Adaptive Split Learning and Resource Allocation over Energy-Constrained Wireless Edge Networks

Zuguang Li*†, Wen Wu*, Shaohua Wu†‡, and Wei Wang§
*Frontier Research Center, Peng Cheng Laboratory, China
†School of Electronics and Information Engineering, Harbin Institute of Technology (Shenzhen), China
‡Department of Broadband Communication, Peng Cheng Laboratory, China
§College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, China
Email: {lizg01, wuw02}@pcl.ac.cn, hitwush@hit.edu.cn, wei_wang@nuaa.edu.cn

***Abstract*—Split learning (SL) is a promising approach to enable edge devices for training artificial intelligence (AI) models. We consider the heterogeneous devices collaborating with a base station (BS) to train an AI model in a distributed manner, for privacy concerns and communication resource constraints. However, due to device heterogeneity and channel uncertainty, the differentiated computing capabilities among devices and the time-varying connections between the BS and devices affect the resource utilization of the BS and devices, and thus, the training delay. In this paper, we design an adaptive split learning (ASL) scheme that determines split point selection and computing resource allocation. We formulate an optimization problem to minimize the average training latency subject to a long-term energy consumption constraint. The major challenges in solving this problem are the lack of future information and mixed integer programming (MIP). To solve it, we propose an online algorithm, named OPEN. Leveraging the Lyapunov theory, the proposed algorithm transforms the problem into a new problem only with the current information, which is a MIP problem. Then, a two-layer optimization method is proposed to solve the MIP problem. Extensive simulation results demonstrate that the ASL scheme can reduce the average training delay and energy consumption by 53.7% and 22.1%, respectively, as compared to the existing SL schemes.**

I. Introduction

The coming sixth generation (6G) network is expected to evolve from connecting people and things to connecting intelligence [1], [2]. At the same time, as the burgeoning edge devices and wireless sensing technology generate big data, edge computing will become an engine for edge network intelligence [3]. The edge network intelligence deeply integrating wireless communications and artificial intelligence (AI) enables the edge devices for training AI models [4]. Privacy concerns and limited communication resources hinder edge devices from offloading their data to a central server for training AI models [5]. Split learning (SL) is a promising approach to edge network intelligence, in which each participant trains only on its segment using its local data and transmits only the small-scale intermediate results at the split point rather than raw data. Therefore, SL can mitigate the aforementioned privacy and communication concerns to a significant extent [6].

In recent years, SL has triggered a fast-growing research interest. In [7], the per-layer execution time and energy consumption in different deep neural networks (DNNs) were analyzed, and then the optimal point could be selected for the best latency or best mobile energy. In [8], an architecture called splitfed learning was proposed to address the issues in federated learning (FL) and SL, where all clients train their local device-side model in parallel and the fed server conducts the aggregation of the client-side local models. In [9] a multi-split machine learning framework was proposed which reimagines the multi-split problem as a min-cost graph search task and optimally selects split points across multiple computing nodes to minimize the cumulative system latency. Similarly, in [10], a split FL framework was designed to enhance communication efficiency for splitfed learning, where both the selection of split points and bandwidth allocation are jointly optimized to minimize the overall system latency.

However, in practical systems, due to device heterogeneity and variation of channel conditions, static split points are adopted for the devices participating in model training, which is not optimal. In addition, the energy consumption performance should be considered in a system with constrained energy. Thus, in wireless edge networks with heterogeneous devices and constrained energy, it is necessary to determine a proper split point and computing resource allocation decisions, to minimize the training delay while simultaneously satisfying a long-term energy consumption constraint. The challenge of this problem is multiple-fold. *Firstly,* split point selection should dynamically adapt to the heterogeneous computing capabilities and uncertain channels during each training iteration. *Secondly,* guaranteeing a long-term energy consumption constraint requires future network information, e.g., channel conditions of devices and energy overhead of the system. Such information is unknown beforehand.

In this paper, firstly, we design an adaptive split learning (ASL) scheme aiming to address the device heterogeneity and channel uncertainty, and thus to reduce the model training delay. In ASL, the heterogeneous devices collaborate with an edge server to train an AI model. Secondly, we formulate a joint optimization problem of the split point selection and computing resource allocation to minimize the average system latency subject to a long-term energy consumption constraint. Thirdly, considering that future network information is required to solve this problem, we propose an Online sPlit point selection and computing rEsource allocatioN (OPEN) algorithm, which jointly determines the split point selection and computing resource allocation in an online manner only with current information. The OPEN algorithm based on the Lyapunov theory transforms the problem into a new problem only with the current information, and a two-layer optimization method is proposed to solve it. Finally, extensive simulations evaluate the performance of the proposed scheme. The main contributions

