ESFL: Efficient Split Federated Learning over Resource-Constrained Heterogeneous Wireless Devices

Guangyu Zhu, Yiqin Deng, *Member, IEEE*, Xianhao Chen, *Member, IEEE*, Haixia Zhang, *Senior Member, IEEE*, Yuguang Fang, *Fellow, IEEE*, Tan F. Wong, *Senior Member, IEEE*

***Abstract*—Federated learning (FL) allows multiple parties (distributed devices) to train a machine learning model without sharing raw data. How to effectively and efficiently utilize the resources on devices and the central server is a highly interesting yet challenging problem. In this paper, we propose an efficient split federated learning algorithm (ESFL) to take full advantage of the powerful computing capabilities at a central server under a split federated learning framework with heterogeneous end devices (EDs). By splitting the model into different submodels between the server and EDs, our approach jointly optimizes user-side workload and server-side computing resource allocation by considering users' heterogeneity. We formulate the whole optimization problem as a mixed-integer non-linear program, which is an NP-hard problem, and develop an iterative approach to obtain an approximate solution efficiently. Extensive simulations have been conducted to validate the significantly increased efficiency of our ESFL approach compared with standard federated learning, split learning, and splitfed learning.**

***Index Terms*—Distributed machine learning, federated learning, split learning, wireless networking.**

I. Introduction

In recent years, machine learning (ML) has attracted intensive attention in many fields and numerous artificial intelligence (AI) applications, such as computer vision, smart health, connected and autonomous driving, information access control, and security surveillance [1]. According to Cisco [2], there were nearly 850 zettabyte of data generated by people, machines, and things at the network edge in 2021. It is definitely infeasible to simply send this huge data volume to a central server to process or compute. To do so, a tremendous network bandwidth is required, incurring intolerable latency. Hence, distributed machine learning algorithms have been developed to cope with the aforementioned challenges for large geographically distributed volume of data by distributing the ML workloads to EDs [3]. Moreover, awareness and concerns about users' privacy have been raised in the digitalized world [4]. Following the Privacy-by-Design (PbD) principle, the best way to achieve user privacy is not to disclose raw data, and so it would be more effective for EDs not to share private raw data as much as possible during the machine learning process.

Federated learning (FL), a privacy-preserving distributed ML technique enables EDs to collaboratively learn a global ML model without sharing their raw data, consequently reducing the requirement of communication bandwidth between EDs and a central server as well. The intuitive idea of privacy-preserving distributed ML was investigated independently by some researchers Xu et al. [5]–[7], Shokri et al. [8], Gong et al. [9]. McMahan et al. [10] first constructed a distributed ML framework based on decentralized datasets held by different users, and coined their algorithm as FL. The original FL algorithm updates ML models on EDs locally, and aggregates the updated ML models to obtain the global model at a central server remotely, in contrast to the traditional ML requiring EDs to upload their private data to where the ML model is trained. Since FL retains users' private data on EDs without sharing raw data with the server, the server only aggregates users' locally updated models, and thus the privacy of end users is naturally preserved to some extent. Besides, as all training processes are performed by the EDs, the computation and communication capacities of EDs can also be utilized. FL provides various advantages, such as data privacy preservation,

Guangyu Zhu and Tan F. Wong are with Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA. (e-mail: gzhu@ufl.edu, twong@ece.ufl.edu).

Yiqin Deng and Haixia Zhang are with School of Control Science and Engineering and with Shandong Key Laboratory of Wireless Communication Technologies, Shandong University, Jinan 250061, Shandong, China (e-mail: yiqin.deng@email.sdu.edu.cn; haixia.zhang@sdu.edu.cn).

Xianhao Chen is with Department of Electrical and Electronic Engineering, University of Hong Kong, Pok Fu Lam, Hong Kong, China (e-mail: xchen@eee.hku.hk).

Yuguang Fang is with Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, China (e-mail: my.fang@cityu.edu.hk).

This work was supported in part by US National Science Foundation under grant CNS-2106589.

Corresponding author: Yiqin Deng.

reduced communication latency, and enhanced learning performance [2]. However, pushing all training workload to EDs sometimes is impracticable. Training complex ML models often consumes unacceptable amount of training memory and computing power on Internet of Things (IoT) devices with limited communication and computing resources, and incurs intolerable latency. Besides, the ML model size sometimes reaches GBs and even TBs(e.g., GPT-1 has 0.12 billion parameters, GPT-2 has 1.5 billion parameters, whereas GPT-3 has more than 175 billion parameters), resulting in a significant communications burden duo to the need of frequently uploading local ML models.

To cope with the dilemma between insufficient ED resources and complicated ML models, we leverage another ML technique, called split federated learning (SFL) [11], which introduces model splitting from split learning (SL) [12] to FL. The SFL framework was proposed by Thapa et al. [11] in 2020 to integrate federated learning with split learning, shown in Figure 1. Under SFL framework, the main server helps the training process for each client, and the Fed server is tasked with the aggregation of all locally updated models. The SFL approach presents a compelling advantage for resource-constrained environments, since the main server undertake parts of local training workload.

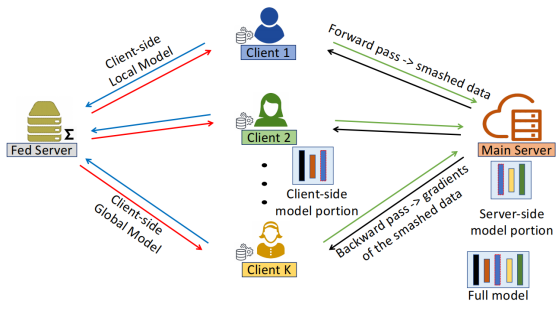


Fig. 1. The architecture of splitfed learning (SFL) system.

From the conventional FL and SFL implementations across heterogeneous devices, we have identified several limitations that can significantly impede the system's overall performance and efficiency. In synchronous FL and SFL, the aggregation process is inherently constrained by the pace of the slowest participant due to the necessity for the server to collect updated local gradients from all selected EDs. This synchronicity results in a scenario where devices with abundant computational resources are rendered idle as they wait for the transmission of local models from less capable devices, often referred to as "stragglers". This inefficiency is primarily attributed to the varying and unpredictable communication and computational capabilities of heterogeneous EDs. Therefore, addressing resource heterogeneity (RH), also called the system heterogeneity in FL, is critical to enhancing the efficiency and overall feasibility of SFL.

In this paper, we propose a novel *efficient split federated learning* algorithm (ESFL) to boost training efficiency and performance by considering the privacy-preserving constraints. The cornerstone of ESFL lies in its proactive strategic utilization of heterogeneity in system resources and device capabilities. Unlike standard synchronous FL and SFL, our approach involves the server in the training phase but dynamically adjusts the distribution of user-side workload and server-side resources. The ESFL algorithm thereby capitalizes on the intrinsic resource variability across EDs to optimize ML outcomes and system efficiency.

Our major contributions in this paper are summarized as follows.

1) We design ESFL, a novel distributed machine learning framework, which significantly improves the training efficiency of SFL by taking ED heterogeneity into consideration.
2) We formulate a mixed-integer non-linear program (MINLP) by jointly considering the allocation of user-side workload (model separation) and server-side resource, and develop an iterative optimization algorithm to find a suboptimal solution with a low time complexity.
3) We evaluate the performance of our ESFL approach through extensive simulations compared to the state-of-the-art methods such as FL, SL and SFL, and demonstrate the superiority of the proposed ESFL framework.

The remainder of this paper is organized as follows. In Section II, we discuss related works about FL, SFL, and RH. In Section III, we expound on the framework and system model of ESFL. In Section IV, we present the optimization problem formulation and the solution to the joint resource allocation and model separation problem. In Section V, results of extensive simulations and experiments are presented. In Section VI, we summarize the proposed algorithm and experimental results and conclude the paper.

II. Related Works

There exist quite extensive research on splitting learning (SL) and federated learning (FL) in the current literature. In this section, we will concentrate on closely related works to review. In [13], Ang et al. offered a robust architecture for FL to increase communication efficiency by reducing transmission noise in wireless networks. To address the communications overhead in FL, Wang et al. [14] developed an algorithm, named Communication Mitigated Federated Learning (CMFL), to eliminate irrelevant local model updates that were trained over users' biased data. It should be noted that the aforementioned

two methods only address communication efficiency in FL, as all training processes are executed by users at EDs. FedMMD was proposed in [15] to reduce communication and computing costs during the local training process in FL using a two-stream model with Maximum Mean Discrepancy (MMD) to replace the local training for a single model in FedAVG [10]. However, while this method reduces the number of communication rounds, it concurrently increases the user-side workload due to the need of computing the MMD loss functions. Shi et al. [16] address both communication and computing resource heterogeneities in wireless FL by providing a joint device scheduling and resource allocation strategy. Despite this integrated approach, the limitation of FL is still evident, as the user-side workload cannot be reduced. There exist also some other innovative approaches to ease the work load for communications and computing for federated learning. Watanabe et al. [17] and Chen et al. [18] leveraged wireless mesh networks to either facilitate communications or reduce communications traffic. Guo et al. [19], [20] utilized edge nodes and federated reinforcement learning to help resource-constrained D2D devices in industrial IoT systems and 5G networks, which can address the device heterogeneity issue to some extent.

