading strategy [10] minimizes training latency for heterogeneous clients. DNN partitioning techniques [30] reduce FL latency by optimizing participation rates, energy consumption, and memory usage.

TABLE 1
Comparison of Related Works and this Paper

Literature	[5]	[20]	[21], [23], [24], [26]	[22], [25], [27]	This work
Model splitting			✓	✓	✓
Independent splitting					✓
Parallel training	✓	✓		✓	✓
Client workload	High	Middle	Low	Low	Low
Dynamic allocation		✓		✓	✓

Summary: This paper builds on existing works by combining edge computing and SL to enhance FL training efficiency. We jointly address the challenges of cut-layer

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

selection and resource allocation, which have not been explored together in prior studies. Table 1 compares this work with previous research.

3 System Model

We consider a FL system comprising a PS and K clients. The PS aims to train an AI model using client data without direct data sharing. We first review the basic FedAvg algorithm, followed by the SFL method, which integrates edge computing and split learning into FL.

3.1 Basics of FedAvg

In FedAvg [5], the PS broadcasts an initial model $\mathbf{w}$ to the K clients. Each client k trains a local model $\mathbf{w}_k$ independently using its dataset $\mathcal{D}_k = \{(x_{k,n}, y_{k,n}) \mid n = 1, \ldots, n_k\}$, where n_k is the dataset size, and $x_{k,n}$ and $y_{k,n}$ are the input and label, respectively.

Local Training at Client k: The client performs Stochastic Gradient Descent (SGD) over a mini-batch $\mathcal{B}_k \subseteq \mathcal{D}_k$ in each training epoch. For each sample $(x_{k,n}, y_{k,n}) \in \mathcal{B}_k$, client k computes the forward pass (FP) to get the prediction $\hat{y}_{k,n} = \mathtt{fp}(x_{k,n}; \mathbf{w}_k)$, followed by the backward pass (BP) to compute the gradient $\nabla \mathcal{L}(\hat{y}_{k,n}, y_{k,n}; \mathbf{w}_k)$, where $\mathcal{L}$ is the loss function. The local model is updated as

$$\mathbf{w}_k = \mathbf{w}_k - \eta \nabla \mathcal{L}(\hat{y}_{k,n}, y_{k,n}; \mathbf{w}_k), \tag{1}$$

where η is the learning rate. This process is repeated I_k times. After training, client k uploads $\mathbf{w}_k$ to the PS for aggregation.

Global Model Update at the PS: After collecting all local models, the PS aggregates them using the weighted average

$$\mathbf{w} = \sum_{k \in \mathcal{K}} \frac{n_k}{\sum_{k \in \mathcal{K}} n_k} \mathbf{w}_k, \tag{2}$$

where n_k is the size of client k's dataset. The global model is updated iteratively until the learning goal is met.

3.2 Framework of SFL

Following [9], an AI model is divided into L layers

$$\mathbf{w} = w^{(1)} \uplus \cdots \uplus w^{(l)} \uplus \cdots \uplus w^{(L)}, \tag{3}$$

where $w^{(l)}$ denotes the l^{th} layer. The model $\mathbf{w}_k$ for client k is split into two parts

$$\mathbf{w}_k = \mathbf{w}_k^{\mathrm{C}} \uplus \mathbf{w}_k^{\mathrm{S}}, \tag{4}$$

where $\mathbf{w}_k^{\mathrm{C}} = w^{(1)} \uplus \cdots \uplus w^{(l_k)}$ is the client-side model, and $\mathbf{w}_k^{\mathrm{S}} = w^{(l_k+1)} \uplus \cdots \uplus w^{(L)}$ is the server-side model. The cut-layer $w^{(l_k)}$ marks the split point for client k [31].

In Fig. 2, we outline the SFL workflow, which involves five phases: 1) Model splitting; 2) Model distribution; 3) Local training; 4) Model collection; and 5) Model aggregation. The key difference from FedAvg occurs in phases 3 and 5, as detailed below.

Phase 3 (Training Local Models): The training in SFL involves collaboration between the client and the PS, as the model is split. This process includes:

- *Forward Propagation (FP)*:

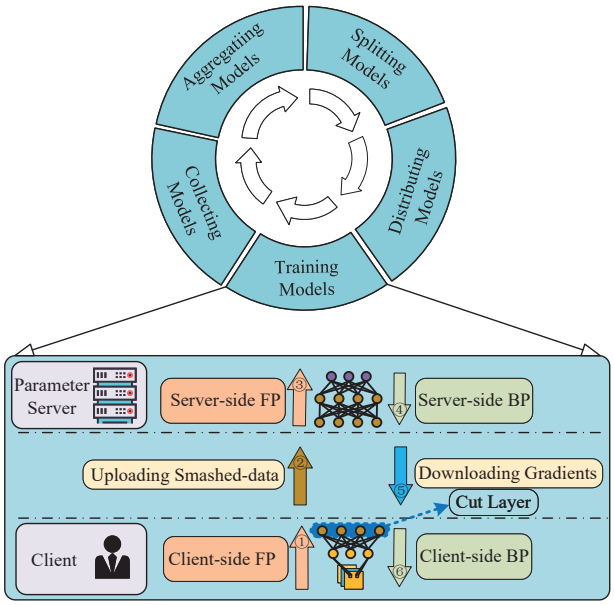


Fig. 2. The workflow of the SFL.

(1) *Client-side FP*: Client k performs the forward pass on $\mathbf{w}_k^{\mathrm{C}}$ using mini-batch $\mathcal{B}_k$, yielding the output at the cut-layer

$$\mathcal{S}_{k,n} = \mathtt{fp}(x_{k,n}; \mathbf{w}_k^{\mathrm{C}}). \tag{5}$$

(2) *Uploading Smashed Data*: Client k uploads the *smashed data* $\mathcal{S}_{k,n}$ and label $y_{k,n}$ to the PS for further processing. Uploading labels to PS undermines the data-privacy of clients. To address this issue, the authors in [32] proposed a three-stage SL method to avoid the leakage. For reconstruction attacks on training data [33], the leakage risk can also be reduced by using the differential privacy [34] [35].

(3) *Server-side FP*: The PS continues the FP on $\mathbf{w}_k^{\mathrm{S}}$ to compute the prediction $\hat{y}_{k,n}$ as

$$\hat{y}_{k,n} = \mathtt{fp}(\mathcal{S}_{k,n}; \mathbf{w}_k^{\mathrm{S}}). \tag{6}$$

- *Backward Propagation (BP)*:

(4) *Server-side BP*: The PS calculates the gradient and updates $\mathbf{w}_k^{\mathrm{S}}$ as

$$\mathbf{w}_k^{\mathrm{S}} = \mathbf{w}_k^{\mathrm{S}} - \eta \nabla \mathcal{L}(\hat{y}_{k,n}, y_{k,n}; \mathbf{w}_k^{\mathrm{S}}). \tag{7}$$

(5) *Downloading Gradients*: The PS transfers the gradient $\mathcal{G}_{k,n}$ to client k for further BP.

(6) *Client-side BP*: Client k completes the BP and updates $\mathbf{w}_k^{\mathrm{C}}$ as

$$\mathbf{w}_k^{\mathrm{C}} = \mathbf{w}_k^{\mathrm{C}} - \eta \nabla \mathcal{L}(\mathcal{G}_{k,n}; \mathbf{w}_k^{\mathrm{C}}). \tag{8}$$

Phase 5 (Aggregating Local Models): After local training, client k uploads the updated $\mathbf{w}_k^{\mathrm{C}}$ to the PS. The PS combines it with the server-side model $\mathbf{w}_k^{\mathrm{S}}$ to form the full local model $\mathbf{w}_k = \mathbf{w}_k^{\mathrm{C}} \uplus \mathbf{w}_k^{\mathrm{S}}$. The global model is updated by aggregating all local models as in eq. (2).

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

4 Problem Formulation

Although SFL reduces the computational burden on clients by training local models, it inevitably increases communication overhead. This is because intermediate results, $\mathcal{S}_{k,n}$ and $\mathcal{G}_{k,n}$, are exchanged multiple times between the PS and each client during the collaborative training process.

In this section, we first quantify the latency of each phase in SFL, followed by the proposal of a training latency minimization problem.

4.1 Latency of Each Phase

Referring to Sec. 3.2, we analyze the latency of each phase of SFL as follows.