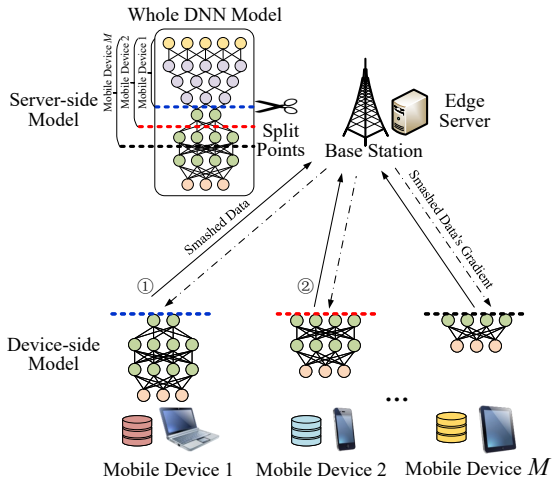


Fig. 1: System Model.

of this work are summarized as follows:

- We propose an ASL scheme to reduce the model training delay in energy-constrained wireless edge networks, which addresses the device heterogeneity and channel uncertainty.
- We design an online algorithm (i.e., OPEN algorithm) to jointly determine the optimal split point and computing resource allocation decisions, and keep the average energy consumption within a threshold.

The rest of this paper is organized as follows. Section II introduces the system model and formulates the problem. We elaborate on the OPEN algorithm in Section III. Section IV presents the simulation results. Finally, Section V concludes this paper.

II. System Model and Problem Formulation

A. Considered Scenario

In this paper, a typical SL scenario is considered, in which multiple mobile devices (MDs) collaborate with a base station (BS) to train an AI model. Each participant only trains a portion of the whole AI model, as depicted in Fig. 1. To complete model training process, the execution *procedure* between the BS and MDs is as follows.

First, the BS transmits the device-side model to a chosen MD, which then trains its own device-side model using its local data until reaching the split point, thereby producing smashed data. Second, this MD transmits the smashed data over the wireless channel to the BS, where it serves as input for the forward propagation on the remaining model layers. Third, at the BS, the edge server computes the loss by comparing the actual labels with the model's predictions and subsequently conducts backward propagation from the last layer up to the split point, yielding the gradient of the smashed data. Fourth, the BS relays the gradient of the smashed data at the split point back to the MD, enabling the MD to perform the backward propagation for its device-side model based on this gradient. Finally, upon completion of the local update, the MD transmits the updated device-side model to the BS, whereupon the device-side and server-side models are merged into an updated global model. The steps above are repeated for all the participating MDs. A training episode refers to all the devices that have been trained one time. Multiple training episodes are performed until a satisfactory model performance is achieved.

The MDs are trained in a line manner, which are indexed by set $\mathcal{M} = \{1, 2, ..., M\}$. The model training episode is indexed by $n \in \mathcal{N} = \{1, 2, ..., N\}$. For the m-th MD participating in the SL, the split point decision at episode n is denoted by $s_{m,n} \in \mathcal{S} = \{1, 2, ..., S\},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}$, where $\mathcal{S}$ represents the available split points in the considered AI model. This means that the front $s_{m,n}$ layers of the AI model are trained on the m-th MD, while the remained layers are trained on the edge server. To guarantee the computing resource constraint, the edge computing resource allocated to the m-th MD at episode n is denoted by $c_{m,n}$, satisfying $0 \leq c_{m,n} \leq 1,\ \forall m \in \mathcal{M},\ n \in \mathcal{N}$.

B. Training Delay

Executing SL requires model computation at the MD and BS and data exchange between them, which incurs computation delay and transmission delay. Both delay components are analyzed as follows.

1) Computation delay: In each local update, the device-side model and the server-side model are trained at the MD and BS, respectively, whose delays are analyzed as follows:

Device-side model computation delay: Let $\eta_D(s_{m,n})$ denote the total computation workload of the m-th MD's device-side model at training episode n, where the amount of computation workload can be represented as the number of floating point operations (FLOPs) [10], [11]. Therefore, the computation delay of the device-side model can be defined as:

$$d_{m,n}^{D,C} = \frac{\eta_D(s_{m,n})}{F_m^D \delta_m^D \sigma_m^D},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}, \tag{1}$$

where F_m^D represents the processing unit frequency of the m-th MD, δ_m^D is the number of FLOPs processed by the m-th MD in a processing unit cycle depending on the architecture of the processor [12], and σ_m^D is the number of cores the m-th MD has.