In [11], Thapa et al. designed a novel framework, called *split federated learning* (SFL), to take advantage of parallel training among different users in FL and the model splitting in SL to reduce the computing workload on EDs. In [21], Gao et al. implemented SFL and evaluated the performance on IoT devices. In [22], Lin et al. developed an efficient parallel split learning algorithm by applying the last-layer gradient aggregation to reduce communication and computing overheads in SL. In [23], Wu et al. designed a resource allocation strategy for cluster-based SFL. In [24], Kim et al. devised a bargaining game to negotiate the cut layer in personalized parallel SL. All aforementioned literature on SFL overlook the crucial aspect of the joint consideration of both user-side resource and workload heterogeneities. This oversight is evidenced by the fact that the cut layer remains the same at all EDs. However, it is obvious that adjusting cut layers can significantly change user-side workload distribution.

To reduce the computing and communication workload at EDs when considering RH in FL, Sattler et al. [25] designed a novel model compression algorithm to extend the commonly used top-k gradient sparsification to FL to compress both model uploading and downloading. However, since model compression inevitably results in performance degradation, it is advisable to maintain the integrity of the model architecture throughout the training process. With the emergence of edge computing [26], utilizing the computing resources at the both the central server and edges can be leveraged to reduce workload at EDs. For instance, in [27], Wang et al. proposed to enable EDs to collaborate with edge nodes by exchanging model parameters to reduce the user-side workload, and to apply deep reinforcement learning to optimize the operations of multi-access edge computing (MEC), caching, and communications. Several researchers attempted to leverage edge nodes to assist FL by fully uploading local training tasks to trusted edge nodes to reduce the user-side workloads [28]. Unfortunately, the trustworthiness of training edge nodes is often hard to guarantee. Our ESFL algorithm introduce an integrated strategy for the allocation of server-side computing resource and user-side training workloads. This approach is designed to accommodate the variations in computing and communication resources inherent in heterogeneous EDs. We will present the detailed design next.

III. Efficient Split Federated Learning

A. Motivation

While SFL only address the resource constraints at EDs, it does not account for the variations in data distribution (DH) and resource availability (resource heterogeneity or RH) inherent in EDs. Within the SFL framework, clients or EDs are required to partition their models following an identical structure, leading to a scenario where the communication and computational workload on the client side is influenced solely by the volume of data. In contrast, our Efficient Split Federated Learning (ESFL) takes a holistic approach, considering both the client-side workload and the server-side computing resource allocation, which is designed to mitigate "stragglers" problem in FL, thereby enhancing the training efficiency of SFL. It is worth noting that our method is a scheme for joint optimization of resource and workload allocation, which can be integrated with any user selection algorithms presently existed in FL. The ESFL framework demonstrates the capability to increase training efficiency across the board, independent of the particular user selection algorithm integrated in the framework. This underscores the inherent adaptability and effectiveness of our proposed scheme in enhancing the training efficacy of FL.

B. ESFL framework

In this subsection, we elaborate our framework of ESFL consist of four components, namely, *split training*, *federated aggregation*, *communication model*, and *resource allocation*. We assume that an ML task is composed of multiple training rounds. The FL server first initializes ML model in the initial round of training and the subsequent training rounds. The one-round training procedure is given as follows.

1) **User selection**: The server selects users randomly from the *available users* (EDs), who are willing and ready to participate this round of training (these users are called the *selected users*).
2) **Model splitting and resource allocation**: The server first acquires the information of the selected EDs, including data amount, available communication, and computing resources. Then, the server jointly splits the ML model into user-side models and server-side models (the so-called *cut layer decision*) and allocates adequate server-side resources based on the users' information and the ML model structure for the *selected user*.
3) **Model distribution**: The server distributes user-side models (with the corresponding architectures) to the *selected users*.
4) **Split training**: The server and all *selected users* then collaboratively update both user-side and server-side ML models simultaneously and the server determines the ML hyperparameters, such as learning rate, data batch size, and local training epochs.
5) **Federated aggregation**: After repeating several epochs of the **split training** step, all selected users that have finished their training upload their updated user-side ML models to the server, and then the server generates an updated global ML model based on the user-side ML models collected from the *selected users* and the corresponding server-side ML models at the server.

In what follows, we will provide more details for the whole procedure below.

1) Split Training: Different from federated learning, which only relies on EDs to update local ML models, we split the local training across EDs and the server. Similar to the *SplitFed* [11], local training is repeatedly conducted ϵ_i epochs for user i before one-round global ML model aggregation at the server. The user-side ML model for user i is $\mathbf{W}_{u,i}^{r,e}$ at epoch e in the r-th round. We denote the updated local user-side ML model at the epoch $e+1$ as

$$\mathbf{W}_{u,i}^{r,e+1} = BP(\mathbf{W}_{u,i}^{r,e}, \rho_r, \nabla \mathbf{A}_{l_i}^{r,e}), \tag{1}$$

where BP is the backpropagation algorithm, ρ_r is the local learning rate in the r-th round, and $\nabla \mathbf{A}_{l_i}^{r,e} = \mathbf{A}_{l_i}^{r,e} - \hat{\mathbf{A}}_{l_i}^{r,e}$ is the activation difference, which is the loss value for BP, and $\hat{\mathbf{A}}_{l_i}^{r,e}$ is the updated activation calculated by the server using backpropagation algorithm with the shared labels and the estimated labels.

The server-side ML model for user i at epoch $e+1$ in the r-th round is given by

$$\mathbf{W}_{s,i}^{r,e+1} = BP(\mathbf{W}_{s,i}^{r,e}, \rho_r, \nabla \ell(\mathbf{W}_{s,i}^{r,e}, \hat{\mathbf{Y}}, \mathbf{Y})). \tag{2}$$

TABLE I
SUMMARY OF IMPORTANT NOTATIONS OF ESFL

Notation	Description
S	The set of selected users
R	The total training rounds
ϵ_i	The number of local training epochs of user i
$\mathbf{W}_{u,i}^{r,e}$	The user-side model for user i at epoch e at round r
$\mathbf{W}_{s,i}^{r,e}$	The server-side model for user i at epoch e at round r
$\mathbf{A}_{l_i}^{r,e}$	The activation calculated by the user i at epoch e at round r using forward propagation algorithm
η	The step size factor of the federated aggregation
N	The number of total one-round training data samples
n_i	The number of one-round training data samples for user i
$\mathbf{W}^r$	The global model at round r
$\mathbf{W}_i^r$	The cascaded local model generated by user $\mathbf{W}_{u,i}$ and server-side model $\mathbf{W}_{s,i}$ at round r
b_i^u	Uploading data rate of user i
b_i^d	Downloading data rate of user i
B_i	The bandwidth allocated to user i
P_i^u	The uplink transmission power of user i
P_i^d	The downlink transmission power of user i
γ_i^u	The uplink channel gain of user i
γ_i^d	The downlink channel gain of user i
N_0	The noise power density
T	The total training time
$T_{r,i}$	The r-th round training time for user i
T_i^U	The r-th round local model uploading time for user i
T_i^D	The r-th round global model downloading time for user i
T_i^e	The split training time at epoch e for user i
T^{agg}	The federated aggregation time for the server at round r
M_i^{li}	The local model uploading/ downloading time for user i
$t_i^{e,c}$	The user-side computing time for user i at epoch e
$t_i^{e,b}$	The uploading time of the user-side activation for user i at epoch e
$t_i^{e,C}$	The server-side computing time for user i at epoch e
$t_i^{e,B}$	The downloading time of the server-side updated activation for user i at epoch e
D_c^l	The user-side computing workload of the cut layer l for training one sample
D_b^l	The user-side communication workload of the cut layer l for training one sample
D	The computing workload for training one data sample
c_i	The local available computing resource for user i
C_i	The allocated server-side computing resource to user i
x_i	The cut layer indicator vector of user i
s_i	The available storage space for user i
m_i	The available memory space for user i

To update the server-side ML model, we use the loss function $\ell(\mathbf{W}_{s,i}^{r,e}, \hat{\mathbf{Y}}, \mathbf{Y})$, where $\hat{\mathbf{Y}}$ is the estimated result computed by the server-side ML model and $\mathbf{Y}$ is the true label shared by the user. The estimated result $\hat{\mathbf{Y}}$ is the output of the current server-side ML model $\mathbf{W}_{s,i}^{r,e}$ and the input data $\mathbf{A}_{l_i}^{r,e}$ is the activation generated by the user i's samples and user-side ML model. Split training divides the local training into four parts in FL, *i.e.*, user-side forward propagation, server-side forward propagation, server-side backpropagation, and user-side backpropagation. The entire split training procedure is shown in Figure 2.