Phase 1 (Splitting Local Models): The latency in this phase arises from executing **Algorithm 1**, the joint cut-layer selection and computing resource allocation algorithm from Sec. 5.5. Since this algorithm has polynomial time complexity and runs on the PS with sufficient computing power, its latency is negligible.

Phase 2 (Distributing Local Models): Let r_k denote the data rate between the PS and client k in the current round of global training. The latency for client k to download the *client-side* model $\mathbf{w}_k^{\mathrm{C}}$ from the PS is given by

$$\mathtt{DM}_k = \frac{|\mathbf{w}_k^{\mathrm{C}}|}{r_k},\ \forall k \in \mathcal{K}, \tag{9}$$

where $|\mathbf{w}_k^{\mathrm{C}}|$ is the size (in bits) of $\mathbf{w}_k^{\mathrm{C}}$.

Phase 3 (Training Local Models): As discussed in Sec. 3.2, client k trains its local model $\mathbf{w}_k = \mathbf{w}_k^{\mathrm{C}} \uplus \mathbf{w}_k^{\mathrm{S}}$ over six stages. The latency for each stage is as follows:

(1) *Client-side FP:* Let F_k^{C} be the number of floating-point operations (Flops) required for client k to perform sample-wise FP on $\mathbf{w}_k^{\mathrm{C}}$, and f_k^{C} (Flops/s) the client's computing power. The latency to process $|\mathcal{B}_k|$ samples is

$$\mathtt{FP}_k^{\mathrm{C}} = \frac{F_k^{\mathrm{C}}|\mathcal{B}_k|}{f_k^{\mathrm{C}}},\ \forall k \in \mathcal{K}. \tag{10}$$

(2) *Uploading Smashed Data:* Let $|\mathcal{S}_{k,n}|$ denote the size (in bits) of $\mathcal{S}_{k,n}$. The latency to transmit $|\mathcal{B}_k|$ pieces of smashed data is

$$\mathtt{TS}_k = \frac{|\mathcal{S}_{k,n}||\mathcal{B}_k|}{r_k},\ \forall k \in \mathcal{K}. \tag{11}$$

(3) *Server-side FP:* Let F_k^{S} represent the number of Flops required for the PS to perform sample-wise FP on $\mathbf{w}_k^{\mathrm{S}}$, and f_k^{S} (Flops/s) the computing power allocated by the PS to client k. The latency for processing $|\mathcal{B}_k|$ samples is

$$\mathtt{FP}_k^{\mathrm{S}} = \frac{F_k^{\mathrm{S}}|\mathcal{B}_k|}{f_k^{\mathrm{S}}},\ \forall k \in \mathcal{K}. \tag{12}$$

(4) *Server-side BP:* Let B_k^{S} denote the Flops required for the PS to perform sample-wise BP on $\mathbf{w}_k^{\mathrm{S}}$. The latency to process $|\mathcal{B}_k|$ samples is

$$\mathtt{BP}_k^{\mathrm{S}} = \frac{B_k^{\mathrm{S}}|\mathcal{B}_k|}{f_k^{\mathrm{S}}},\ \forall k \in \mathcal{K}. \tag{13}$$

(5) *Downloading Gradient:* Let $|\mathcal{G}_{k,n}|$ (in bits) denote the size of $\mathcal{G}_{k,n}$. The latency to download $|\mathcal{B}_k|$ gradients is

$$\mathtt{TG}_k = \frac{|\mathcal{G}_{k,n}||\mathcal{B}_k|}{r_k},\ \forall k \in \mathcal{K}. \tag{14}$$

(6) *Client-side BP:* Let B_k^{C} be the Flops required for client k to perform sample-wise BP on $\mathbf{w}_k^{\mathrm{C}}$. The time required to process $|\mathcal{B}_k|$ samples is

$$\mathtt{BP}_k^{\mathrm{C}} = \frac{B_k^{\mathrm{C}}|\mathcal{B}_k|}{f_k^{\mathrm{C}}},\ \forall k \in \mathcal{K}. \tag{15}$$

Phase 4 (Collecting Client Models): After I_k local training epochs, client k uploads the updated *client-side* model $\mathbf{w}_k^{\mathrm{C}}$ to the PS for aggregation. The latency is

$$\mathtt{UM}_k = \frac{|\mathbf{w}_k^{\mathrm{C}}|}{r_k},\ \forall k \in \mathcal{K}. \tag{16}$$

Phase 5 (Aggregating Local Models): Aggregating client models requires minimal computation. Since the PS has sufficient computing power, the latency is negligible.

4.2 Overall Time Consumption of SFL

In general, the *smashed-data* and *gradients* generated for one data sample have the same size: $|\mathcal{S}_{k,n}| = |\mathcal{G}_{k,n}| = \Lambda_k$. The latency for client k to complete a local training session (i.e., I_k epochs) is

$$\begin{aligned} T_k &= \mathtt{DM}_k + \mathtt{UM}_k + \\ &\quad I_k|\mathcal{B}_k|(\mathtt{FP}_k^{\mathrm{C}} + \mathtt{TS}_k + \mathtt{FP}_k^{\mathrm{S}} + \mathtt{BP}_k^{\mathrm{S}} + \mathtt{TG}_k + \mathtt{BP}_k^{\mathrm{C}}) \\ &= 2\frac{|\mathbf{w}_k^{\mathrm{C}}|}{r_k} + I_k|\mathcal{B}_k|\left(\frac{F_k^{\mathrm{C}} + B_k^{\mathrm{C}}}{f_k^{\mathrm{C}}} + \frac{F_k^{\mathrm{S}} + B_k^{\mathrm{S}}}{f_k^{\mathrm{S}}} + 2\frac{\Lambda_k}{r_k}\right), \\ &\quad \forall k \in \mathcal{K}. \end{aligned} \tag{17}$$

When the *cut-layer* is selected as $l_k = L$, we have $\mathbf{w}_k^{\mathrm{C}} = \mathbf{w}$ and $\mathbf{w}_k^{\mathrm{S}} = \emptyset$, meaning client k trains the entire model $\mathbf{w}_k$ locally, similar to FedAvg. In this case, the following holds

$$F_k^{\mathrm{S}} = 0,\ B_k^{\mathrm{S}} = 0,\ \Lambda_k = 0,\ \text{if } l_k = L,\ \forall k \in \mathcal{K}. \tag{18}$$

By substituting eq. (18) into eq. (17) , the latency for client k to complete a local training session in this case is given by

$$T_k = 2\frac{|\mathbf{w}|}{r_k} + I_k|\mathcal{B}_k|\frac{\Gamma}{f_k^{\mathrm{C}}},\ \text{if } l_k = L,\ \forall k \in \mathcal{K}, \tag{19}$$

where $|\mathbf{w}|$ and Γ are the size of model $\mathbf{w}$ (in bits) and total amount of computing load for one data-sample of model $\mathbf{w}$, respectively.

Combining eqs. (17) and (19), we can represent the training latency T_k of client k in the following form.

$$T_k = \begin{cases} 2\dfrac{|\mathbf{w}_k^{\mathrm{C}}|}{r_k} + I_k|\mathcal{B}_k|\left(\dfrac{F_k^{\mathrm{C}} + B_k^{\mathrm{C}}}{f_k^{\mathrm{C}}} + \dfrac{F_k^{\mathrm{S}} + B_k^{\mathrm{S}}}{f_k^{\mathrm{S}}} + 2\dfrac{\Lambda_k}{r_k}\right), \\ \qquad \forall l_k \in \{1, \cdots, L-1\},\ \forall k \in \mathcal{K}, \\ 2\dfrac{|\mathbf{w}|}{r_k} + I_k|\mathcal{B}_k|\dfrac{\Gamma}{f_k^{\mathrm{C}}},\ \text{if } l_k = L,\ \forall k \in \mathcal{K}. \end{cases} \tag{20}$$

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

4.3 Training Latency Minimization Problem

The latency for client k to train the model and communicate intermediate results depends on the selected cut-layer l_k, as shown in eq. (17). The choice of the split layer affects both computational and communication loads. *Computational Load*: A higher split layer increases local computation on the client, resulting in higher latency. Conversely, a lower split layer shifts the computation to the PS, reducing local processing time but increasing remote training time at the PS. *Communication Load*: The communication load is determined by the volume of data exchanged between the client and the PS. Different split layers affect the amount of intermediate data transmitted, influencing both time and bandwidth usage. While the PS is more powerful than individual clients, it serves multiple clients simultaneously, which limits its resources. Therefore, the communication and computation resources of both the clients and the PS must be jointly scheduled to optimize the training process. The method aims to select the optimal split layer for each client, balancing computational and communication delays to minimize the overall training latency of SFL. The optimal split layer is the result of a tradeoff between local computation and communication time, tailored to each client's hardware and network conditions.