Server-side model computation delay: Let η denote the total computation workload of the model. Consequently, the server-side model's computation workload is the difference, i.e., $\eta - \eta_D(s_{m,n})$. Similar to that in (1), the computation delay of the server-side model is given by

$$d_{m,n}^{B,C} = \frac{\eta - \eta_D(s_{m,n})}{c_{m,n} F^B \delta^B \sigma^B},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}, \tag{2}$$

where F^B, δ^B, and σ^B represent the frequency of the computing server at the BS, the number of FLOPs processed by a cycle, and the number of cores, respectively. Here, $c_{m,n}F^B$ is the amount of allocated edge computing resources for the m-th MD.

2) Transmission delay: In SL, necessary data, MDs and the BS exchange essential data (e.g., device-side models, smashed data, and gradients) which incur transmission delays. The following provides a detailed analysis.

Device-side model downloading delay: To cooperatively train the AI model with each MD, BS needs to send the latest device-side model to the MDs. Let $R_{m,n}^B = W^B \log_2(1 + (P^B |g_{m,n}^B|^2)/((N_0 + I)W^B))$ represent the transmission rate from the BS to the m-th MD at episode n. Here, W^B, P^B, and $g_{m,n}^B$ represent the downlink bandwidth, the transmission power

at the BS, and the channel coefficient between BS and the m-th participating MD at episode n, respectively. N_o and I denote the power spectrum density of thermal noise and interference caused by other BSs, respectively. Denote the data size of the device-side model with split point $s_{m,n}$ as $\xi(s_{m,n})$, which can be quantified by the count of parameters within that portion of the model. Hence, we have the device-side model downloading delay as follows:

$$d_{m,n}^{B,D} = \frac{\xi(s_{m,n})}{R_{m,n}^{B}},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}. \tag{3}$$

Smashed data transmission delay: In each local update, MDs send the smashed data to the BS for the forward propagation of the server-side model. The transmission rate from the m-th participating MD to the BS is represented by $R_{m,n}^{D} = W_m^D \log_2(1 + (P_m^D \left|g_{m,n}^D\right|^2)/((N_0 + N)W_m^D))$, where W_m^D, $g_{m,n}^D$, and P_m^D represent the bandwidth, the channel coefficient from the m-th MD to the BS at episode n, and MD's transmission power in the uplink transmission, respectively. In addition, the data size of the smashed data given the split point $s_{m,n}$ is denoted by $\beta(s_{m,n})$, which is measured by the number of parameters. As such, the smashed data transmission delay is given by

$$d_{m,n}^{D,S} = \frac{\beta(s_{m,n})}{R_{m,n}^{D}},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}. \tag{4}$$

Gradient transmission delay: The BS needs to send the smashed data's gradient to MDs for the device-side model's backward propagation. Let $\gamma(s_{m,n})$ denote the data size of smashed data's gradient given the split point $s_{m,n}$, which is also measured by the number of parameters. The corresponding delay is calculated as follows:

$$d_{m,n}^{B,G} = \frac{\gamma(s_{m,n})}{R_{m,n}^{B}},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}. \tag{5}$$

Device-side model uploading delay: After a few local updates, the MD needs to upload the latest device-side model to the BS for training the AI model with the next MD. The corresponding delay is given by

$$d_{m,n}^{D,D} = \frac{\xi(s_{m,n})}{R_{m,n}^{D}},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}. \tag{6}$$

3) Overall model training delay: Taking all the computation and communication delay components into account, the model training delay taken for the m-th participating MD at each episode is given by

$$\begin{aligned} D(s_{m,n}, c_{m,n}) = {} & d_{m,n}^{D,C} + d_{m,n}^{B,C} + d_{m,n}^{B,D} + d_{m,n}^{D,B} \\ & + d_{m,n}^{B,G} + d_{m,n}^{D,D},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}. \end{aligned} \tag{7}$$

The performance of a model is intricately linked to the batch size and number of episodes used during training. A model requires approximately 100 episodes to achieve satisfactory performance, especially when there are a large amount of participating MDs. The overall model training times, denoted by MN, can be considered as long terms. The average model training delay across all MDs and the BS until a satisfactory model performance is achieved, defined as $\overline{D}$, can be represented by

$$\overline{D} = \lim_{MN \to \infty} \frac{1}{MN} \sum_{n=1}^{N} \sum_{m=1}^{M} D(s_{m,n}, c_{m,n}). \tag{8}$$

C. Energy Consumption

In SL, all participating MDs need to work with the edge server to complete the model training, which imposes a heavy energy consumption burden on MDs, the BS, and its edge server. The split point selections and computation resource allocation decisions will incur different energy consumption in this system. The energy consumption of the system consists of the components as follows.