The pseudocode for our ESFL is shown in Algorithm 1 and Algorithm 2. The split training is shown in Algorithm 1, while Step 2 is to let users send activations of

Algorithm 1: SplitUpdate

Input: At epoch e of round r, user-side model $\mathbf{W}_{u,i}^{r,e}$, server-side model $\mathbf{W}_{s,i}^{r,e}$, and local learning rate ρ;
Output: Updated user-side $\mathbf{W}_{u,i}^{r+1,e}$ and server-side model $\mathbf{W}_{s,i}^{r+1,e}$;
1 User i forwards propagation of $\mathbf{W}_{u,i}^{r,e}$ and generate activation $\mathbf{A}_{l_i}^{r,e}$ and the label vector $\mathbf{Y}$;
2 User i sends $\mathbf{A}_{l_i}^{r,e}$ at the cut layer and $\mathbf{Y}$ to the server;
3 The server uses backpropagation to calculate the server-side gradient $\nabla \ell(\mathbf{W}_{s,i}^{r,e}, \mathbf{Y})$ and the activation difference $\nabla \mathbf{A}_{l_i}^{r,e}$;
4 The server-side model update: $\mathbf{W}_{s,i}^{r,e+1} = BP(\mathbf{W}_{s,i}^{r,e}, \rho, \nabla \ell(\mathbf{W}_{s,i}^{r,e}, \mathbf{Y}))$;
5 The server sends $\nabla \mathbf{A}_{l_i}^{r,e}$ to user i;
6 The user-side model update: $\mathbf{W}_{u,i}^{r,e+1} = BP(\mathbf{W}_{u,i}^{r,e}, \rho, \nabla \mathbf{A}_{l_i}^{r,e})$

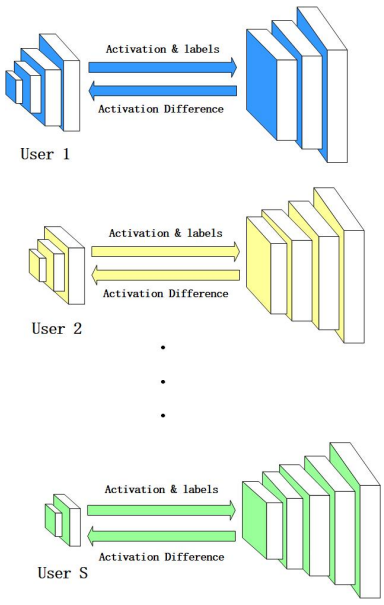


Fig. 2. The split training procedure, where all selected users simultaneously transmit activation data and labels to the server, and the server sends back the corresponding activation difference.

the cut layer together with the label to the server.

2) Federated Aggregation: In ESFL, we leverage the *FedAVG* algorithm [10] to aggregate multiple user-side and server-side updated ML models using η as a step size factor. We call this aggregation method *federated aggregation*, which mitigates training oscillations by leveraging the global ML model in the previous round. This differs from FedAVG which only aggregates the current-round updated local ML models to generate the next-round global ML model. The architecture of federated aggregation is shown in Figure 3. We define the whole ML model trained by user i in round r as ω_i^r and the global model at round r is $\mathbf{W}^r$. The update of the global ML model at round $r+1$ is

$$\begin{aligned}\mathbf{W}^{r+1} &= \mathbf{W}^r - \eta * (\mathbf{W}^r - \mathbf{W}_*^r) \\ &= \mathbf{W}^r - \eta * \left(\mathbf{W}^r - \sum_i \frac{n_i * \mathbf{W}_i^r}{N}\right),\end{aligned} \tag{3}$$

where $N = \sum_i n_i$ indicating the number of total training samples.

The whole training model is split into two parts, namely, user-side model $\mathbf{W}_{u,i}^r$ trained by a user-side ED and the server-side model $\mathbf{W}_{s,i}^r$ trained by a virtual server v_i. All virtual servers are either virtual machines or containers at the central server, which are allocated with corresponding computing and communication resources according to the model training demands at this round. Thus, after concatenating the user-side ML model and the server-side ML model, for each selected user, the concatenated ML model will have the same architecture as the global ML model. The updated concatenated ML model for user i at round $r+1$ is

$$\mathbf{W}_i^{r+1} \leftarrow \{\mathbf{W}_{u,i}^{r+1}, \mathbf{W}_{s,i}^{r+1}\}. \tag{4}$$

In Algorithm 2, the resource allocation scheme and cut layer decision based on idle resource states of selected users are shown in Step 5. In this section, we focus on synchronous federated learning, where the aggregation of all selected user-side ML models and server-side models can only be conducted after one-round training is finished.

3) Communication Model: Since we only intend to demonstrate the effectiveness of our proposed ML scheme, we will adopt a simple communication system model for our study. For uplink and downlink transmissions, we assume uploading and downloading transmission bandwidth are equal. Specifically, the orthogonal multiple access (OMA) techniques are adopted where each user can be allocated with one orthogonal spectrum band for the needed data transmissions determined by

Algorithm 2: Efficient Split Federated Learning

Input: The number of model aggregation round R, global learning rate η, local learning rate ρ, and user i's number of local epochs ϵ_i;

Output: Final Updated global model $\mathbf{W}^{R+1}$;

Initialize the global model parameters $\mathbf{W}$;
for $r = 1, 2, ..., R$ **do**
  The server randomly selects users S to participate the r-th round training;
  The server acquires states of available computing resource c^*, communication resource (uplink data rate) b^*, memory m^*, and storage space s^* from S;
  The server allocates its resources C^* and B^* and determines cut layer l^* based on c^* and b^*;
  for *selected user* $i = 1, 2, ..., |S|$ **do**
    Send $\mathbf{W}_{u,i}^r$ to user i based on l_i and $\mathbf{W}^{r-1}$;
    for *local epoch* $e = 1, 2, ..., \epsilon_i$ **do**
      $\mathbf{W}_{u,i}^{r,e+1}, \mathbf{W}_{s,i}^{r,e+1} \leftarrow$**SplitUpdate**$(\mathbf{W}_{u,i}^{r,e}, \mathbf{W}_{s,i}^{r,e}, \rho)$;
    end
    User i sends back the updated $\mathbf{W}_{u,i}^r$ to the server;
  end
  Server-side model update: $\mathbf{W}_s^{r+1} = \mathbf{W}_s^r - \eta \sum_i \frac{n_i}{N} \nabla \mathbf{W}_{s,i}^r$;
  User-side model update: $\mathbf{W}_u^{r+1} = \mathbf{W}_u^r - \eta \sum_i \frac{n_i}{N} \nabla \mathbf{W}_{u,i}^r$;
end

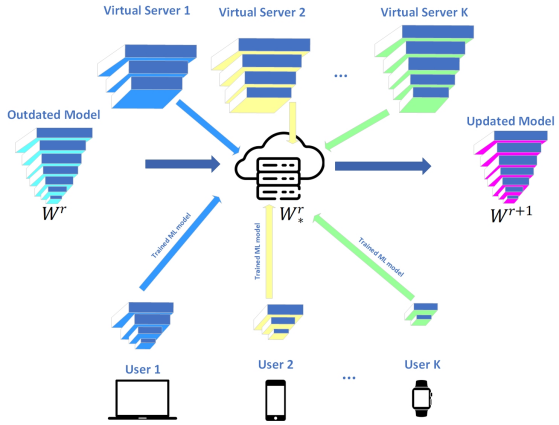


Fig. 3. This figure illustrates the federated aggregation, where the previous global ML model is $\mathbf{W}^r$, the aggregated global ML model is $\mathbf{W}_*^r$ and the updated global ML model is $\mathbf{W}^{r+1}$

the server (*e.g.*, the base station). The uploading and downloading data rate for user i is given by:

$$b_i^u = B_i \log\left(1 + \frac{P_i^u \gamma_i^u}{B_i N_0}\right), \qquad b_i^d = B_i \log\left(1 + \frac{P_i^d \gamma_i^d}{B_i N_0}\right), \tag{5}$$

where the communication bandwidth allocated to the user i is B_i, and the total available bandwidth is B where $\sum_{i=1}^{S} B_i \leq B$, γ_i^u is the uplink channel gain and γ_i^d is the downlink channel gain for user i, P_i^u and P_i^d are the uplink and downlink transmission powers, respectively, when the resource block B_i is used, where P_i^u, P_i^d, γ_i^u, and γ_i^d are predetermined constant values, and N_0 is the noise power density. We assume that the transmission environment is stationary during one training round. For example, EDs can be cameras in smart homes, whose deployment locations remain fixed for a relatively long period.

