

# Resource Optimized Split Federated Learning: A Reinforcement Learning and Optimization Approach

Maher Guizani<sup>‡</sup>, Latif U. Khan<sup>†</sup>, Waseem Ullah<sup>†</sup>, Mohammad A. Islam<sup>‡</sup>  
[mahergzani@gmail.com, latif.khan2@gmail.com, mislam@uta.edu]

<sup>‡</sup> University of Texas at Arlington, USA.

<sup>†</sup> Mohamed Bin Zayed University of Artificial Intelligence, United Arab Emirates.

**Abstract**—Federated learning (FL) offers many benefits, such as better privacy preservation and less communication overhead for scenarios with frequent data generation. In FL, local models are trained on end-devices and then migrated to the network edge or cloud for global aggregation. This aggregated model is shared back with end-devices to further improve their local models. This iterative process continues until convergence is achieved. Although FL has many merits, it has many challenges. The prominent one is computing resource constraints. End-devices typically have fewer computing resources and are unable to learn well the local models. Therefore, split FL (SFL) was introduced to address this problem. However, enabling SFL is also challenging due to wireless resource constraints and uncertainties. We formulate a joint end-devices computing resources optimization, task-offloading, and resource allocation problem for SFL at the network edge. Our problem formulation has a mixed-integer non-linear programming problem nature and hard to solve due to the presence of both binary and continuous variables. We propose a double deep Q-network (DDDQN) and optimization-based solution. Finally, we validate the proposed method using extensive simulation results.

**Index Terms**—Federated learning, split federated learning, double deep Q-network.

## I. INTRODUCTION

Recently, federated learning (FL) has been extensively investigated to realize better privacy-aware machine learning for various applications [1]. In wireless systems, FL is important mainly for two reasons. These reasons include better privacy preservation and efficient communication learning for scenarios with frequent data generation (e.g., autonomous cars generate 40,000 Gigabyte of data every day). Therefore, it is very challenging to send frequently generated data to the cloud in such scenarios. FL is a suitable solution for such scenarios. Although FL enables many benefits, it also has challenges. It is challenging to train local FL models at end devices due to computing resource constraints. Additionally, sharing learning updates over a wireless channel is challenging and involves many impairments. These impairments will degrade the quality of the FL learning process. To address the challenge of the computing resource constraint, split FL (SFL) was presented in [2]. In SFL, a partial local model is learned at end-devices and partial at the network edge. Few works considered SFL [3]–[6]. The work in [3] evaluated the performance of SFL

and FL over wireless networks. Another work [4] proposed a hybrid architecture of SFL and FL. They modeled a wireless communication for their framework and performed significant analysis using simulation results. The work in [5] proposed a novel federated split learning architecture and presented results. Otoum *et al.* [6] presented the use of federated learning, split learning, and transfer learning for intelligent transportation system applications. Different from the works in [3]–[6], we propose a novel dueling-based double deep Q-network for efficient task-offloading and resource allocation in SFL over wireless networks. Our contributions are as follows:

- A joint task-offloading and resource allocation problem for SFL is formulated. Moreover, we consider the latency constraint as a QoS metric for SFL.
- Due to the mixed-integer non-programming (MINLP) nature of the formulated problem, we decompose the main problem into sub-problems: (a) devices computing resource optimization, and (b) joint task-offloading and resource allocation problem. For devices computing resource allocation, we use convex optimization, whereas for the joint task-offloading and resource allocation problem, we use a double deep Q-network (DDQN) due to its combinatorial nature.
- Finally, extensive simulations are carried out for validation of the proposal.