1) Transmission-related energy consumption: It is caused by data transmission, including device-side model, smashed data, and gradients. The energy consumption of the m-th MD is a product to the MD's transmission power and communication period, i.e.,

$$E_{m,n}^{D,T} = P_m^D \left(d_{m,n}^{D,S} + d_{m,n}^{D,D}\right). \tag{9}$$

Similarly, the energy consumption of the BS is denoted as

$$E_{m,n}^{B,T} = P^B \left(d_{m,n}^{B,D} + d_{m,n}^{B,G}\right). \tag{10}$$

2) Computation-related energy consumption: It is caused by the model training. The energy consumption for each central processing unit (CPU) cycle can be represented by κF^2, where κ reflects the effective switched capacitance, a parameter that depends on the chip architecture [13], [14]. The energy consumption for device-side and sever-side model training can be defined as

$$E_{m,n}^{D,C} = \kappa \delta_m^D \sigma_m^D \left(F_m^D\right)^2 \eta_D(s_{m,n}), \tag{11}$$

and

$$E_{m,n}^{B,C} = \kappa c_{m,n} \delta^B \sigma^B \left(F^B\right)^2 (\eta - \eta_D(s_{m,n})). \tag{12}$$

3) Overall model energy consumption: All energy consumption of the m-th MD at episode n is $E_{m,n}^{D}$, denoted as $E_{m,n}^{D} = E_{m,n}^{D,T} + E_{m,n}^{D,C}$. For the BS, the overall energy consumption at episode n is $E_{m,n}^{B}$, denoted as $E_{m,n}^{B} = E_{m,n}^{B,T} + E_{m,n}^{B,C}$. Taking all energy consumption of the m-th MD and BS into account, the overall energy consumption at each episode is given by

$$E(s_{m,n}, c_{m,n}) = E_{m,n}^{D} + E_{m,n}^{B},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}. \tag{13}$$

Similar to (8), the average energy consumption of the MDs and BS until achieving a satisfactory model performance, defined as $\overline{D}$, can be represented by

$$\overline{E} = \lim_{MN \to \infty} \frac{1}{MN} \sum_{n=1}^{N} \sum_{m=1}^{M} E(s_{m,n}, c_{m,n}). \tag{14}$$

In the following optimization problem, we keep the average energy consumption within a threshold to save the cherish edge energy.

D. Problem Formulation

We jointly determine split point selection and edge computing resource allocating decisions to minimize the average model training delay while satisfying the average energy consumption requirement, which is formulated as the following problem:

$$\begin{aligned} \mathbf{P}_1: \min_{\{s_{m,n}, c_{m,n}\}} \ & \overline{D} \\ \text{s.t. } & \overline{E} \le E_{th}, \\ & 0 \le c_{m,n} \le 1,\ \forall m \in \mathcal{M},\ n \in \mathcal{N} \\ & s_{m,n} \in \mathcal{S},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}. \end{aligned} \tag{15}$$

Here, the first constraint requires that the system's average energy consumption does not surpass upper limit E_{th}. The second and third constraints guarantee the feasibility of edge computing resource allocation and split point selection.

Problem $\mathbf{P}_1$ is a long-term stochastic optimization problem due to the time-varying channel conditions. In addition, obtaining the optimal policy requires the complete future information, which is impossible. Even if the future information is available, the problem is a mixed integer programming (MIP) since the split point selection is discrete whereas the edge computing resource allocation is continuous, which poses extra difficulty in solving it.

III. OPEN ALGORITHM

A. Problem Transformation

The primary obstacle in tackling problem $\mathbf{P}_1$ lies in handling the long-term constraints. To overcome this hurdle, we employ the Lyapunov optimization method [15], [16]. The essence of this method involves constructing energy deficit queues to represent the fulfillment status of long-term energy consumption constraints, thereby steering the system towards adhering to these constraints. The process of problem transformation is presented as follows.

First, let T denote the total number of model training times between the BS and MDs, i.e., $T = MN$, indexed by $t \in \mathcal{T} = \{1, 2, 3, ..., T\}$. To manage energy consumption, We introduce an energy deficit queue for the system with its dynamic evolves as follows:

$$Q_{m,n}^{t+1} = \left[Q_{m,n}^{t} + E_{m,n}^{t} - E_{th}\right]^{+}, \ \forall m \in \mathcal{M}, \ n \in \mathcal{N}, \quad (16)$$

where $Q_{m,n}^{t}$ indicates the queue backlog in the time slot t representing the accumulated part of current energy consumption that exceeds the upper limit, and $E_{m,n}^{t}$ is the corresponding overall energy consumption $E(s_{m,n}, c_{m,n})$ in the time slot t. The initial state is set to $Q_{m,n}^{0} = 0$ and the queue is updated according to (16).