4) Workload and Resource Allocation: To reduce the training workload on user-side EDs, we split ML training into local training and server ML training. Thus, the next problem is how to appropriately determine user-side communication and computing workload, alongside the strategic allocation of computing resources on the server to different virtual servers. Due to the varying sizes of data collected by different users/EDs and the heterogeneous available resources on EDs, simply assigning the same amount of resources and randomly choosing the cut layers for different users will not help the training time and the training performance. To maximize the training efficiency, one should address a joint optimization of workload and resource.

We denote the total training time as T and the r-th round training time as T_r. To cope with the aforementioned joint optimization problem, we formulate the total training time as

$$T = \sum_{r=1}^{R} T_r = \sum_{r=1}^{R} \max_i T_{r,i}, \tag{6}$$

where $T_{r,i}$ is the r-th round training time for user i.

Since the times of individual training rounds are inde-

pendent, minimizing the total training time is equivalent to minimizing the training time for each training round. Thus, in the subsequent development, we will only focus on one round of training and omit the training round index r for notational simplicity. One-round training time is composed of four parts, namely, model distribution time T_i^D, model upload time T_i^U, model aggregation time T^{agg}, and training time T_i^e for local epoch e. Thus, the one-round training time can be expressed as

$$\begin{aligned} T_i &= T_i^U + T_i^D + \sum_{e=1}^{\epsilon_i} T_i^e + T^{agg}, \\ T_i^U &= \frac{M_i^{l_i}}{b_i^u},\ T_i^D = \frac{M_i^{l_i}}{b_i^d}, \end{aligned} \tag{7}$$

where b_i^U and b_i^D are the uplink and downlink data rate for user i as defined in (5). We denote the index of the cut layer for user i as l_i and ϵ_i as the number of local epochs for user i. The uploading and downloading workload are assumed to be M_i^l, equal to the user-side model size of user i. Since each selected user applies the same local dataset to train ML model and the idle resource is also stationary, to simplify the following optimization problem, in this paper, we assume that the training times of all epochs are constant in each round of training time.

Since both user-side EDs and the server participate in each epoch model updating, one-epoch training time contains four parts, namely, local/user-side computing time $t^{e,c}$, uploading time for activation $t^{e,b}$, remote/server-side computing time $t^{e,C}$, and downloading time for updated activation $t^{e,B}$. Thus, one epoch time can be denoted as

$$\begin{aligned} T_i^e &= t_i^{e,c} + t_i^{e,b} + t_i^{e,C} + t_i^{e,B} \\ &= \frac{w_i^{e,c}}{c_i} + \frac{w_i^{e,b}}{b_i} + \frac{W_i^{e,C}}{C_i} + \frac{w_i^{e,B}}{B_i} \\ &= \frac{\sum_l D_c^l x_i^l \cdot n_i}{c_i} + \frac{\sum_l D_b^l x_i^l \cdot n_i}{b_i^u} + \\ &\frac{(D - \sum_l D_c^l x_i^l) \cdot n_i}{C_i} + \frac{\sum_l D_b^l x_i^l \cdot n_i}{b_i^d}. \end{aligned} \tag{8}$$

The cut layer for user i is $l_i \in [1, 2, ..., L]$, where the whole ML neural network is composed of L layers, and the server-side model and the user-side model for user i are split at the l_i-th layer. To simplify the notation of workloads, we denote x_i^l as an indicator function, where $\sum_l x_i^l = 1$, and $x_i^l = 1$ indicates that l layer is chosen as the cut layer for user i while $x_i^l = 0$ indicates that l layer is not selected as the cut layer. The total computing workload for training one sample is D and the user-side computing workload for user i is $w_i^c = \sum_l D_c^l x_i^l \cdot n_i$, where D^l is the computing workload for one training sample for user-side EDs when the cut layer is l. The upload and download data size for user i is $w_i^b = \sum_l D_b^l x_i^l \cdot n_i$, where D_b^l is the size of activation data for one training sample while the cut layer is l. For user i, computing capability is c_i, uplink transmission rate is b_i^u, and downlink transmission rate is b_i^d. The computing resource that the server allocates to user i is C_i.

IV. Optimization and Solution Approach

The ultimate objective of the ESFL training is to minimize the total training time. However, since every round training time are independent, minimizing the total training time is equivalent to minimizing the total training time in each round including computing and communication time given by (as we mentioned earlier, we will omit the dependence of the round number r for notational simplicity)

$$\min T = \min \sum_r \max_i T_i. \tag{9}$$

To run Algorithm 2, we need to consider a few optimization problems for resource allocation, which should be solved during the ESFL training. The joint resource allocation and model splitting in our ESFL is a min-max optimization problem. To linearize the formulated optimization problem, we introduce an auxiliary variable T_{max}, which is no less than the training time for the straggler (*i.e.*, the client who takes the longest time to complete one-round training). Thus, our problem can be formulated as

$$\begin{aligned} &\min_{x_i^l, C_i} T_{max} \\ &s.t.\ T_i \le T_{max},\ i \in \{1, ..., S\} \\ &\sum_{i=1}^{S} C_i \le C_{total}, \\ &\sum_{l=1}^{L} x_i^l M_i^l \le s_i, \\ &\sum_{l=1}^{L} x_i^l m_i^l \le m_i, \\ &\sum_{l=1}^{L} x_i^l = 1,\ x_i^l \in \{0, 1\}. \end{aligned} \tag{10}$$

The total computing resource owned by the server is C_{total}, M_i^l is the data size of user i's user-side ML model, and s_i is the available storage space at user i. To compute the user-side model, m_i^l is the required memory space, and m_i is the available memory space for user i. We denote $x_i = \{x_i^1, ..., x_i^L\}$ as the cut layer indicator vector, where x_i^l indicates whether the ML model is split at layer l, in the sense that $x_i^l = 1$ when $l = l_i$, and $x_i^l = 0$ otherwise.

Algorithm 3: Alternative Optimization

Initialization: Allocating identical computing resource to all users, $C_i = \frac{C_{total}}{|S|}$;
while $C_i^{n-1} = C_i^n$ **do**
  Obtain the optimal $\{l_*\}$ of subproblem for cut layer decision for given $\{C_*\}$ by solving (11);
  Obtain the optimal $\{C_*\}$ of subproblem for resource allocation for the given cut layer decision $\{l_*\}$ by solving (12);
end
Output: $\{l_*\}$, $\{C_*\}$ for problem (10)

The alternative optimization algorithm is shown in Algorithm 3. For the notational convenience, we use $\{C_*\}$ to denote $\{C_1, C_2, ..., C_S\}$ and $\{l_*\}$ denote $\{l_1, l_2, ..., l_S\}$. The reason why we use an alternative optimization approach is that the cut layer decisions for different selected users lead to the variance at the user-side workload. Moreover, the allocation of server-side computing resource to individual one user will affect the availability of resources for others, given the fixed total capacity of server-side resources, where the joint optimization of worklaods and resources incurs the coupling effect for different users. To address this coupling effect, our ESFL algorithm transforms the optimization problem into a mixed-integer non-linear program (MINLP), which is typically NP-hard. To solve the problem efficiently, we decompose it into two subproblems and solve them iteratively. We construct the first subproblem for cut layer decision by treating computing resource allocation as fixed decision variables:

$$\begin{aligned} \min_{x_i^l} T_i^e = &\frac{\sum_l D_c^l x_i^l \cdot n_i}{c_i} + \frac{\sum_l D_b^l x_i^l \cdot n_i}{b_i^U} + \\ &\frac{(D - \sum_l D_c^l x_i^l) \cdot n_i}{C_i} + \frac{\sum_l D_b^l x_i^l \cdot n_i}{b_i^D} \\ s.t. \ &\sum_{l=1}^{L} x_i^l = 1, \ x_i^l \in \{0, 1\}, \\ &M_i^l \leq s_i, \\ &m_i^l \leq m_i. \end{aligned} \tag{11}$$

Leveraging the iterative optimization approach, the correlation between different users can be eliminated in the sense that we can focus on solving the first subproblem for each user independently, as shown in (11). This is because the cut layer decision for each user is independent of others when computing resource allocation is given. The resulting subproblem for cut layer decision can be easily solved by a linear programming (LP) solver or exhaustive search with the time complexity reduced from $\mathcal{O}(L^S)$ to $\mathcal{O}(SL)$.