## II. SYSTEM MODEL AND PROBLEM FORMULATION

Consider Fig. 1, which depicts a system of Internet of Things (IoT) devices with limited resources. An IoT device set  $\mathcal{S}$  with  $S$  resource devices is available in our system. Each device  $s$  has  $B_s$  data points and a local data  $\mathcal{B}_s, \forall s \in \mathcal{S}$ . Local models are expected to be learned by these devices. Furthermore, a set  $\mathcal{L}$  of  $L$  edge servers will exist. There are limits on computing resources (i.e., CPU-cycles/sec) of end-devices to train local models. Additional computational resources are therefore required. To tackle this issue, [2] introduces the idea of split federated learning (SFL), which is illustrated in Fig. 1. Even though SFL makes a lot of advantages possible, end devices and edge servers must allocate wireless and computational resources effectively. Furthermore, the devices involved in SFL must effectively transfer their tasks to the edge servers. We

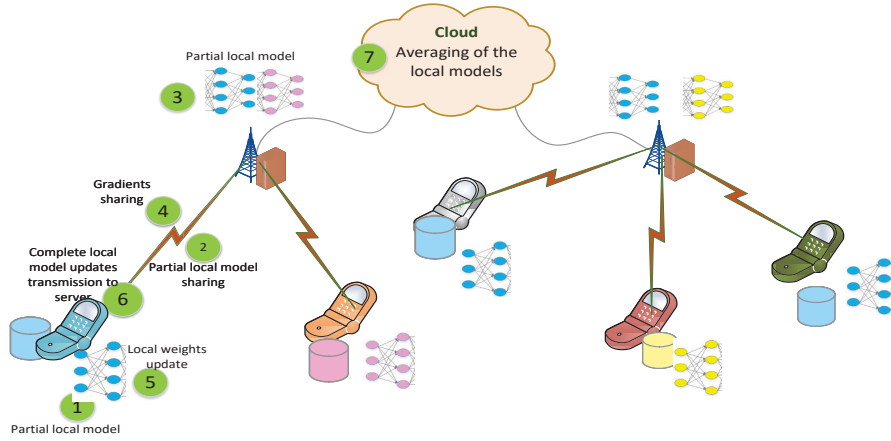


Fig. 1: Split federated learning system model.

continue by discussing the wireless SFL model in the next section.

#### A. Wireless SFL Model

$S'$  of  $S'$  resource-constrained consumer electronics want to take part in a global model's learning process. The set  $S$  of  $S$  will take part in the learning process because of communication resource limitations among  $S'$  of  $S'$ . Because of limitations in computing power, each device learns a partial local model. Let  $\mathcal{T}_s(c_s, b_s, t_s) \forall s \in \mathcal{S}$  be the learning task, where  $(c_s$  and  $b_s$  represent the computational resources required for learning a specific local model for a single data point and entire local data set points, respectively). The local latency in computing a partial model for a specific local model architecture is determined by:

$$\ell_s^{\text{local}}(\varphi) = \frac{c_s b_s}{\varphi_s}, \forall s \in \mathcal{S}, \quad (1)$$

where  $\varphi_s$  indicates the computing resource that device  $s$  has been allocated. Increasing the local computing resource will reduce latency, but at the expense of high local energy consumption (i.e., energy is proportional to the square of a device's running frequency), as (1) makes evident.

$$\Xi_s^{\text{local}}(\varphi) = \alpha_s c_s b_s \varphi_s^2, \forall s \in \mathcal{S}, \quad (2)$$

where  $\alpha_s$  represents device  $s$ 's effective capacitance coefficient. However, the following constraints should be adhered to by the local device's computational resources.

$$\varphi_{\min} \leq \varphi_s \leq \varphi_{\max}, \forall s \in \mathcal{S}, \quad (3)$$

where the minimum and maximum limits are indicated by  $\varphi_{\min}$  and  $\varphi_{\max}$ , respectively. However, the total computing power of all devices should not be greater than  $\varphi_{\text{MAX}}$ , the upper limit.

$$\sum_{s=1}^S \varphi_s \leq \varphi_{\text{MAX}}. \quad (4)$$

In addition to computing the partial local model, it is shared with the edge server, which does the remaining local model computation. There are two methods for calculating the partly remaining local model at the network edge. There are two methods: first, labels can be provided at the network edge; second, data can be shared back to end devices for label-based loss function computation. The output labels are assumed to be accessible at the network edge in this research. At the network edge, additional computing occurs following the transmission of a partial local model to the edge servers. The gradients will be shared back with the devices in addition to computation at the edge. On the other hand, we explore an orthogonal frequency division multiple access (OFDMA) for communication between end devices and edge servers. Given the limited availability of communication resources, it makes sense for learning devices to reuse the resources that are currently in use. We can write the following to determine the transmission latency:

$$\ell_{\text{trans}}(\boldsymbol{\kappa}, \mathbf{v}) = \left( \frac{\kappa_{(s,l)} v_{(s,r)} O_s}{T_r \left( \log_2 \left( 1 + \frac{p_s h_{s,l}}{\sum_{c \in \mathcal{C}} p_c h_{c,b} + N_o^2} \right) \right) \right), \quad (5)$$

where the bandwidth of the resource block  $r$  and the learning data overhead of the device  $s$  are indicated by  $T_r$  and  $O_s$ , respectively. At resource block  $r$ , the inference is  $\sum_{c \in \mathcal{C}} p_c h_{c,b}$ . Noise power is  $N_o^2$ . The variables for task offloading and resource allocation are  $\kappa_{(s,l)}$  and  $v_{(s,r)}$ , respectively. The value of the task offloading variable is  $\kappa_{(s,l)} = 1$ . if  $\kappa_{(s,l)} = 0$  and device  $s$  transfers its tasks to edge server  $l$ , otherwise." Likewise, if the device  $s$  is assigned the resource block  $r$ , the variable  $v_{(s,r)} = 1$ , and if not,  $v_{(s,r)} = 0$ . Each edge server can serve end devices by calculating partial learning tasks for them to a certain extent. Consequently, a limit is required.

$$\sum_{s=1}^S \kappa_{s,l} \leq \kappa_s^{max}, \forall l \in \mathcal{L}, \quad (6)$$

In this instance,  $\kappa_s^{max}$  indicates the upper limit for serving end devices. On the other hand, a single edge server should receive the learning task from a single device.

$$\sum_{l=1}^L \kappa_{s,l} \leq 1, \forall s \in \mathcal{S}, \quad (7)$$

As with task-offloading, resource allocation must be constrained; that is, only one resource block should be assigned to a single device.

$$\sum_{r=1}^R v_{s,r} \leq 1, \forall s \in \mathcal{S}. \quad (8)$$

Wireless resource blocks are required in order to transmit learning updates to the edge servers. The size of the learning data, which is based on the complexity of the end-device model architecture, determines how many resource blocks are required to convey learning updates. For an end-device similar to the work in [7], we employ a single resource block.

$$\sum_{s=1}^S v_{s,r} \leq 1, \forall r \in \mathcal{R}. \quad (9)$$

The transmission energy, however, can be provided by:

$$\Xi_{\text{trans}}(\boldsymbol{\kappa}, \mathbf{v}) = \left( \frac{\kappa_{(s,l)} v_{(s,r)} O_s}{T_r \left( \log_2 \left( 1 + \frac{p_s h_{s,l}}{\sum_{c \in \mathcal{C}} p_c h_{c,b} + N_o^2} \right) \right) \right) p_s, \quad \forall s \in \mathcal{S}, \quad (10)$$

A small cell base station is used to transmit the local model to the distant cloud once it has been computed. The following formula can be used to determine the number of rounds between the end devices and the edge servers:

$$I_g(\varepsilon, \theta) = \frac{\Gamma \log(1/\varepsilon)}{1 - \theta}, \quad (11)$$

where the local dataset dependent constant is  $\Gamma$ . The relative local accuracy is represented by the term  $\theta$ . The interpretation of  $\theta$  differs from that of the local accuracy. In contrast to greater values, lower values of  $\theta$  are preferred, and vice versa. (11) makes it clear that the global rounds are dependent on  $\theta$ ; that is, more global rounds will often occur when  $\theta$  is higher, and vice versa. Additionally, the latency due to transmission of learning updates should not exceed the maximum limit.

$$\left( \frac{\kappa_{(s,l)} v_{(s,r)} O_s}{T_r \left( \log_2 \left( 1 + \frac{p_s h_{s,l}}{\sum_{c \in \mathcal{C}} p_c h_{c,b} + N_o^2} \right) \right) \right) \leq \phi_{max}, \quad (12)$$

$\forall s \in \mathcal{S}, r \in \mathcal{R}.$

Next, we formulate our problem and write it down.