Theorem 1. *Equation* (16) *is essentially equivalent to the long-term throughput constraint in* (15)*, if the stability condition* $\lim_{T\to\infty} Q_{m,n}^{T}/T = 0, \ \forall m \in \mathcal{M}, \ n \in \mathcal{N}$ *can be satisfied.*

Proof. Please refer to the Proof of Theorem 1 in an online appendix [1]. ■

Second, we utilize a Lyapunov function, defined as $L(Q_{m,n}^{t}) = (Q_{m,n}^{t})^2/2$, to capture the fulfillment status of the long-term energy consumption constraint [17]. The Lyapunov function serves as a quantitative indicator of congestion across all queues. A smaller value of $L(Q_{m,n}^{t})$ signifies a lesser queue backlog, thus indicating enhanced stability of the virtual queue. $L(Q_{m,n}^{t})$ is large when queue backlogs are elevated, resulting in more flows experiencing throughput below the relevant required values. To ensure the energy deficit queue remains stable, i.e., to continuously enforce the energy consumption constraints by driving down the Lyapunov function, we introduce *one-shot Lyapunov drift*, defined as

$$\triangle(Q_{m,n}^{t}) = \mathbb{E}[L(Q_{m,n}^{t+1}) - L(Q_{m,n}^{t})|Q_{m,n}^{t}]. \quad (17)$$

[1]https://github.com/lizuguang/Appendix.

Then, we have

$$\begin{aligned}\triangle(Q_{m,n}^{t}) &= \frac{1}{2}\mathbb{E}\left[(Q_{m,n}^{t+1})^2 - (Q_{m,n}^{t})^2|Q_{m,n}^{t}\right] \\ &\le \frac{1}{2}\mathbb{E}\left[\left(Q_{m,n}^{t} + E_{m,n}^{t} - E_{th}\right)^2 - (Q_{m,n}^{t})^2|Q_{m,n}^{t}\right] \\ &= B_1 + Q_{m,n}^{t}\mathbb{E}\left[\left(E_{m,n}^{t} - E_{th}\right)|Q_{m,n}^{t}\right] \\ &\le B_2 + Q_{m,n}^{t}\mathbb{E}\left[\left(E_{m,n}^{t} - E_{th}\right)|Q_{m,n}^{t}\right],\end{aligned} \quad (18)$$

where $B_1 = \left(E_{m,n}^{t} - E_{th}\right)^2/2$, and $B_2 = \left(E_{m,n}^{\max} - E_{th}\right)^2/2$ is a constant. $E_{m,n}^{\max}$ is the maximum energy that can be produced by the m-th MD and BS at episode n. The first inequality in (18), is due to $(Q_{m,n}^{t})^2 \le \left(Q_{m,n}^{t} + E_{m,n}^{t} - E_{th}\right)^2$, and the second inequality is because $E_{m,n}^{t} \le E_{m,n}^{\max}$.

Third, leveraging the Lyapunov optimization theory, the original problem $\mathbf{P}_1$ can be reformulated into minimizing a *drift-plus-cost* at each time slot, as expressed below.

$$\begin{aligned}&\triangle\left(Q_{m,n}^{t}\right) + V\mathbb{E}\left[D_{m,n}^{t}|Q_{m,n}^{t}\right] \\ &\le B_2 + Q_{m,n}^{t}\mathbb{E}\left[\left(E_{m,n}^{t} - E_{th}\right)|Q_{m,n}^{t}\right] + V\mathbb{E}\left[D_{m,n}^{t}|Q_{m,n}^{t}\right],\end{aligned} \quad (19)$$

where $D_{m,n}^{t}$ represents the overall energy consumption $D(s_{m,n}, c_{m,n})$ at the time slot t, and V is a positive tunable parameter that adjusts the balance between the training delay and energy consumption. The inequality in (19) is due to the upper limit in (18).

Because both B_2 and E_{th} are constants, the original problem $\mathbf{P}_1$ can be reformulated into a new optimization problem $\mathbf{P}_2$ as follows:

$$\begin{aligned}\mathbf{P}_2: \min_{\substack{\{s_{m,n}^{t}, c_{m,n}^{t}\}\forall m\in\mathcal{M},\\ n\in\mathcal{N}}} \ & f \\ \text{s.t. } & 0 \le c_{m,n}^{t} \le 1, \ \forall m \in \mathcal{M}, \ n \in \mathcal{N}, \\ & s_{m,n}^{t} \in \mathcal{S}, \ \forall m \in \mathcal{M}, \ n \in \mathcal{N},\end{aligned} \quad (20)$$

where $f = VD_{m,n}^{t} + Q_{m,n}^{t}E_{m,n}^{t}$. It is worth noting that solving problem $\mathbf{P}_2$ only requires the current information as input.