Based on the determined cut layers, we construct the second subproblem for the resource allocation scheme for computing resources for user i as

$$\begin{aligned} \min_{C_i} \max T_i^e = &\frac{w_c^{l_i} \cdot n_i}{c_i} + \frac{w_b^{l_i} \cdot n_i}{b_i^u} + \\ &\frac{(D - w_c^{l_i}) \cdot n_i}{C_i} + \frac{w_b^{l_i} \cdot n_i}{b_i^d} \\ s.t. \ &\sum_{i=0}^{S} 0 \leq C_i \leq C_{total}, \end{aligned} \tag{12}$$

where communication and computing workloads for all users are constant since the cut layers l_i are predetermined by solving the previous subproblem. Plus, the downlink b_i^d, uplink communication resource b_i^u and user-side available computing resource c_i are constant. Therefore, the equation (12) can be abbreviated as:

$$\min_{C_i} \max T_i^e = \frac{a}{C_i} + b, \tag{13}$$

where a and b are constant. To solve this min-max problem, we assume there exists a variable K, where for all C_i, $K \geq \frac{a}{C_i} + b$. Then, we construct the equation(13) as a minimizing problem:

$$\begin{aligned} &\min K \\ s.t. \ &\sum_{i=0}^{S} 0 \leq C_i \leq C_{total}, \\ &K \geq \frac{a_1}{C_1} + b_1 \\ &K \geq \frac{a_2}{C_2} + b_2 \\ &\vdots \\ &K \geq \frac{a_{|S|}}{C_{|S|}} + b_{|S|}. \end{aligned} \tag{14}$$

When $C_i \geq 0$, T_i^e is a convex function ($\nabla^2 T_i^e \geq 0$). Then, a convex optimization solver [29] can be leveraged to solve this subproblem.

V. Experiments

In this section, we demonstrate that our ESFL can inherently offer similar performance under the same resource limitation for all users, while significantly reducing total training time (*time efficiency*). We then show the superior training performance with limited resources and limited training time (*model performance*). Finally, we validate our iterative optimization approach under various system settings.

A. Experimental Setup

For all following experiments, we evaluate the performance on image classification tasks over the common

dataset CIFAR-10 and leverage VGG13, VGG16 and VGG19 [30] framework as the neural network architecture to implement the distributed applications. We compare our ESFL with the alternatives such as FedAVG (FL), original split learning (SL), and splitfed learning (SFL).

Dataset: CIFAR-10 [31] [32] contains $50,000$ color training images and $10,000$ testing images with 32×32 resolution in 10 classes, with $6,000$ images per class. We assume that there are 100 users participating in the whole training process, while only 10 users are randomly selected to join one-round training. Under the assumption that all users' data samples are independently and identically distributed (IID), the data is shuffled and then partitioned into 100 clients with no replacement, every user owning 500 training samples.

Training Configuration: We use a distributed machine learning framework, similar to federated learning, which has several learning hyperparameters including local learning rate ρ_r, where $\rho_0 = 0.01$ and ρ_r is decaying as the round number r increases and the constant number of local epochs $\epsilon_i = 5$. Moreover, we introduce a global learning rate $\eta = 0.5$ to control the global model updating pace. According to the experimental results, when choosing a mini-batch size of 32, we can obtain a well trained model.

Neural Network Architecture: We deploy the VGG19 network [30] as the training model, which primarily consists of convolutional layers (CONV), fully-connected layers (FC), and softmax layer (SoftMax). We resize the input layer of the original VGG13, VGG16, and VGG19 from 224×224 to 32×32 to fit the CIFAR-10 dataset. The mini-batch size is set to 32. We present VGG19 architecture and workload of each layer in Table II.

TABLE II
VGG19 NETWORK ARCHITECTURE AND PARAMETERS

Layer	Layer size (MBs)	FP FLOPs (MBs)	Activation (MBs)
CONV1	0.0017	1.796	0.0655
CONV2	0.0369	37.749	0.0328
CONV3	0.0737	18.874	0.0328
CONV4	0.147	37.749	0.0164
CONV5	0.295	18.874	0.0164
CONV6	0.590	37.749	0.0164
CONV7	0.590	37.749	0.0164
CONV8	0.590	37.749	0.0082
CONV9	1.180	18.874	0.0082
CONV10	2.359	37.749	0.0082
CONV11	2.359	37.749	0.0082
CONV12	2.359	37.749	0.0020
CONV13	2.359	9.437	0.0020
CONV14	2.359	9.437	0.0020
CONV15	2.359	9.437	0.0020
CONV16	2.359	9.437	0.0010
FC1	102.760	2.097	4.08E-5
FC2	16.777	0.524	4.08E-5
FC3	4.096	0.131	1E-5
SoftMax	\	\	\

B. Model Performance

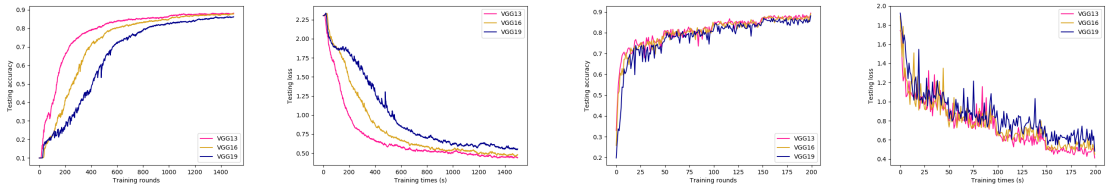
One key hyperparameter in our ESFL that affect the final convergence performance such as testing accuracy and loss is the number of training rounds, since we leverage **FedAVG** for all distributed ML algorithms except SL. For a fair comparison, we set the training threshold for VGG13 to 88%, VGG16 to 87.5% and VGG19 to 86.5% testing accuracy based on the worst converged accuracy. Three distributed ML algorithms (FL, SFL, ESFL) achieve the expected convergence performance at the 1500-th training round, and SL achieves the expected convergence performance at the 200-th training round. The testing performance results are shown in Figure. 4.

The rationals for choosing IID configuration rather than Non-IID in our model training process is to mitigate the impact of data distribution heterogeneities for fair training performance comparisons across different distributed ML frameworks.

C. Time Efficiency

In our simulation, at each round, the server randomly selects 10% users (the *selected users*) from *available users* to join one-round training. Since we assume that the server possesses sufficient but limited computing resources, in this experiment, the training server is installed with an A100 GPU with 130 teraFLOPs (TFLOPs) computing capability and 128 GigaBytes (GBs) memory space. We compare the time efficiency of our ESFL with original federated learning (FL) [10], split learning [33] and splitfed learning (SFL) [11].

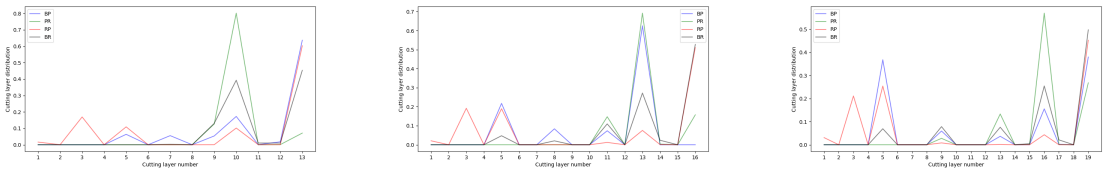
1) Resource Limitation: We separate the impacts of user-side communication and computing resource limitation by simulating four resource settings shown in Table III: **Both Poor (BP)** indicates that both communication and computing resources are highly limited at EDs, **Poorcom Richcmp (PR)** indicates that communication resources are highly limited while computing resources



(a) Testing accuracy and loss of VGG13, VGG16 and VGG19 using FL

(b) Testing accuracy and loss of VGG13, VGG16 and VGG19 using SL

Fig. 4. Testing accuracy and loss over CIFAR-10 testing dataset for FL, SFL, ESFL and SL using three different NN (VGG13, VGG16 and VGG19). Fair comparison are guaranteed by the required training rounds to achieve the convergence threshold.