As we can only predict the available computing resources and data rate for the clients and PS for a short time into the future [10], our objective is to minimize $\mathcal{T}$, the latency to complete one round of global training. With SGMU adopted, the total latency $\mathcal{T}$ depends on $\mathcal{T} = \max_{k\in\mathcal{K}} T_k$, which is the maximum latency for all K clients to complete a local training session. Let $\mathbf{L} = (l_1, \ldots, l_K)$ and $\mathbf{F} = (f_1^{\mathrm{S}}, \ldots, f_K^{\mathrm{S}})$ denote the cut-layer and computing resources for each client. The training latency minimization problem for SFL is formulated as

$$\min_{\mathbf{L},\mathbf{F}} \mathcal{T}, \quad \mathcal{T} = \max_{k\in\mathcal{K}} T_k, \tag{21}$$

$$\text{s.t.} \quad l_k \in \{1, \cdots, L\}, \ \forall k \in \mathcal{K}, \tag{21.1}$$

$$\sum_{k=1}^{K} f_k^{\mathrm{S}} \leqslant F^{\max}, \tag{21.2}$$

$$|\mathbf{w}_k^{\mathrm{C}}| + |\mathbf{w}_k^{\mathrm{S}}| = |\mathbf{w}|, \ \forall k \in \mathcal{K}, \tag{21.3}$$

$$F_k^{\mathrm{C}} + F_k^{\mathrm{S}} + B_k^{\mathrm{S}} + B_k^{\mathrm{C}} = \Gamma, \ \forall k \in \mathcal{K}, \tag{21.4}$$

where constraint (21.1) limits the cut-layer for client k, constraint (21.2) ensures the maximum computing resources for the PS are $F^{\max}$, constraint (21.3) guarantees model integrity after being split, as in eq. (4), and constraint (21.4) enforces the balance of computing resources across client-side and server-side operations.

Given the AI model, the total computational resources required for training with one data sample is Γ, which is the sum of resources required for FP and BP. The resources for FP and BP at client k are $F^{\mathrm{tot}} = F_k^{\mathrm{C}} + F_k^{\mathrm{S}}$ and $B^{\mathrm{tot}} = B_k^{\mathrm{C}} + B_k^{\mathrm{S}}$, respectively, leading to the equation

$$\Gamma = F^{\mathrm{tot}} + B^{\mathrm{tot}} = F_k^{\mathrm{C}} + F_k^{\mathrm{S}} + B_k^{\mathrm{S}} + B_k^{\mathrm{C}}, \ \forall k \in \mathcal{K}. \tag{22}$$

Since both continuous variables $\mathbf{F}$ and discrete variables $\mathbf{L}$ are involved, this problem is a mixed integer nonlinear programming (MINLP) problem, which cannot be solved directly by conventional methods. In the next section, we propose effective methods to address this challenge.

5 Solving the Problem

In problem (21), the model size $|\mathbf{w}_k^{\mathrm{C}}|$, the resources needed for client-side computation $F_k^{\mathrm{tot}} = F_k^{\mathrm{C}} + B_k^{\mathrm{C}}$, and the amount of generated intermediate results Λ_k all depend on the selected cut-layer l_k for client k. However, as shown in Figs. 3, 5, and 6, there is no explicit relationship between the cut-layer and these parameters. Since DNNs often have hundreds of layers, searching for the optimal cut-layers could lead to high time and space complexity. Currently, there is no direct solution to this issue. In this paper, we propose a regression method to quantify the relationship between the cut-layer and the model parameters. Based on this, we develop a fast and effective approach to find the optimal solution to problem (21).

Logistic regression and curve fitting methods have been widely used to model such relationships. For instance, in [36] and [37], these methods were employed to build nonlinear energy harvesting models for wireless information and power transfer (SWIPT) systems.

5.1 Fit The Relationship

Different AI models have distinct neural network architectures. To the best of our knowledge, no existing method provides an explicit relationship expression for these models. In this paper, we use the widely adopted AI model *EfficientNetV2* [11] as a case study and introduce the *regression method* to fit the relationship between the *cut-layer* and other model parameters.

EfficientNetV2 offers high configurability in terms of depth, width, and resolution, making it suitable for various tasks and hardware setups. However, it also introduces complex dependencies between the cut-layer and the associated computational and communication overheads. The proposed regression-based approach for selecting the optimal split layer can be extended to other neural network architectures. By applying this method, we can determine the optimal cut layers for different models, ensuring a balanced distribution of local and remote computations across diverse AI architectures.

5.1.1 Client-side model size against different cut-layers

As shown in Fig. 3, with increasing l_k, the increase of $|\mathbf{w}_k^{\mathrm{C}}|$ is not obvious at the beginning, but the growth rate increases sharply when $l_k > 36$. Therefore, we set the relationship between l_k and $|\mathbf{w}_k^{\mathrm{C}}|$ as

$$|\mathbf{w}_k^{\mathrm{C}}| = \alpha (l_k)^2, \ \forall k \in \mathcal{K}, \tag{23}$$

where $\alpha \geq 0$ is the parameter to be fitted.

5.1.2 Training load of client against different cut-layers

The training of the *client-side* model includes the FP and BP. The computational load of performing the FP can be obtained by using the python package *torchinfo* [38], however, there is no direct way to know the computational load of performing the BP. To address this issue, we trained *EfficientNetV2* 1500 times using a dataset of size 32. The time consumed by FP and BP in each training session is shown in Fig. 4.

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

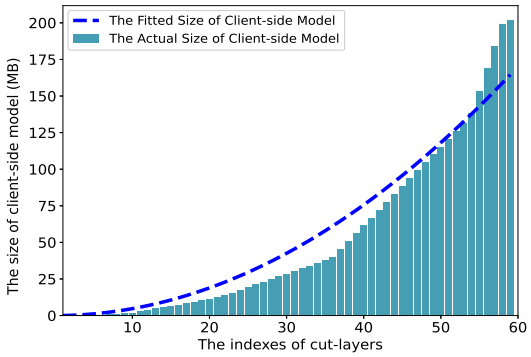


Fig. 3. The quantitative-relationship between l_k and $|\mathbf{w}_k^{\mathrm{C}}|$.

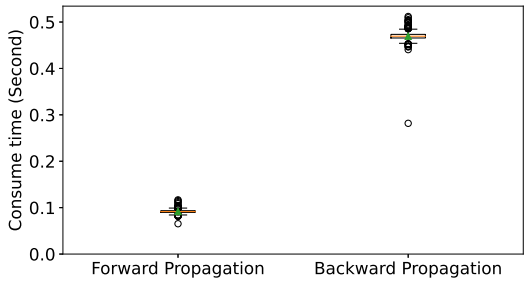


Fig. 4. The time consumption of FP and BP

From Fig. 4, we see that with the same computing power the time consumed by BP is roughly several times the time consumed by FP. So we have

$$B_k^{\mathrm{tot}} \approx \kappa F_k^{\mathrm{tot}},\ \forall k \in \mathcal{K}, \tag{24}$$

where $\kappa \geqslant 1$. Fig. 5 shows the computational load of client k for training the *client-side* model (that is, $F_k^{\mathrm{tot}} = F_k^{\mathrm{C}} + B_k^{\mathrm{C}}$) with different *cut-layers*. There is an approximate linear relationship between them, which is given by

$$F_k^{\mathrm{tot}} = F_k^{\mathrm{C}} + B_k^{\mathrm{C}} = \beta l_k(1+\kappa),\ \forall k \in \mathcal{K}, \tag{25}$$

where $\beta > 0$ is the parameter to be fitted.

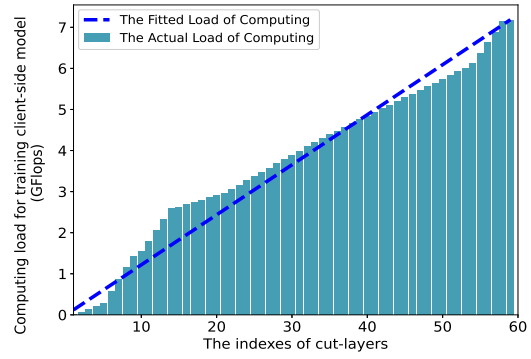


Fig. 5. The quantitative-relationship between l_k and F_k^{tot}.