### B. Problem Formulation

The first step in formulating a problem is to determine an objective function. The cost of training the SFL model serves as our goal function. In our situation, we write the objective function taking into account both energy and latency. As stated in [1], we first write energy consumption as follows:

$$\Xi_{\text{total}}(\boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\varphi}, \theta) = (1 + \theta) \Xi_{\text{trans}}(\boldsymbol{\kappa}, \mathbf{v}) + \Xi_s^{\text{local}}(\boldsymbol{\varphi}), \quad (13)$$

$\theta$  is used in (13), to represent the impact of relative local accuracy on energy cost. The cost will be higher for larger values of  $\theta$  and vice versa. The overall latency is now written as:

$$\ell_{\text{total}}(\boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\varphi}, \theta) = \ell_{\text{trans}}(\boldsymbol{\kappa}, \mathbf{v}) + \ell_s^{\text{local}}(\boldsymbol{\varphi}) \quad (14)$$

Like (13), (14) can be rewritten as follows:

$$\ell_{\text{total}}(\boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\varphi}, \theta) = (1 + \theta) \ell_{\text{trans}}(\boldsymbol{\kappa}, \mathbf{v}) + \ell_s^{\text{local}}(\boldsymbol{\varphi}) \quad (15)$$

Next, we write the overall cost as:

$$\mathcal{C}(\boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\varphi}, \chi, \theta) = \beta \ell_{\text{total}} + (1 - \beta) \Xi_{\text{total}}, \quad (16)$$

When calculating the cost of learning the SFL model, the proportion of energy and latency can be scaled using  $\beta$ , which stands for the scaling constant. We now develop a problem with the objective of minimizing the total cost of learning the SFL model.

$$\begin{aligned} \mathbf{P} - \mathbf{M} : & \text{ minimize } \mathcal{C}(\boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\varphi}, \theta) \\ & \text{ subject to:} \\ & (3), (4), (6) - (9), (12). \end{aligned} \quad (17)$$

It is a MINLP problem that has been formulated. A relative local accuracy term in Problem **P-M** is  $\theta$ . For the solution, we leave it fixed, so the problem is reduced to three optimization variables ( $\boldsymbol{\kappa}$ ,  $\mathbf{v}$ , and  $\boldsymbol{\varphi}$ ). Directly resolving the situation is challenging. Furthermore, traditional optimization strategies are not applicable. As outlined in the following section, we shall thus employ a decomposition-based scheme.

### III. PROPOSED OPTIMIZATION AND MULTI-AGENT REINFORCEMENT LEARNING SOLUTION

The MINLP problem that we have defined is difficult to solve. Consequently, we suggest a decomposition-based approach. The foundation of our approach is the separation of the primary problem into two separate problems. These sub-problems include resource allocation, collaborative task-offloading, and device computing resource allocation.

#### A. Devices Computing Resource Allocation Problem

The device computing resource allocation problem is expressed as follows:

$$\mathbf{P} - \mathbf{DR} : \underset{\varphi}{\text{minimize}} \mathcal{C}(\varphi) \quad (18)$$

subject to:

$$\varphi_{min} \leq \varphi_s \leq \varphi_{max}, \forall s \in \mathcal{S}, \quad (18a)$$

$$\sum_{s=1}^S \varphi_s \leq \varphi_{MAX}. \quad (18b)$$

The goal of **P-DR** is to reduce the latency of end devices when computing partial local models. By examining the objective function and constraints, it is possible to demonstrate the convexity of the optimization problem **P-DR**. The objective function only depends on  $\varphi$  for fixed task-offloading variables, resource allocation, and edge servers computing resource allocation. We first take the derivative twice with respect to  $\varphi$  in order to verify the convexity of the objective function. The objective function produces positive results for feasible values of  $\varphi$ . This allows us to conclude that the objective function is convex. The constraints are also linear inequality constraints. Thus, **P-DR** is a convex optimization problem.