(a) Layers distributions of VGG13

(b) Layers distributions of VGG16

(c) Layers distributions of VGG19

Fig. 5. Cut layer distributions (user-side workloads allocation) of three NNs under four different resource limitations using ESFL algorithm.

TABLE III
COMMUNICATION AND COMPUTING RESOURCE SETTINGS

	Communication(KBps)	Computing(TFLOPs)
BP	[10, 15, 20, 25]	[1.3, 1.95, 2.6, 3.25]
PR	[10, 15, 20, 25]	[6.5, 9.75, 13, 16.25]
RP	[50, 75, 100, 125]	[1.3, 1.95, 2.6, 3.25]
BR	[50, 75, 100, 125]	[6.5, 9.75, 13, 16.25]

are slightly limited (five times larger than that for the highly limited case), **Richcom Poorcmp (RP)** indicates that communication resources are slightly limited while computing resources are highly limited, and **Both Rich (BR)** indicates that both communication and computing resources are slightly limited. The selection process for each user-side resource setting is similar to Table III. For learning algorithmic implementation, we use original FL and SL, and SFL, which is similar to that for ESFL introduced in Section III.

Table IV presents the average one-round training and communication time and one-round communication time for different NNs under different resource scenarios using FL, SL, SFL and efficient split federated learning ESFL, respectively. Figure. 5 shows the allocation results of user-side training workload represented as cut layer distributions. The cut layer distribution represents the empirical probability of selecting layer l for user i in the total training rounds, which is $P_{i,l} = \sum_r^R \frac{x_{i,r}^l}{R}$, where $x_{i,r}^l$ is the cutting layer decision showing in equation 8 and $\sum_l P_{i,l} = 1$. The cut layer distribution combining with the amounts of user-side data indicates the allocated user-side computing and communication workload. Therefore, from cut layer distributions, as the user-side resource becomes more sufficient, our ESFL applies more identical cut layer distributions strategies for all NNs. For the results in those two tables, our ESFL algorithm significantly reduces one-round training and communication time under all circumstances. These advantages often stem from the dynamics between user-side computing and communication resource. In scenarios where local computing resource are poor whereas communication resource are rich (**RP**), ESFL remains fewer layers of user-side models by leveraging more on server-side computing power. Conversely, under the **PR** scenario, ESFL mitigates these limitations by remaining more layers of user-side model to rely less on communication. Comparing one-round training and communication time of SFL and FL, an intriguing phenomenon emerges. Although SFL leverages the server-side resource to accelerate training, improper user-side workload allocation (model separation) and server-side resource allocation lead to decreased time efficiency.

Table V indicates the total training and communication time for four ML algorithms to achieve the expected convergence performance. In the context of contrasting FL and SFL, it is imperative to acknowledge that although

TABLE IV
ONE-ROUND TRAINING AND COMMUNICATION TIME FOR DIFFERENT RESOURCE SETTINGS

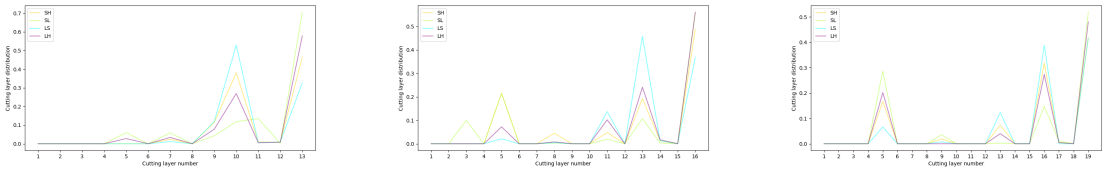
	NNs	Training and communication time(s)				Communication time(s)			
		FL	SL	SFL	ESFL	FL	SL	SFL	ESFL
BP	VGG13	40.916	226.444	42.501	**28.583**	11.182	123.365	21.968	**8.726**
	VGG16	52.960	254.787	48.476	**31.125**	**11.599**	115.900	20.284	12.828
	VGG19	63.096	282.426	53.231	**31.128**	**10.200**	107.575	18.253	16.216
PR	VGG13	18.351	148.283	27.775	**8.245**	13.300	127.126	23.043	**2.461**
	VGG16	20.933	143.427	27.065	**10.242**	13.401	114.513	20.616	**2.858**
	VGG19	23.700	141.961	27.532	**12.901**	14.017	105.299	18.927	**3.433**
RP	VGG13	34.128	130.896	29.229	**19.297**	**4.238**	50.829	8.430	10.961
	VGG16	45.713	156.239	35.828	**19.657**	**4.292**	45.390	7.585	11.135
	VGG19	58.275	181.994	42.161	**19.855**	**4.566**	42.756	7.019	11.123
BR	VGG13	10.937	73.141	14.334	**6.961**	5.005	51.208	9.202	**0.989**
	VGG16	12.929	75.028	15.194	**8.671**	4.674	45.801	8.118	**2.621**
	VGG19	15.968	78.177	16.220	**9.470**	5.428	42.736	7.549	**3.184**

TABLE V
TOTAL TRAINING AND COMMUNICATION TIME FOR DIFFERENT RESOURCE SETTINGS

	NNs	Training and communication time(s)				Communication time(s)			
		FL	SL	SFL	ESFL	FL	SL	SFL	ESFL
BP	VGG13	61,374	45,288	63,751	**42,874**	16,773	24,673	32,952	**13,089**
	VGG16	79,440	50,957	72,714	**46,687**	**17,398**	23,180	30,426	19,242
	VGG19	94,644	56,485	79,725	**46,692**	**15,300**	21,515	27,379	24,324
PR	VGG13	27,526	29,656	41,662	**12,367**	19,950	25,425	34,564	**3,691**
	VGG16	31,399	28,685	41,298	**15,363**	20,101	22,902	30,924	**4,287**
	VGG19	35,550	28,392	41,298	**19,351**	21,025	21,059	28,390	**5,149**
RP	VGG13	51,192	**26,179**	43,843	28,945	**6,357**	10,165	12,645	16,441
	VGG16	68,569	31,247	53,742	**29,485**	**6,438**	9,078	11,377	16,702
	VGG19	87,412	36,398	63,241	**29,782**	**6,849**	64,134	10,528	16,684
BR	VGG13	16,405	14,628	21,501	**10,441**	7,507	10,241	13,803	**1,483**
	VGG16	19,393	15,005	22,791	**13,006**	7,011	9,160	12,177	**3,931**
	VGG19	23,952	15,635	24,330	**14,205**	8,142	8,547	11,323	**4,776**

SFL, similar to ESFL, utilizes server-side computing resource, its overall performance is significantly influenced by the harmonization of user-side workload and server-side resource allocation. The lack of effective resource allocation strategies can result in inferior performance in SFL when compared to FL. Especially in **RP** and **BR**, when communication resource is notably limited in EDs, FL outperforms SFL. This performance discrepancy is attributed to the lack of effective allocating strategies, which results in inferior performance in SFL when compared to FL. Nevertheless, our ESFL exhibits a significant increase in efficiency compared to both FL and SFL across all tested scenarios. This provides the evidence that a well-conceived strategy for workload and resource allocation can markedly enhance the efficiency of the whole training process. While the model used here and other ML algorithms are not the state-of-the-art for this task, it does provide sufficient evidence to show that our ESFL can significantly reduce training latency and improve training efficiency by considering user-side resource heterogeneity.

2) Resource Heterogeneity: In this section, we assume there exist four different communication and computing resource settings to evaluate the efficiency of our ESFL algorithm under different heterogeneous scenarios. The detail of our resource simulation setting is shown



(a) Layers distributions of VGG13 (b) Layers distributions of VGG16 (c) Layers distributions of VGG19

Fig. 6. Cut layer distributions (user-side workloads allocation) of three NNs under four different resource heterogeneities using ESFL algorithm.