5.1.3 Size of intermediate results against different cut-layers

The head of a DNN is always a convolutional neural network (CNN) to extract features, thus reducing the computational burden of the subsequent feedforward neural network (FNN). Hence, the size of the *intermediate results* decreases rapidly in the CNN layers but changes slowly in the FNN layers, as shown in Fig. 6. Accordingly, we set the relationship between l_k and Λ_k as

$$\Lambda_k = |\mathcal{S}_{k,n}| = |\mathcal{G}_{k,n}| = \frac{\gamma_1}{l_k + \gamma_2},\ \forall k \in \mathcal{K}, \tag{26}$$

where $\gamma_1 > 0$ and $\gamma_2 \geqslant 0$ are the parameters to be fitted.

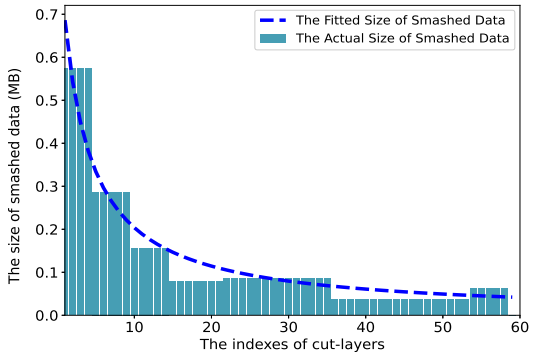


Fig. 6. The quantitative-relationship between l_k and Λ_k.

5.2 Problem Reformulation

By substituting eqs. (23), (25), and (26) into eq. (20), the expression for T_k can be written as

$$T_k = \begin{cases} 2\frac{\alpha l_k^2}{r_k} + I_k\left|\mathcal{B}_k\right|\left(\beta(1+\kappa)\left(\frac{l_k}{F_k^{\mathrm{C}}} + \frac{L-l_k}{F_k^{\mathrm{S}}}\right) + 2\frac{\frac{\gamma_1}{l_k+\gamma_2}}{r_k}\right), \\ \qquad\qquad \text{if } 1 \leqslant l_k < L,\ \forall k \in \mathcal{K}, \\ \frac{2|\mathbf{w}|}{r_k} + I_k|\mathcal{B}_k|\frac{\Gamma}{f_k^{\mathrm{C}}},\ \text{if } l_k = L,\ \forall k \in \mathcal{K}. \end{cases} \tag{27}$$

Then, the MINLP problem (21) is transformed into the following continuous problem.

$$\min_{\mathbf{L},\mathbf{F}} \mathcal{T},\ \ \mathcal{T} = \max_{k\in\mathcal{K}}\ T_k, \tag{28}$$

$$\text{s.t.}\ \ 1 \leqslant l_k \leqslant L,\ \forall k \in \mathcal{K}, \tag{28.1}$$

$$\sum_{k=1}^{K} f_k^{\mathrm{S}} \leqslant F^{\max}. \tag{28.2}$$

One can first find the solution of problem (28), and then, rounds it to an integer. We note that if the computing resource allocation $\mathbf{F}$ for the K clients were known, problem (28) can be decomposed into K independent subproblems, and the goal of the k^{th} subproblem is to minimize the latency for client k to complete a local training session. These subproblems can be solved in parallel without loss of optimality.

Based on the above findings, we can solve the *cut-layer selection* problem (to find the optimal $\mathbf{L}$) and the *computing resource allocation* problem (to find the optimal $\mathbf{F}$) alternately and iteratively. Finally, the solution to problem (28) can be obtained.

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

5.3 Solving The Cut-Layer Selection Problem

For any given $\mathbf{F}$, problem (28) can be decomped into K independent subproblems as given below.

$$\min_{l_k} T_k,\ \forall k \in \mathcal{K}, \tag{29}$$
$$\text{s.t.}\quad 1 \leqslant l_k \leqslant L,\ \forall k \in \mathcal{K}. \tag{29.1}$$

The solution of problem (29), i.e., the optimal *cut-layer* of any client k is given in the following lemma.

Lemma 1. *With $\alpha > 0,\ \beta > 0,\ \kappa > 0,\ \gamma_1 > 0,\ and\ \gamma_2 \geqslant 0$, the optimal l_k^* for any client k is given by the following rules.*

$$l_k^* = \begin{cases} 1, & \frac{\partial T_k}{\partial l_k}\big|_{l_k=1} > 0, \\ L, & \frac{\partial T_k}{\partial l_k}\big|_{l_k=L} < 0, \\ \left\lfloor \arg\limits_{l_k} \left(\frac{\partial T_k}{\partial l_k} = 0\right) \right\rfloor, & \frac{\partial T_k}{\partial l_k}\big|_{l_k=1} \leqslant 0\ and\ \frac{\partial T_k}{\partial l_k}\big|_{l_k=L} \geqslant 0. \end{cases} \tag{30}$$

Proof. Please refer to Appendix A. □

In the 3rd case of eq. (30), $\frac{\partial T_k}{\partial l_k} = 0$ contains a cubic terms of l_k and can be solved by using the *Cardano's formula* [39].

Since the obtained l_k is a continuous value, it must be rounded to obtain the specific *cut-layer*. Here, we did not use more complex rounding algorithms, such as progressive rounding [40] and pair-wise rounding [41] to obtain the optimal solution for l_k. We use a simpler rounding method, namely the threshold method, which rounds the value of l_k in continuous space to the nearest lower integer $\lfloor l_k \rfloor$. Our focus is to minimize the completion time of SFL. In addition, it is worth noting that the complexity of finding the suboptimal *cut-layers* for the K clients is only $\mathcal{O}(K)$, thus avoiding an exhaustive search on the actual complex relationships between the parameters of an AI-model with extremely high time and space complexity.

5.4 Computing Resource Allocation of the PS

Substituting the obtained *cut-layers* of the K clients, namely, $\mathbf{L}^* = (l_1^*, \cdots, l_K^*)$, into eq. (27), one can get

$$T_k = \begin{cases} 2\frac{\alpha (l_k^*)^2}{r_k} + I_k \left|\mathcal{B}_k\right| \left(\beta(1+\kappa)\left(\frac{l_k^*}{F_k^{\mathrm{C}}} + \frac{L-l_k^*}{F_k^{\mathrm{S}}}\right) + 2\frac{\frac{\gamma_1}{l_k^*+\gamma_2}}{r_k}\right), \\ \qquad\qquad \text{if } 1 \leqslant l_k^* < L,\ \forall k \in \mathcal{K}, \\ \frac{2|\mathbf{w}|}{r_k} + I_k |\mathcal{B}_k| \frac{\Gamma}{f_k^{\mathrm{C}}},\ \text{if } l_k^* = L,\ \forall k \in \mathcal{K}. \end{cases} \tag{31}$$

Problem (28) can be simplified to

$$\min_{\mathbf{F}}\ \mathcal{T} = \max_{k \in \mathcal{K}} T_k, \tag{32}$$
$$\text{s.t.}\quad \sum_{k=1}^{K} f_k^{\mathrm{S}} \leqslant F^{\max}. \tag{32.1}$$

The difficulty in solving problem (32) is that the objective, $\mathcal{T} = \max_{k \in \mathcal{K}} T_k$, to be optimized is a nonlinear function w.r.t T_k, and T_k is a piecewise function w.r.t l_k^*. To solve this difficulty, we define $\tilde{T}_k$ as the latency for client k to complete a local training session using only the local computing power f_k^{C}. We can sort the K clients in ascending order according to $\{\tilde{T}_k | \forall k \in \mathcal{K}\}$ as shown in eq. (33), where the reordered index of client k is represented as $\tilde{k}$, $\forall \tilde{k} \in \mathcal{K}$.

$$\overbrace{\tilde{T}_1 \leqslant \cdots \leqslant \tilde{T}_{\tilde{k}-1=\theta-1}}^{\texttt{FedAvg}} \leqslant \overbrace{\tilde{T}_{\tilde{k}=\theta} \leqslant \cdots \leqslant \tilde{T}_K}^{\texttt{SFL}} \tag{33}$$

Next, we derive the following two lemmas.