Problem $\mathbf{P}_2$ is a non-convex optimization problem, in which two decision variables are mixed integers and the objective function is non-convex. To solve the MIP problem, we propose a two-layer optimization method to jointly determine the optimal split point and computing resource allocation decisions, which is detailed as follows.

B. Upper Layer Analysis

When the split point selecting decision is fixed, the edge computing resource allocation problem can be transformed into an upper-layer problem as follows:

$$\begin{aligned}\mathbf{P}_3: \min_{c_{m,n}^{t}, \ \forall m\in\mathcal{M}, \ n\in\mathcal{N}} \ & f \\ \text{s.t. } & 0 \le c_{m,n}^{t} \le 1, \ \forall m \in \mathcal{M}, \ n \in \mathcal{N}.\end{aligned} \quad (21)$$

Theorem 2. *For a given split point selecting decision* $\{s_{m,n}^{t}\}$*, the optimal solution for problem* $\mathbf{P}_3$ *is given by*

$$(c_{m,n}^{t})^{\star} = \begin{cases} 1, \ \textit{if } \sqrt{\frac{V\omega_1}{\omega_4 Q_{m,n}^{t}}} > 1, \\ \sqrt{\frac{V\omega_1}{\omega_4 Q_{m,n}^{t}}}, \textit{otherwise}, \end{cases} \quad (22)$$

Algorithm 1 OPEN algorithm

Input: $Q_{m,n}^0 \leftarrow 0, F_m^D, \delta_m^D, \sigma_m^D, F^B, \delta^B, \sigma^B, P_m^D, P^B, E_{th}$;
Output: split point selecting decisions $(s_{m,n}^t)^\star$, and computing resource allocating decisions $(c_{m,n}^t)^\star$;

```
Initiate R^{D,t}_{m,n}, R^{B,t}_{m,n}, f_min, Q^{t-1}_{m,n};
Set (c^t_{m,n})* = 1, and (s^t_{m,n})* = 0;
while ||(c^t_{m,n})* - c^{t,last}_{m,n}|| > 0.01 and (s^t_{m,n})* ≠ s^{t,last}_{m,n} do
    c^{t,last}_{m,n} = (c^t_{m,n})*, s^{t,last}_{m,n} = (s^t_{m,n})*;
    Compute (c^t_{m,n})* based on (22);
    for s^t_{m,n} = 1 to S do
        Compute D^t_{m,n} based on (7);
        Compute E^t_{m,n} based on (13);
        Q^t_{m,n} = max{Q^{t-1}_{m,n} + E^t_{m,n} - E_th, 0};
        Compute f(s^t_{m,n}) based on (24);
        if f(s^t_{m,n}) < f_min then
            f_min = f(s^t_{m,n});
            (s^t_{m,n})* = s^t_{m,n};
        end if
    end for
end while
return (s^t_{m,n})*, (c^t_{m,n})*
```

where $\omega_1 = \left(\eta - \eta_D\left(s_{m,i}^t\right)\right)/\left(F^B\delta^B\sigma^B\right)$, *and* $\omega_4 = \kappa\delta^B\sigma^B\left(F^B\right)^2\left(\eta - \eta_D\left(s_{m,i}^t\right)\right)$.

Proof. Please refer to the Proof of Theorem 2 in the online file. ■

C. Lower layer Analysis

When the optimal computing resource allocation is given, the split point selection problem can be transformed into a lower-layer problem as follows:

$$\mathbf{P}_4: \min_{s_{m,n}^t,\ \forall m\in\mathcal{M},\ n\in\mathcal{N}} \ f \quad \text{s.t. } s_{m,n}^t \in \mathcal{S},\ \forall m \in \mathcal{M},\ n \in \mathcal{N}. \quad (23)$$

According to (7) and (13), the objective function in (23) can be rewritten as

$$\begin{aligned} f\left(s_{m,n}^t\right) &= VD_{m,n}^t + Q_{m,n}^t E_{m,n}^t \\ &= V\left(d_{m,n}^{D,C} + d_{m,n}^{B,C} + d_{m,n}^{B,D} + d_{m,n}^{D,B} + d_{m,n}^{B,G} + d_{m,n}^{D,D}\right) \\ &\quad + Q_{m,n}^t\left(E^{B,T}(s_{m,n}^t) + E^{B,C}(s_{m,n}^t, c_{m,n}^t)\right). \end{aligned} \quad (24)$$

The objective function in problem $\mathbf{P}_4$ is non-convex because the data sizes (e.g., device-side model, smashed data, and its gradient) are arbitrary functions in terms of the split point. This renders an analytical closed-form expression for the optimal split point infeasible, necessitating numerical methods for its calculation. Given the finite number of potential split points within the AI model, the lower-layer problem can be tackled using an exhaustive searching method.