TABLE VI
COMMUNICATION AND COMPUTING RESOURCE SETTINGS

	Communication(KBps)	Computing(TFLOPs)
SH	[10, 15, 20, 25]	[1.3, 1.95, 2.6, 3.25]
SL	[10, 15, 20, 25]	[0.65, 1.3, 2.6, 4.55]
LS	[5, 10, 20, 35]	[1.3, 1.95, 2.6, 3.25]
LH	[5, 10, 20, 35]	[0.65, 1.3, 2.6, 4.55]

in Table VI, where in every training round, the available communication and computing conditions of the *selected users* are randomly chosen from resource options in one scenario. In particular, considering the fairness, the average resource amounts in different scenarios are equal, and only resource distributions are dissimilar to simulate different resource heterogeneity. Four heterogeneous scenarios are considered: **Small Heterogeneity (SH)**, implying that both communication and computing resource heterogeneity at EDs is small, **Smallcom Largecmp (SL)**, implying that communication heterogeneity is small while computing resource is large, **Largecom Smallcmp (LS)**, implying that communication heterogeneity is large while computing resource is small, and **Large Heterogeneity (LH)**, implying that both communication and computing resource heterogeneities are large. For instance, in **SH**, each user will randomly choose one communication condition from [10, 15, 20, 25] kiloBytes (KBps) and one computing condition from [1.3, 1.95, 2.6, 3.25] TFLOPs as their available resources, and the server will base on the resource information of the selected users to allocate appropriate server-side computing resource and make user-side cutting layer decision for all selected users. To simulate the training workload heterogeneity, we assume that all *available users* have heterogeneous but constant amounts of data samples, which are chosen from [200, 400, 600, 800].

Table VII presents the average one-round training and communication time and one-round communication time, for different NNs under different resource scenarios using FL, SL, SFL and ESFL, respectively. For the results in Table VII, our ESFL algorithm significantly reduces one-round training and communication time under all circumstances and is least affected by resource heterogeneity, where both communication and computing heterogeneity seriously impact the training efficiencies of the other three ML algorithms. It is noteworthy that as the distribution of resources approaches a state of greater uniformity (**SH**), the gap of time efficiency between SFL and ESFL is decreased. Conversely, with an increase in resource heterogeneity (**LH**), the performance differential between ESFL and SFL widens notably, which highlights the robustness of ESFL in diverse resource environments. For instance, the training latency of SFL in **LH** is increased by nearly two times compared with that in **SH** while training latencies of ESFL are nearly the same in all scenarios. The user-side training workload allocation results (cut layer distributions) is shown in Figure. 6. From this simulation result, ESFL algorithm demonstrates a trend implementing more uniform cut layer distributions across all NNs, under more identical resource distributions.

Table VIII presents the total training and communication time under different **RH**. It can be seen from the results that our ESFL method is significantly more efficient compared with the original FL, SL, and SFL in most scenarios, only except VGG13 in **SL**. Comparing communication time of FL in SL and LS, the performance of FL is markedly impacted by the **RH**. However, ESFL capitalizes on these heterogeneities through joint workload and resource allocation: in the environment with low communication heterogeneity (**SL**), it increases the user-side communication workload while decreases computing workload; in the environment with high communication heterogeneity (**LS**), it conversely allocates more computing workload to EDs. Therefore, our approach optimizes the time efficiency across diverse scenarios by adaptively leveraging **RH**.

D. Resource Allocation Convergence Analysis

From Section IV, the joint resource allocation and model splitting problem has been decomposed into two

TABLE VII
ONE-ROUND TRAINING AND COMMUNICATION TIME FOR DIFFERENT HETEROGENEITIES

	NNs	Training and communication time(s)				Communication time(s)			
		FL	SL	SFL	ESFL	FL	SL	SFL	ESFL
SH	VGG13	50.706	362.412	66.277	**34.043**	22.853	257.278	45.766	**6.170**
	VGG16	64.136	362.872	67.640	**40.658**	24.544	225.720	40.473	**11.093**
	VGG19	76.288	381.228	69.879	**43.074**	24.063	210.248	35.558	**22.941**
SL	VGG13	78.595	418.746	83.431	**55.912**	19.913	251.753	44.067	**17.867**
	VGG16	103.394	457.030	92.968	**58.576**	**21.957**	229.795	39.047	26.674
	VGG19	126.801	502.072	105.241	**61.022**	**22.728**	214.884	36.744	31.815
LS	VGG13	79.478	518.936	109.171	**37.625**	53.200	415.204	90.510	**9.832**
	VGG16	87.448	512.143	109.142	**48.904**	51.658	369.021	82.441	**10.685**
	VGG19	95.622	522.882	106.500	**54.211**	49.811	347.786	74.140	**19.720**
LH	VGG13	91.815	565.541	123.469	**61.865**	41.606	399.900	86.496	**14.211**
	VGG16	117.439	608.637	130.643	**73.708**	40.871	386.282	80.686	**17.929**
	VGG19	145.186	601.846	130.470	**75.520**	44.473	323.242	66.901	**42.704**

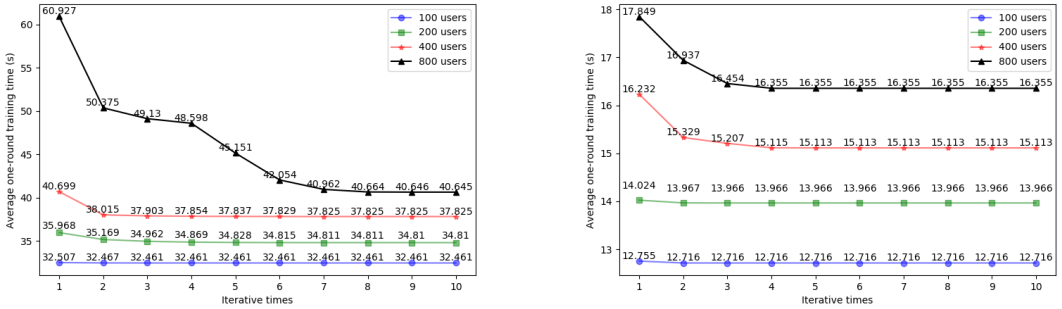
TABLE VIII
TOTAL TRAINING TIME TO ACHIEVE THE CONVERGENCE PERFORMANCE FOR DIFFERENT HETEROGENEITIES

	NNs	Training and communication time(s)				Communication time(s)			
		FL	SL	SFL	ESFL	FL	SL	SFL	ESFL
SH	VGG13	76,059	72,482	99,415	**51,064**	34,279	51,455	68,649	**9,255**
	VGG16	96,204	72,574	101,460	**60,987**	36,816	45,144	60,709	**16,639**
	VGG19	114,432	72,574	104,818	**64,611**	36,816	42,049	53,337	**34,411**
SL	VGG13	117,892	**83,749**	125,146	83,868	29,869	50,350	66,100	**26,800**
	VGG16	155,091	91,406	139,452	**84,217**	**32,935**	45,959	58,570	40,011
	VGG19	190,201	100,414	157,861	**91,533**	**34,092**	42,976	55,122	47,722
LS	VGG13	119,217	103,787	163,756	**56,437**	79,800	83,040	135,765	**14,748**
	VGG16	131,232	102,428	163,713	**73,356**	77,487	73,804	123,661	**16,027**
	VGG19	143,433	104,576	159,750	**81,316**	74,716	69,557	111,210	**29,580**
LH	VGG13	137,722	113,108	185,203	**92,797**	62,409	79,980	129,744	**21,316**
	VGG16	176,158	121,727	195,964	**110,562**	61,306	77,256	121,029	**26,893**
	VGG19	217,779	120,369	195,705	**113,280**	66,709	64,648	100,351	**64,056**

subproblems due to the time complexity, and we leverage an alternative optimization approach to solve those subproblems. To evaluate the convergence of our alternative method, we simulate iterative results at four users' scales in four resource scenarios (Table III), **100 users**, **200 users**, **400 users**, and **800 users**. The simulation results in Figure. 7 shows that even for the largest user scale (**800 users**) in all different resource scenarios, our alternative approach only needs 9 iterations to achieve convergence, while when the user scales are small, it only requires a few iterations to achieve convergence. Comparing average one-round training time across different scenarios, our method significantly enhances training efficiency, particularly in the resource-constrained scenario (**BP**).