Lemma 2. *The K clients participating in an FL task can be divided into two groups: the clients in group $\mathcal{K}_{\texttt{FedAvg}} = \{\tilde{1}, \cdots, \theta-1\}$ adopt FedAvg and train $\boldsymbol{w}_{\tilde{k}}$ independently, while the clients in group $\mathcal{K}_{\texttt{SFL}} = \{\theta, \cdots, \tilde{K}\}$ adopt SFL and train $\boldsymbol{w}_{\tilde{k}}$ in collaboration with the PS, where θ is the index of the first client in $\mathcal{K}_{\texttt{SFL}}$.*

Proof. Please refer to Appendix B. □

Lemma 3. *The optimum of the objective of problem* (32) *is $\mathcal{T} = T_\theta$, $\theta \in \tilde{\mathcal{K}}$.*

Proof. Please refer to Appendix C. □

By using **Lemmas 2** and **3**, problem (32) can be converted into the following form.

$$\min_{\theta,\ \mathbf{F}}\ T_\theta, \tag{34}$$
$$\text{s.t.}\quad \mathcal{K}_{\texttt{FedAvg}} \cup \mathcal{K}_{\texttt{SFL}} = \mathcal{K}, \tag{34.1}$$
$$\mathcal{K}_{\texttt{FedAvg}} \cap \mathcal{K}_{\texttt{SFL}} = \emptyset, \tag{34.2}$$
$$T_{\tilde{k}} \leqslant T_\theta,\ \forall \tilde{k} \in \mathcal{K}_{\texttt{FedAvg}}, \tag{34.3}$$
$$T_{\tilde{k}} = T_\theta,\ \forall \tilde{k} \in \mathcal{K}_{\texttt{SFL}}. \tag{34.4}$$

By solving problem (34), the optimal amount of computing resources that the PS allocates to any client k is obtained in closed-form, as shown in the following lemma.

Lemma 4. *The amount of computing resources allocated by the PS to each client $\tilde{k} \in \mathcal{K}$ is given by the following formula.*

$$f_{\tilde{k}}^{S} = \begin{cases} 0, & \forall \tilde{k} \in \mathcal{K}_{FedAvg}, \\ \frac{I_{\tilde{k}} \left|\mathcal{B}_{\tilde{k}}\right| (F_{\tilde{k}}^{\mathrm{S}} + B_{\tilde{k}}^{\mathrm{S}})}{T_\theta - \frac{I_{\tilde{k}} \left|\mathcal{B}_{\tilde{k}}\right| (F_{\tilde{k}}^{\mathrm{C}} + B_{\tilde{k}}^{\mathrm{C}})}{f_{\tilde{k}}^{\mathrm{C}}} - 2\frac{I_{\tilde{k}} \left|\mathcal{B}_{\tilde{k}}\right| \Lambda_{\tilde{k}} + |\boldsymbol{w}_{\tilde{k}}^{\mathrm{C}}|}{r_{\tilde{k}}}}, & \forall \tilde{k} \in \mathcal{K}_{SFL}. \end{cases} \tag{35}$$

Proof. Please refer to Appendix D. □

According to **Lemma 4**, once $\mathcal{T} = T_\theta$ is known, the optimal resource allocation of the PS, $\mathbf{F}$, can be obtained. In order to solve T_θ, we substitute formula (35) into constraint (32.1) and convert this inequality constraint into an equality constraint as

$$\sum_{\tilde{k} \in \mathcal{K}_{\texttt{SFL}}} \frac{I_{\tilde{k}} \left|\mathcal{B}_{\tilde{k}}\right| (F_{\tilde{k}}^{\mathrm{S}} + B_{\tilde{k}}^{\mathrm{S}})}{T_\theta - \frac{I_{\tilde{k}} \left|\mathcal{B}_{\tilde{k}}\right| (F_{\tilde{k}}^{\mathrm{C}} + B_{\tilde{k}}^{\mathrm{C}})}{f_{\tilde{k}}^{\mathrm{C}}} - 2\frac{I_{\tilde{k}} \left|\mathcal{B}_{\tilde{k}}\right| \Lambda_{\tilde{k}} + |\mathbf{w}_{\tilde{k}}^{\mathrm{C}}|}{r_{\tilde{k}}}} = F^{\max} \tag{36}$$

Lemma 5. *T_θ is strictly convex w.r.t F^{max}.*

Proof. Please refer to appendix E. □

Lemma 5 ensures that for a given $F^{\max}$ an unique solution T_θ exists for eq. (36), which can be solved by using the CVX tool [42].

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

Algorithm 1 The overall algorithm to solve problem (21).

1: Use eq. (19) to obtain $\tilde{T}_k, \forall k \in \mathcal{K}$;
2: Sort the K clients in ascending order according to $\{\tilde{T}_k | k \in \mathcal{K}\}$, and update the index of client k to its serial number $\tilde{k}$;
3: Set the iteration number $i = 0$, $\mathbf{F}^i = \{\frac{F^{\max}}{K}, \cdots, \frac{F^{\max}}{K}\}$, $\theta^i = 1$, and $\mathcal{T} = T_{\theta^i}$;
4: **repeat**
5: With the given $\mathbf{F}^i$, select *cut-layers* $\mathbf{L}^i$ for the clients using eq. (30);
6: Substitute $\mathbf{L}^i$ and θ^i into eq. (34) and obtain $T_{\theta^{i+1}}$ using the CVX tool [42];
7: With the obtained $T_{\theta^{i+1}}$, update $\mathbf{F}^{i+1}$ using eq. (35);
8: $\theta^{i+1} = \Phi\left(f^{\mathrm{S}}_{\tilde{1}}, \cdots, f^{\mathrm{S}}_{\tilde{K}}\right)$;
9: $i = i + 1$;
10: **until** $\mathcal{T} = T_{\theta^i}$ converges, or the iteration number reaches the maximum $i = i^{\max}$;
11: **return** The computing allocation strategy $\mathbf{F}$ of the PS and the cut-layers selected for the K clients $\mathbf{L}$.

5.5 Analysis of Algorithm Complexity

The overall algorithm for solving problem (21) is summarized in Algorithm 1. In *line 1*, each client k uploads its computing power f^{C}_k to the PS, which then calculates the initial time consumption $\tilde{T}_k$ for each client using eq. (19), providing an estimate of the local computation time. In *line 2*, the PS sorts the clients in ascending order based on $\tilde{T}_k$, updating their indices to reflect the varying computational demands. In *lines 3-10*, the PS iteratively selects the cut-layers $\mathbf{L}^i$ for each client using eq. (30) and solves the optimization problem (42) by substituting the selected layers and parameter θ^i into eq. (34). This step calculates the new total time $T_{\theta^{i+1}}$. Specifically, in *line 7*, the PS updates the computation allocation strategy $\mathbf{F}^{i+1}$ based on the newly calculated $T_{\theta^{i+1}}$. In *line 8*, the function $\Phi(\cdot)$ counts the number of non-zero elements in a vector. The process continues in *line 10*, iterating until the total training time T_{θ^i} converges or the maximum number of iterations $i^{\max}$ is reached, indicating the optimal split layer and a balanced distribution between local computations and offloaded tasks. Finally, in *line 11*, once convergence is achieved, the PS returns the final computation allocation strategy $\mathbf{F}$ and the optimal cut-layers $\mathbf{L}$ to each client. Since the algorithm is iterative, the convergence is proven below.

Lemma 6. *Algorithm 1 converges to a stable point.*

Proof. Please refer to appendix F. □

Finally, the computational complexity of **Algorithm 1** is analyzed as follows. The complexity of ranking K clients in *Step 2* is $\mathcal{O}(K^2)$. The complexity of selecting the *cut-layer* for the K clients in *Step 5* is $\mathcal{O}(K)$. In *Step 6*, the optimal T_θ can be obtained via linear search, which requires $\mathcal{O}(K^3)$ to converge [43]. The resource allocation at the PS in *Step 7* requires $\mathcal{O}(K)$. Thus, the overall computational complexity of **Algorithm 1** is $\mathcal{O}\left(K^2 + i^{\max}(K^3 + 2K)\right)$.