The OPEN algorithm is shown in Algorithm 1. In the algorithm, the local optimal decision of the edge computing resource allocation is determined based on (22), and the local optimal split point is obtained by searching those split points to minimize $f\left(s_{m,n}^t\right)$. Then, the optimal split point and computing resource allocation decisions can be determined through several iterations.

TABLE I: Simulation parameters.

Parameter	Value	Parameter	Value
A_m^D	4.11	A_m^B	8
δ_m^D	8	δ^B	16
σ^B	32	κ	10^{-26}
ϕ_m^D	1	ϕ_m^B	1
F^B	3 GHz	E_{th}	3,000 J
f_m^D	2,000 MHz	f_m^B	2,000 MHz
P_m^D	0.4 W	P^B	3 W
W_m^D	20 MHz	W^B	40 MHz
N_o	-174 dBm/Hz	N	-164 dBm/Hz

IV. Simulation Results

A. Simulation Setup

The time-varying wireless channel gain of the m-th MD, i.e., $g_{m,n}^D$, is derived from a Rayleigh fading model as $g_{m,n}^D = \rho_{m,n}^D \bar{g}_m^D$ [18]. Here, $\rho_{m,n}^D$ is an independent random channel fading factor following an exponential distribution with unity mean. Meanwhile, $\bar{g}_m^D$ is the average channel gain calculated by using the free-space path loss model as $\bar{g}_m^D = A_m^D\left(\frac{3\cdot 10^8}{4\pi f_m^D d_m}\right)^{\phi_m^D}$. Here, A_m^D, f_m^D, and ϕ_m^D are the antenna gain, the carrier frequency, and the path loss exponent of the m-th MD, respectively, while d_m is the distance between the m-th MD and BS. Analogous to $g_{m,n}^D$, the BS's time-varying wireless channel gain $g_{m,n}^B$ also adheres to the same fading channel model, but it incorporates different values for some parameters.

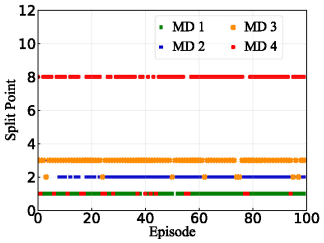
In the simulation, we adopt a 12-layer LeNet model [19] with the widely-adopted MNIST data. The batch size of the training model is set to be 16. Due to the small amount of computation workload and parameters in the activation layers (ReLU), we restrict the selection of split points to only include convolution layers and fully-connected layers. Thus, there are 12 available split points in the LeNet model.

We consider 30 MDs and one BS with an edge server, as illustrated in Fig. 1. Here, $F_m^D \in [0.5, 3]\ GHz$ is set to follow a uniform distribution, $\sigma_m^D \in [1, 8]$ is an integer following a uniform distribution, and $d_m \in [100, 1000]\ m$ follows a uniform distribution. Specially, F_m^D of MD 1, MD 2, MD 3, and MD 4 are set to 0.5 GHz, 1 GHz, 2 GHz, and 4 GHz, their σ_m^D are 1, 4, 8, and 16, respectively, and their d_m are all set to 200 m. The other main simulation parameters are concluded in Table I.

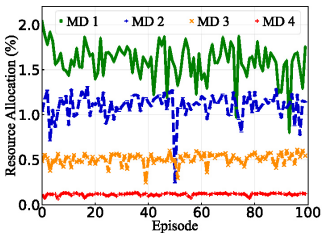
B. Performance Evaluation

We evaluate the optimal split points and computing resource allocation decisions on the MDs with different computing power. As shown in Fig. 2a, we can observe that the optimal split points of MDs dynamically change with episodes, but each of them maintains a split point in general. Furthermore, with the progressive enhancement of an MD's computing capability, its optimal split point shifts closer to the output end. Consequently, it is a lesser need for allocating computing resources, as depicted in Fig. 2b. This is because the MD with strong computing power prefers training the majority of the AI model locally rather than delegating it to the edge server, which benefits reducing the transmission delay.