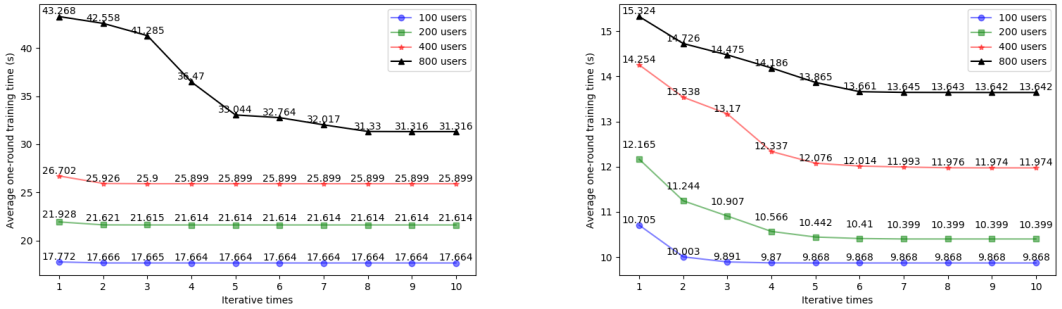
VI. CONCLUSION

In this paper, we have designed ESFL, a novel distributed training approach that tackles the resource heterogeneity inherent in both federated learning and split learning. Unlike previous methods in addressing data heterogeneity in FL, we have provided a new perspective by allocating appropriate server-side resources and user-side workload to effectively address the straggler problem in the synchronous FL framework. By evaluating the training efficiency for different ML algorithms under



(a) Average one-round training time in **BP**

(b) Average one-round training time in **PR**



(c) Average one-round training time in **RP**

(d) Average one-round training time in **BR**

Fig. 7. Average one-round training time using iterative optimization of VGG19 in four different resource scenarios shown in Table III.

different heterogeneous scenarios, we have performed extensive analysis and demonstrated the superiority of our proposed ESFL.

REFERENCES

[1] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys & Tutorials*, 2021.

[2] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2021.

[3] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–16, 2016.

[4] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[5] K. Xu, Y. Guo, L. Guo, Y. Fang, and X. Li, "Control of photo sharing over online social networks," in *2014 IEEE Global Communications Conference*. IEEE, 2014, pp. 704–709.

[6] K. Xu, H. Yue, L. Guo, Y. Guo, and Y. Fang, "Privacy-preserving machine learning algorithms for big data systems," in *2015 IEEE 35th international conference on distributed computing systems*. IEEE, 2015, pp. 318–327.

[7] K. Xu, Y. Guo, L. Guo, Y. Fang, and X. Li, "My privacy my decision: Control of photo sharing on online social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 2, pp. 199–210, 2017.

[8] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[9] Y. Gong, Y. Fang, and Y. Guo, "Privacy-preserving collaborative learning for mobile health monitoring," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[11] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8485–8493.

[12] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[13] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3452–3464, 2020.

[14] L. Wang, W. Wang, and B. Li, "Cmfl: Mitigating communication overhead for federated learning," in *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*. IEEE, 2019, pp. 954–964.

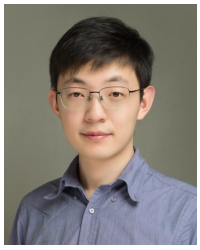
[15] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," in *2018 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2018, pp. 1–4.

[16] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained

wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2020.
[17] Y. Watanabe, Y. Kawamoto, and N. Kato, "A novel routing control method using federated learning in large-scale wireless mesh networks," *IEEE Transactions on Wireless Communications*, vol. 22, no. 12, pp. 9291–9300, 2023.
[18] X. Chen, G. Zhu, Y. Deng, and Y. Fang, "Federated learning over multihop wireless networks with in-network aggregation," *IEEE Transactions on Wireless Communications*, vol. 21, no. 6, pp. 4622–4634, 2022.
[19] Q. Guo, F. Tang, and N. Kato, "Federated reinforcement learning-based resource allocation in D2D-enabled 6G," *IEEE Network*, 2024, DOI : 10.1109/MNET.122.2200102.
[20] ——, "Federated reinforcement learning-based resource allocation for D2D-aided digital twin edge networks in 6G industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 5, pp. 7228–7236, 2023.
[21] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtepe, H. Kim, and S. Nepal, "End-to-end evaluation of federated learning and split learning for internet of things," *arXiv preprint arXiv:2003.13376*, 2020.
[22] Z. Lin, G. Zhu, Y. Deng, X. Chen, Y. Gao, K. Huang, and Y. Fang, "Efficient parallel split learning over resource-constrained wireless edge networks," *arXiv preprint arXiv:2303.15991*, 2023.
[23] W. Wu, M. Li, K. Qu, C. Zhou, W. Zhuang, X. Li, W. Shi *et al.*, "Split learning over wireless networks: Parallel design and resource management," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1051 – 1066, Feb. 2022.
[24] M. Kim, A. DeRieux, and W. Saad, "A bargaining game for personalized, energy efficient split learning over wireless networks," *arXiv preprint arXiv:2212.06107*, 2022.
[25] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.
[26] Y. Deng, X. Chen, G. Zhu, Y. Fang, Z. Chen, and X. Deng, "Actions at the edge: Jointly optimizing the resources in multi-access edge computing," *IEEE Wireless Communications*, vol. 29, no. 2, pp. 192–198, 2022.
[27] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.
[28] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.
[29] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 10.1.*, 2024. [Online]. Available: http://docs.mosek.com/latest/toolbox/index.html
[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[31] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
[32] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
[33] 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.

Guangyu Zhu received the B.Eng. degree from Xidian University, Xi'an, China, in 2019. Since 2019, he has been pursuing the Ph.D degree with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. His research interests include machine learning, wireless networks, and edge computing.

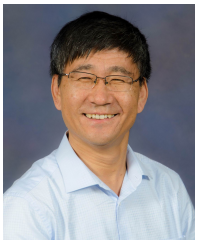
Yiqin Deng (Member, IEEE) received her M.S. degree in software engineering and her Ph.D. degree in computer science and technology from Central South University, Changsha, China, in 2017 and 2022, respectively. She is currently a Postdoctoral Research Fellow withthe School of Control Science and Engineering, Shandong University,Jinan, China. She was a visiting researcher at the University of Florida, Gainesville, from 2019 to 2021. Her research interests include edge/fog computing, Internet of Vehicles, and resource management.



Xianhao Chen (Member, IEEE) received the B.Eng. degree in electronic information from Southwest Jiaotong University in 2017, and the Ph.D. degree in electrical and computer engineering from the University of Florida in 2022. He is currently an assistant professor at the Department of Electrical and Electronic Engineering, the University of Hong Kong. He serves as an Associate Editor of ACM Computing Surveys. He received the 2022 ECE graduate excellence award for research from the University of Florida. His research interests include wireless networking, edge intelligence, and machine learning.

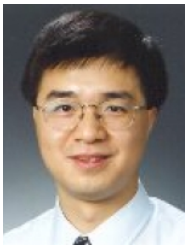
Haixia Zhang (Senior Member, IEEE) received the B.E. degree from the Department of Communication and Information Engineering, Guilin University of Electronic Technology, Guilin, China, in 2001, and the M.Eng. and Ph.D. degrees in communication and information systems from the School of Information Science and Engineering, Shandong University, Jinan, China, in 2004 and 2008, respectively.

From 2006 to 2008, she was with the Institute for Circuit and Signal Processing, Munich University of Technology, Munich, Germany, as an Academic Assistant. From 2016 to 2017, she was a Visiting Professor with the University of Florida, Gainesville, FL, USA. She is currently a Full Professor with Shandong University, Jinan, China. Dr. Zhang is actively participating in many professional services. She is/was an editor of the IEEE Transactions on Wireless Communications, IEEE Internet of Things Journal, IEEE Wireless Communication Letters, and China Communications and serves/served as Symposium Chairs, TPC Members, Session Chairs, and Keynote Speakers of many conferences. Her research interests include wireless communication and networks, industrial Internet of Things, wireless resource management, and mobile edge computing.



Yuguang Fang (Fellow, IEEE) received an MS degree from Qufu Normal University, a PhD degree from Case Western Reserve University, and a PhD degree from Boston University. He joined the Department of Electrical and Computer Engineering at University of Florida in 2000 as an assistant professor, then was promoted to associate professor, full professor, and distinguished professor, in 2003, 2005, and 2019, respectively. Since 2022, he has been a Global STEM Scholar and the Chair Professor of Internet of Things with Department of Computer Science, City University of Hong Kong.

He received many awards including US NSF CAREER Award, US ONR Young Investigator Award, 2018 IEEE Vehicular Technology Outstanding Service Award, and IEEE Communications Society awards (AHSN Technical Achievement Award, CISTC Technical Recognition Award, and WTC Recognition Award). He was the Editor-in-Chief of IEEE Transactions on Vehicular Technology and IEEE Wireless Communications. He is a fellow of ACM, IEEE, and AAAS.



Tan F. Wong (Senior Member, IEEE) received the B.Sc. degree in electronic engineering from the Chinese University of Hong Kong in 1991 and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University in 1992 and 1997, respectively. He was a Research Engineer at the Department of Electronics, Macquarie University, Sydney, Australia. He also served as a Postdoctoral Research Associate at the School of Electrical and Computer Engineering, Purdue University. Since August 1998, he has been with the University of Florida, where he is currently a Professor of Electrical and Computer Engineering.