We now discuss two main challenges in implementing this algorithm: 1) *Synchronization Overhead*: Synchronizing model updates between clients and the PS can cause delays, especially when client capabilities vary or network conditions are unstable. Our approach mitigates this by optimizing the split layer, balancing the computational load, and reducing synchronization time. 2) *Data Privacy Concerns*: While federated learning keeps raw data on client devices, exchanging intermediate model data may still pose privacy risks. Techniques such as secure aggregation and differential privacy can help protect sensitive information, though their implementation is beyond the scope of this paper and remains a subject for future work.

6 Experiment Results

To evaluate the performance of the proposed `SFL` method, the *EfficientNetV2* model is trained for image classification tasks on the MNIST [12] and CIFAR-10 [44] datasets. MNIST consists of grayscale images with dimensions 28×28 pixels, while CIFAR-10 includes colored images of $32 \times 32 \times 3$ pixels, resulting in a higher-dimensional input space and more complex feature representations. In the independent and identically distributed (IID) setting, the samples of each class are evenly distributed across the clients involved in the federated learning (FL) task. In the non-IID setting, the dataset is partitioned among 30 clients following the Dirichlet distribution [45]. Using the data generation method in [19], the number of samples per client ranges from 361 to 3,578. The local dataset is split randomly, with 75% allocated for training and 25% for testing. The SGD algorithm, with a fixed batch size of $|\mathcal{B}_k| = 32$, is used to train the local model $\mathbf{w}_k$ for each client k. The available computing resources of the PS are set to $F^{\max} = 3,000$ G*Flops*, and a total of 30 clients are available as candidates for the PS-initiated FL tasks. The clients are heterogeneous, with varying local computing power f^{C}_k and transmission rates r_k. The PS randomly selects a fraction of the candidate clients to participate in each round of global training.

6.1 Effectiveness of The Regression Method in Finding The Optimal Cut-Layer

The *regression method*, proposed in Sec. 5.1, is used to quickly find the approximately optimal *cut-layer*. To verify its validity, we set the local computing power of client k to $f^{\mathrm{C}}_k = 100$ G*Flops*, the edge computing power of the PS allocated to client k to $f^{\mathrm{S}}_k = 1,484$ G*Flops*, the transmission rate to $r_k = 4$ Mb/s, and the local dataset size of client k to $|\mathcal{D}_k| = 3,396$. The latency for client k to complete a local training session with different *cut-layers* is shown in Fig. 7. For each *cut-layer*, the latency of training the *client-side* model $\mathbf{w}^{\mathrm{C}}_k$ and the *server-side* model $\mathbf{w}^{\mathrm{S}}_k$ and transmitting the *intermediate results* ($\mathcal{S}_{k,n}$ and $\mathcal{G}_{k,n}$) are separately annotated in Fig. 7.

As shown in Fig. 7, the `SFL` can considerably reduce the training latency of client k, compared to the `FedAvg`. Specially, when the optimal *cut-layer* $l_k = 11$ is selected, up to 300 seconds can be saved. When $l_k < 11$, the training latency mainly results from transferring the *intermediate results* between the client and PS. When $l_k > 11$, the training latency mainly comes from the training of the *client-side* model $\mathbf{w}^{\mathrm{C}}_k$. In all cases, the latency for the PS to train the *server-side* model $\mathbf{w}^{\mathrm{S}}_k$ and the latency for the client to download/upload the *client-side* model $\mathbf{w}^{\mathrm{C}}_k$ are negligible for the overall training latency.

From Fig. 7, we also note that the optimal *cut-layer* for the client found by using the *regression method* is consistent with

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

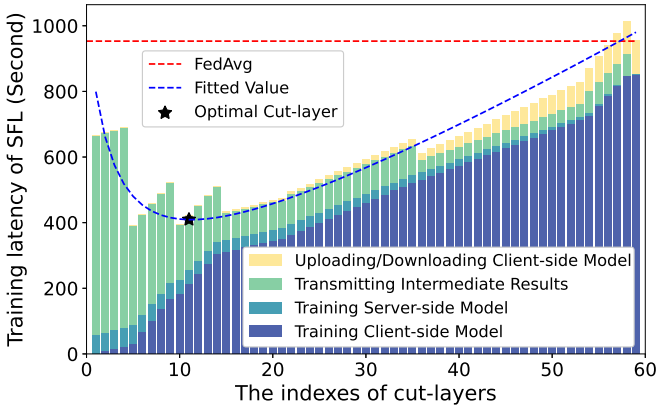


Fig. 7. Training latency of SFL with different *cut-layers*.

Fig. 8. Training latency of different clients in the SFL.

that obtained by an exhaustive search on the space created by the actual quantitative relationship between the parameters of *EfficientNetV2*. In this way, the high computational complexity caused by searching for the optimal *cut-layer* of an AI-model is avoided, and the bias between the regression curve and the real-world relationship is also minimized.

To quantify the prediction accuracy of the proposed *regression method*, we adopt the determination coefficient $\mathcal{R}$ [9] as the evaluation metric. $\mathcal{R} \triangleq 1 - \frac{\sum_{l=1}^{L}(x-\hat{x}_l)^2}{\sum_{l=1}^{L}(x_l-\tilde{x}_l)^2}$, where x_l, $\hat{x}_l$, and $\tilde{x}_l$ represent the true value, the predicted value, and the historical mean value, respectively. The closer the value of $\mathcal{R}$ is to 1, the better the regression performance is. In the following Tab. 2, we show the determination coefficient $\mathcal{R}$ of the fitted values of $|\mathbf{w}_k^{\mathrm{C}}|$ (see eq. (23)), F_k^{tot} (see eq. (25)) and Λ_k (see eq. (26)).

As shown in Tab. 2, $0.9 < \mathcal{R} < 1$ for all the predicted items. This indicates that the proposed *regression method* can accurately predict the relationship between the selected *cut-layer* and other parameters of an AI-model, which ensures that the proposed alternate-optimization-based **Algorithm 1** can obtain high-quality solutions as analyzed in Sec. 5.5.

TABLE 2
The Determination Coefficient of The Proposed Regression Method

Fitted Item	$\mathcal{R}$
$\lvert\mathbf{w}_k^{\mathrm{C}}\rvert$	0.9482
F_k^{tot}	0.9659
Λ_k	0.9065

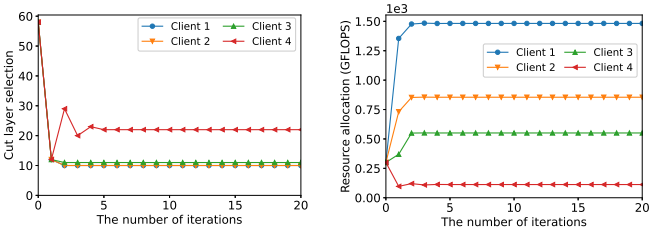
6.2 Results for joint cut-layer selection and computing resource allocation.

To validate the performance of the proposed joint *cut-layer* selection and computing resource allocation algorithm, namely **Algorithm 1**, we select 10 clients from the 30 candidates, each with the same number of local training epochs $I_k = 20$. Fig. 8 compares the training latency for each client using the proposed SFL to the vanilla FedAvg. For the SFL, the latency for each client to train the *client-side* and *server-side* models and communicate the *intermediate results* are also separately annotated in Fig. 8.

As shown in Fig. 8, the PS does not allocate any resources to clients 1 to 6 in SFL, so these clients use FedAvg for local model training. In contrast, clients 7 to 10 share the PS's computing resources to reduce training latency. Consequently, clients 7 to 10 have nearly identical training latencies (410 s), while the training latencies of clients 1 to 6 vary, each being lower than those of clients 7 to 10. This demonstrates that SFL behaves similarly to the *water pouring algorithm* [46], adaptively allocating PS resources to resource-constrained clients, thus reducing latency differences. In comparison, when all clients use FedAvg, each trains its local model $\mathbf{w}_k$ independently. Since the overall training latency is determined by the client with the longest training time, FedAvg results in a significantly higher latency (980 s) than SFL (410 s).

Next, we present the convergence of **Algorithm 1**. As shown in Fig. 9a and Fig. 9b, the convergence of cut-layer selection and resource allocation occurs after approximately 5 iterations, with the resulting training latency shown in Fig. 10. This confirms the low computational complexity of the proposed algorithm.

Finally, we examine how the amount of computing resources available to the PS affects the overall training latency of SFL. By increasing the PS's computing resources from $F^{\max} = 100$ G*Flops* to $F^{\max} = 9,000$ G*Flops*, we observe the variation in training latency for one local training session, as shown in Fig. 11.