We compare the ASL scheme with the following benchmark schemes: (1) *SL,* where all devices train the AI model before

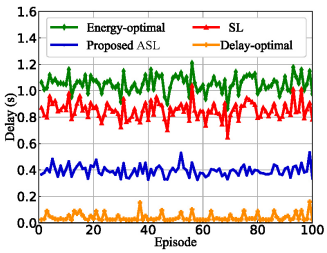


(a) Optimal split point.

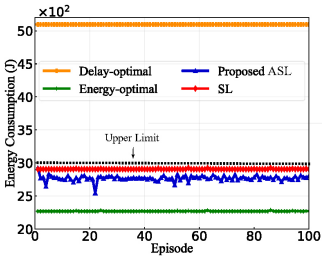


(b) Optimal resource allocation decision.

Fig. 2: Optimal split point and computing resource allocation at each episode.



(a) Delay across all episodes.



(b) Energy consumption across all episodes.

Fig. 3: Delay and energy consumption performance with respect to different schemes.

split point 9 and the edge server releases the total computing resource; (2) *Delay-optimal,* where the goal is to minimize the overall model training delay; (3) *Energy-optimal,* where minimizing the overall energy consumption is the goal. As shown in Fig. 3a, the delays under different schemes fluctuate with the model training, but all remain stable in general. We can observe that the proposed ASL scheme achieves 53.7% and 62.5% reduction in the average training delay, as compared to the SL and energy-optimal schemes, respectively. Although the training delay of delay-optimal scheme is smaller than that of the ASL scheme, its energy consumption is much larger than that of the ASL scheme, as shown in Fig. 3b. In addition, from Fig. 3b, we can see that the energy consumption of the ASL scheme is always maintained below the upper limit, i.e., 3,000 J. Compared to SL scheme, the average energy consumption of the proposed ASL scheme is reduced by 22.1%.

V. Conclusion

In this paper, we have designed a novel ASL scheme for deploying the AI model in energy-constrained wireless edge networks. We have formulated a problem to minimize long-term system latency subject to a long-term energy consumption constraint. To solve this problem, we have proposed an online algorithm to make the optimal split point and computing resource allocation decisions. Extensive simulation results have demonstrated the effectiveness of the ASL scheme in reducing training latency. Due to low training latency and acceptable energy consumption, the ASL scheme can be applied to facilitate AI model training in energy-constrained wireless networks. For future work, we will investigate the split point selection problem in the more complicated AI models.

References

[1] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, 2022.

[2] Y. Chen, P. Zhu, G. He, X. Yan, H. Baligh, and J. Wu, "From connected people, connected things, to connected intelligence," in *Proc. 6G SUMMIT*, 2020, pp. 1–7.

[3] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, 2020.

[4] W. Xu, Z. Yang, D. W. K. Ng, M. Levorato, Y. C. Eldar, and M. Debbah, "Edge learning for B5G networks with distributed signal processing: Semantic communication, edge computing, and wireless sensing," *IEEE J. Sel. Top. Signal Process.*, vol. 17, no. 1, pp. 9–39, 2023.

[5] B. Xie, Y. Sun, S. Zhou, Z. Niu, Y. Xu, J. Chen, and D. Gunduz, "MOBFL: Mobility-aware federated learning for intelligent connected vehicles," in *Proc. IEEE ICC*, 2023, pp. 3951–3957.

[6] W. Wu, M. Li, K. Qu, C. Zhou, X. Shen, W. Zhuang, X. Li, and W. Shi, "Split learning over wireless networks: Parallel design and resource management," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1051–1066, 2023.

[7] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[8] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 8, 2022, pp. 8485–8493.

[9] S. Wang, X. Zhang, H. Uchiyama, and H. Matsuda, "HiveMind: Towards cellular native machine learning model splitting," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 626–640, 2022.

[10] C. Xu, J. Li, Y. Liu, Y. Ling, and M. Wen, "Accelerating split federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, 2023, DOI:10.1109/TWC.2023.3327372.

[11] Z. Xu, F. Yu, C. Liu, and X. Chen, "Reform: Static and dynamic resource-aware DNN reconfiguration framework for mobile device," in *Proc. ACM Des. Autom. Conf.*, 2019, pp. 1–6.

[12] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 7947–7962, 2021.

[13] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, 2020.

[14] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 313–12 325, 2018.

[15] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[16] J. Luo, F. R. Yu, Q. Chen, and L. Tang, "Adaptive video streaming with edge caching and video transcoding over software-defined mobile networks: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1577–1592, 2019.

[17] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2018, pp. 207–215.

[18] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mob. Comput.*, vol. 19, no. 11, pp. 2581–2593, 2020.

[19] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE ICC*, 2019, pp. 1–7.