(a) Convergence of *cut-layer* selection. (b) Convergence of resource allocation.

Fig. 9. Convergence of Algorithm 1 for the two subproblems.

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

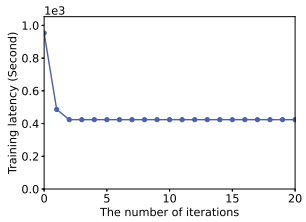


Fig. 10. Convergence of the training latency of SFL.

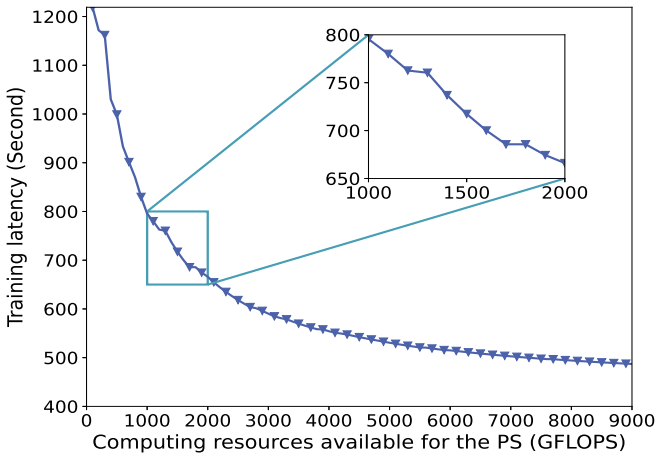


Fig. 11. Training latency of SFL with different $F^{\max}$.

As shown in Fig. 11, increasing the computing resources available to the PS reduces the training latency of SFL. This is because more resource-constrained clients can leverage the PS's resources, alleviating the system bottleneck (i.e., the longest training latency for clients to complete local training) and thereby reducing overall training latency. It can also be observed from Fig. 11 that the training latency of SFL decreases rapidly as server resources increase from $F^{\max} = 100$ G*Flops* to $F^{\max} = 2,000$ G*Flops*, but the rate of decrease slows as resources continue to increase. This suggests that the PS can optimize resource allocation to improve efficiency. Maximizing the marginal benefits of server resources is particularly crucial when the PS simultaneously trains multiple AI models or performs multi-task learning. This topic is outside the scope of this paper and will be explored in future work.

6.3 Test Accuracy of the Trained Model

This section presents the test accuracy of the proposed SFL method. We incorporate the FedDrop technique, introduced in [20], into the fully connected layers of *EfficientNetV2* on the PS. Specifically, a 20% dropout rate is applied to the parameters to mitigate overfitting and enhance model performance. Experiments were conducted on the MNIST and CIFAR-10 datasets under both IID and Non-IID scenarios. In each round of training, 10 clients were randomly selected from a pool of 30, with each client performing 20 local epochs ($I_k = 20$). The test accuracies of the models trained with different methods are shown in Fig. 12 (MNIST) and Fig. 13 (CIFAR-10).

From Figs. 12 and 13, we observe that as the number of global training rounds increases, the test accuracy of

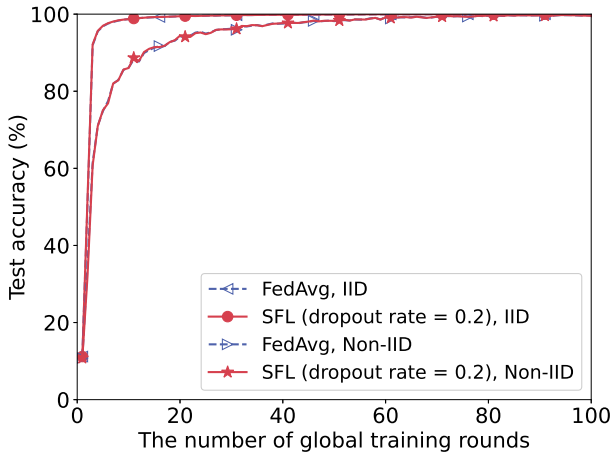


Fig. 12. Test accuracy on MNIST.

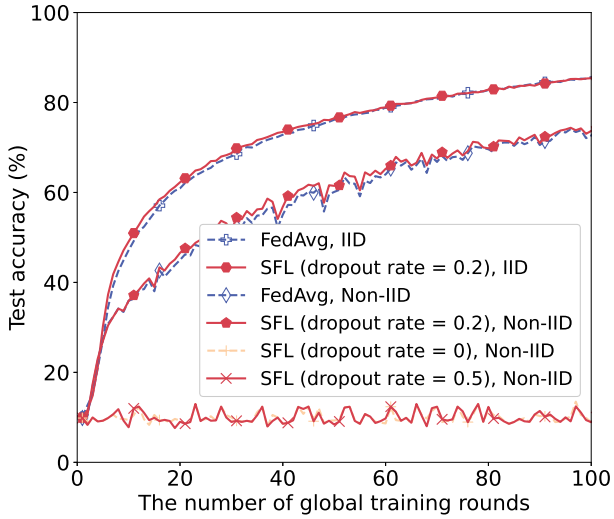


Fig. 13. Test accuracy on CIFAR-10.

models trained with SFL (20% dropout) and FedAvg improves. From Fig. 13, it is noted that the convergence rate of SFL with 20% dropout is similar to those of FedAvg and FedDrop, while using an inappropriate dropout rate (e.g. 0% or 50%) results in failure to converge, highlighting the importance of FedDrop.

For the CIFAR-10 experiment, time consumption for each method is illustrated in Fig. 14. It shows that SFL with 20% dropout consumes significantly less time to achieve similar test accuracy compared to FedAvg and FedDrop.

7 Conclusions

This paper leverages *edge computing* and *split learning* techniques to enhance the training efficiency of SFL, minimizing latency without sacrificing accuracy. We formulate the problem as an MINLP and propose a regression method to convert it into a continuous problem, followed by an alternate-optimization algorithm with polynomial time complexity. The convergence and solution quality are proven and experimentally validated. Results show that SFL matches FedAvg's accuracy with significantly reduced training time.

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

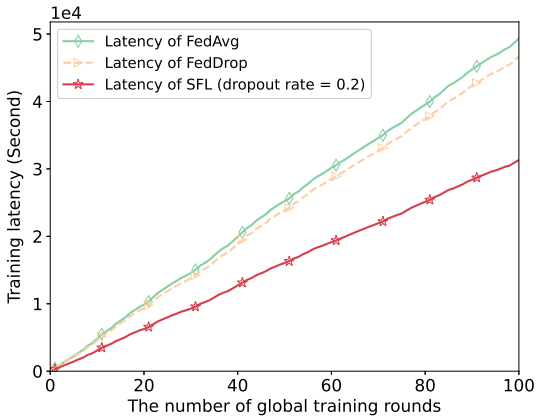


Fig. 14. Training latency of different methods.

Future work will focus on: 1) Enhancing PS efficiency for parallel training and multi-task learning; 2) Investigating the impact of communication challenges, such as Doppler shifts and spectrum scarcity, in large-scale FL; and 3) Integrating differential privacy (DP) mechanisms into SFL to mitigate server-side reconstruction attacks and guide cut-layer selection based on privacy budgets.

REFERENCES

[1] C. Yan, Y. Zhang, Q. Zhang, Y. Yang, X. Jiang, Y. Yang, and B. Wang, "Privacy-preserving online automl for domain-specific face detection," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4124–4134, 2022.

[2] M. Soleymanpour, M. T. Johnson, R. Soleymanpour, and J. Berry, "Synthesizing dysarthric speech using multi-speaker tts for dysarthric speech recognition," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7382–7386.

[3] B. Sriman, S. A. Silviya, S. K. Shriram, S. R. Kumar, R. Shriya, and A. Sujitha, "Virtual assistant for automatic emotion monitoring using perceived stress scale (pss)," in *2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2022, pp. 1529–1534.

[4] N. Yan, K. Wang, C. Pan, and K. K. Chai, "Private federated learning with misaligned power allocation via over-the-air computation," *IEEE Communications Letters*, vol. 26, no. 9, pp. 1994–1998, 2022.

[5] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[6] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios," *IEEE Access*, vol. 8, pp. 23 022–23 040, 2020.

[7] B.-S. Liang, "Ai computing in large-scale era: Pre-trillion-scale neural network models and exa-scale supercomputing," in *2023 International VLSI Symposium on Technology, Systems and Applications (VLSI-TSA/VLSI-DAT)*, 2023, pp. 1–3.

[8] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[9] H. Jiang, M. Liu, S. Sun, Y. Wang, and X. Guo, "Fedsyl: Computation-efficient federated synergy learning on heterogeneous iot devices," in *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, 2022, pp. 1–10.

[10] D. Wu, R. Ullah, P. Harvey, P. Kilpatrick, I. Spence, and B. Varghese, "Fedadapt: Adaptive offloading for iot devices in federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 20 889–20 901, 2022.

[11] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[12] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[13] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 1223–1231.

[14] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR*, vol. abs/1610.02527, 2016. [Online]. Available: http://arxiv.org/abs/1610.02527

[15] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[16] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1552–1564, 2021.

[17] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1698–1707.

[18] D. Wu, R. Ullah, P. Harvey, P. Kilpatrick, I. Spence, and B. Varghese, "Fedadapt: Adaptive offloading for iot devices in federated learning," *IEEE INTERNET OF THINGS JOURNAL*, vol. 9, no. 21, pp. 20 889–20 901, NOV 1 2022.

[19] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, 2021.

[20] D. Wen, K.-J. Jeon, and K. Huang, "Federated Dropout—A Simple Approach for Enabling Federated Learning on Resource Constrained Devices," *IEEE Wireless Communications Letters*, vol. 11, no. 5, pp. 923–927, May 2022.

[21] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *CoRR*, vol. abs/1812.00564, 2018. [Online]. Available: http://arxiv.org/abs/1812.00564

[22] J. Jeon and J. Kim, "Privacy-sensitive parallel split learning," in *2020 International Conference on Information Networking (ICOIN)*, 2020, pp. 7–9.

[23] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "Jointdnn: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 565–576, 2021.

[24] Y. Tian, Z. Zhang, Z. Yang, and Q. Yang, "Jmsnas: Joint model split and neural architecture search for learning over mobile edge networks," in *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2022, pp. 103–108.

[25] M. Krouka, A. Elgabli, C. B. Issaid, and M. Bennis, "Energy-efficient model compression and splitting for collaborative inference over time-varying channels," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2021, pp. 1173–1178.

[26] J. Zhang, L. Zhao, K. Yu, G. Min, A. Y. Al-Dubai, and A. Y. Zomaya, "A Novel Federated Learning Scheme for Generative Adversarial Networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 3633–3649, 2024.

[27] X. Liu, Y. Deng, and T. Mahmoodi, "Wireless distributed learning: A new hybrid split and federated learning approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2650–2665, 2023.

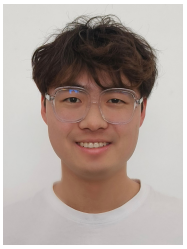
[28] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," *CoRR*, vol. abs/1804.08333, 2018. [Online]. Available: http://arxiv.org/abs/1804.08333

[29] C. Qiu, X. Wang, H. Yao, J. Du, F. R. Yu, and S. Guo, "Networking integrated cloud–edge–end in iot: A blockchain-assisted collective q-learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 694–12 704, 2021.

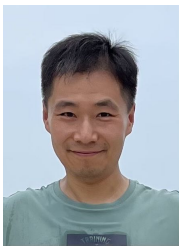
Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSE.2025.3544313, IEEE Transactions on Network Science and Engineering

[30] X. Deng, J. Li, C. Ma, K. Wei, L. Shi, M. Ding, and W. Chen, "Low-latency federated learning with dnn partition in distributed industrial iot networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 3, pp. 755–775, 2023.
[31] W. Wu, M. Li, K. Qu, C. Zhou, X. S. Shen, W. Zhuang, X. Li, and W. Shi, "Split learning over wireless networks: Parallel design and resource management," *IEEE Journal on Selected Areas in Communications*, vol. 41, pp. 1051–1066, 2022.
[32] E. Erdoğan, A. Küpçü, and A. E. Çiçek, "Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning," in *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, ser. WPES'22. New York, NY, USA: Association for Computing Machinery, 2022, p. 115–124. [Online]. Available: https://doi.org/10.1145/3559613.3563201
[33] H. Oh and Y. Lee, "Exploring Image Reconstruction Attack in Deep Learning Computation Offloading," in *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, ser. EMDL '19. New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 19–24.
[34] J. Neera, X. Chen, N. Aslam, K. Wang, and Z. Shu, "Private and utility enhanced recommendations with local differential privacy and gaussian mixture model," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 4151–4163, 2023.
[35] M. Wu, G. Cheng, P. Li, R. Yu, Y. Wu, M. Pan, and R. Lu, "Split learning with differential privacy for integrated terrestrial and non-terrestrial networks," *IEEE Wireless Communications*, pp. 1–8, 2023.
[36] E. Boshkovska, D. W. K. Ng, N. Zlatanov, and R. Schober, "Practical non-linear energy harvesting model and resource allocation for swipt systems," *IEEE Communications Letters*, vol. 19, pp. 2082–2085, 2015.
[37] D. Alqahtani, Y. Chen, W. Feng, and M.-S. Alouini, "A new non-linear joint model for rf energy harvesters in wireless networks," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 895–907, 2021.
[38] K. Shibasaki, S. Fukuzaki, and M. Ikehara, "4k real time image to image translation network with transformers," *IEEE Access*, vol. 10, pp. 73 057–73 067, 2022.
[39] B. n. al-din abed, B. Z. Kamil, M. A. Hameed, and J. N. Abdullah, "Using cardano's method for solving cubic equation in the cryptosystem to protect data security against cyber attack," in *2020 2nd Annual International Conference on Information and Sciences (AiCIS)*, 2020, pp. 127–131.
[40] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading Dependent Tasks in Mobile Edge Computing with Service Caching," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Jul. 2020, pp. 1997–2006.
[41] Z. Zhou, Q. Wu, and X. Chen, "Online Orchestration of Cross-Edge Service Function Chaining for Cost-Efficient Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, Aug. 2019.
[42] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming, version 2.0 beta," 2013.
[43] S. P. Boyd and L. Vandenberghe, "Convex optimization," *IEEE Transactions on Automatic Control*, vol. 51, pp. 1859–1859, 2004.
[44] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
[45] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.
[46] Q. Qi, A. Minturn, and Y. Yang, "An efficient water-filling algorithm for power allocation in ofdm-based cognitive radio systems," in *2012 International Conference on Systems and Informatics (ICSAI2012)*, 2012, pp. 2069–2073.



Yao Wen obtained a bachelor's degree from the College of Information Science and Technology, Nanjing Forestry University, Nanjing, China, in 2021, and a master degree from the School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China, in 2024. He is pursuing the Ph.D. degree at the School of Intelligent Software and Engineering, Nanjing University (Suzhou Campus). His research interests include edge computing and wireless communications.



Guopeng Zhang received the Ph.D. degree from the School of Communication Engineering, Xidian University, Xi'an, China, in 2009. In 2009, he joined the China University of Minin and Technology, Xuzhou, China, where he is currently a Professor with the School of Computer Science and Technology. He has authored or coauthored more than 60 journal and conference papers. His main research include wireless sensor networks and machine learning.

Kezhi Wang received the Ph.D. degree in engineering from the University of Warwick, U.K. He was with the University of Essex and Northumbria University, U.K. Currently, he is a Senior Lecturer with the Department of Computer Science, Brunel University London, U.K. His research interests include wireless communications, mobile edge computing, and machine learning.

Kun Yang received his PhD from the Department of Electronic & Electrical Engineering of University College London (UCL), UK. He is currently a Chair Professor of University of Essex, UK and Nanjing University. His main research interests include wireless networks and communications, communication-computing cooperation, and new AI (artificial intelligence) for wireless. He has published 500+ papers and filed 50 patents. He serves on the editorial boards of a number of IEEE journals (e.g., IEEE WCM, TVT, TNB). He is a Deputy Editor-in-Chief of IET Smart Cities Journal. He has been a Judge of GSMA GLOMO Award at World Mobile Congress – Barcelona since 2019. He was a Distinguished Lecturer of IEEE ComSoc (2020-2021), a Recipient of the 2024 IET Achievement Medals and the Recipient of 2024 IEEE CommSoft TC's Technical Achievement Award. He is a Member of Academia Europaea (MAE), a Fellow of IEEE, a Fellow of IET and a Distinguished Member of ACM.

Copyright © 2025 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.