#### B. Joint Task-Offloading and Resource Allocation Problem

We write a joint task-offloading and resource allocation problem as:

$$\mathbf{P} - \mathbf{TR} : \underset{\kappa, v}{\text{minimize}} \mathcal{C}(\kappa, v) \quad (19)$$

subject to:

$$\sum_{s=1}^S \kappa_{s,l} \leq \kappa_s^{max}, \forall l \in \mathcal{L}, \quad (19a)$$

$$\sum_{l=1}^L \kappa_{s,l} \leq 1, \forall s \in \mathcal{S}, \quad (19b)$$

$$\sum_{r=1}^R v_{s,r} \leq 1, \forall s \in \mathcal{S}, \quad (19c)$$

$$\sum_{s=1}^S v_{s,r} \leq 1, \forall r \in \mathcal{R}, \quad (19d)$$

$$\Omega_{s,r} \leq \phi_{max}, \forall s \in \mathcal{S}, r \in \mathcal{R}, \quad (19e)$$

$$\kappa_{s,l} \in \{0, 1\}, v \in \{0, 1\}, \quad (19f)$$

Convex optimization techniques are unable to solve the combinatorial problem **P-TR**. An approach based on relaxation and decomposition can be used to solve this [8]. Approximation errors in the conversion of binary task-offloading and resource allocation variables into continuous and back to binary variables are an inherent limitation of this approach. Using an exhaustive search methodology is another method to get around this restriction [9]. The exhaustive search strategy, however, has a significant computational complexity. We suggest a DDQN-based solution in light of the above-stated challenges.

Our scenario is first formulated as a stochastic game for DDQN. Every user in the suggested system is selfish and wants to select resource blocks and edge servers to optimize their long-term gains. The current state of the environment and actions of other devices determine the reward of a specific device or user at any given time  $t$ . Our optimization problem may be expressed as follows for the stochastic game formulation:  $\langle \mathcal{S}, \mathcal{G}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$  [10].

- $\mathcal{S}$  denotes the set of  $S$  devices.
- $\mathcal{G}$  represents the set having all possible states.
- $\mathcal{A}_s$  is the device  $s$  action. Moreover, a joint action vector is denoted by  $\vec{a} = (a_1, a_2, a_3, \dots, a_S)^T \in \times_s \mathcal{A}_s$
- $\mathcal{P}$  represents the state transition probability.
- $\Delta_s$  represents the reward.

Since every user in our system will select a resource block and edge server, the action space can be given by.

$$a_{ls}^r(t) = \{l_s^l(t), r_s^r(t)\}. \quad (20)$$

where  $l_s^l(t) \in \{0, 1\}$ ,  $l_s^l(t) \in \{l_s^0(t), l_s^1(t), \dots, l_s^{L-1}(t)\}$  and  $r_s^r(t) \in \{0, 1\}$ ,  $r_s^r(t) \in \{r_s^0(t), r_s^1(t), \dots, r_s^{R-1}(t)\}$ . The product of the number of resource blocks  $LR$  and the number of edge server-based BSs in our system can be used to determine the total number of states. The action of devices (not including device  $s$ ) at a given time  $t$  can be found using the formula  $\mathcal{A}_{-s}(t) = \{a_{11}^r(t), \dots, a_{l_{s-1}}^r(t), a_{l_{s+1}}^r(t), \dots, a_{l_s}^r(t)\}$ . Note that the present action of any device  $s$ ,  $\mathcal{A}_s$ , and the actions of other devices  $\mathcal{A}_{-s}$ , determine the immediate reward  $\Delta_s(t) = \Delta(g, \mathcal{A}_s, \mathcal{A}_{-s})$ . However, the idea of Nash Equilibrium (NE) can be used to identify mutually optimal and stable tactics in stochastic games. If the following is true, the stochastic game is said to accomplish the NE.

$$\Delta(g, a_s^*, a_{-s}^*) \geq \Delta(g, a_s, a_{-s}^*), \quad \forall a_i \in \mathcal{A} \quad (21)$$

We model the stochastic game using a finite-state Markov decision process (MDP). The devices' reward is dependent upon actions and the current state. As a result, the Markov property is followed by the process.  $\langle \mathcal{S}, \mathcal{G}, \mathcal{A}, \mathcal{P}, \Delta \rangle$  can be used to define the MDP. In our system,  $\Delta$  is equivalent to the

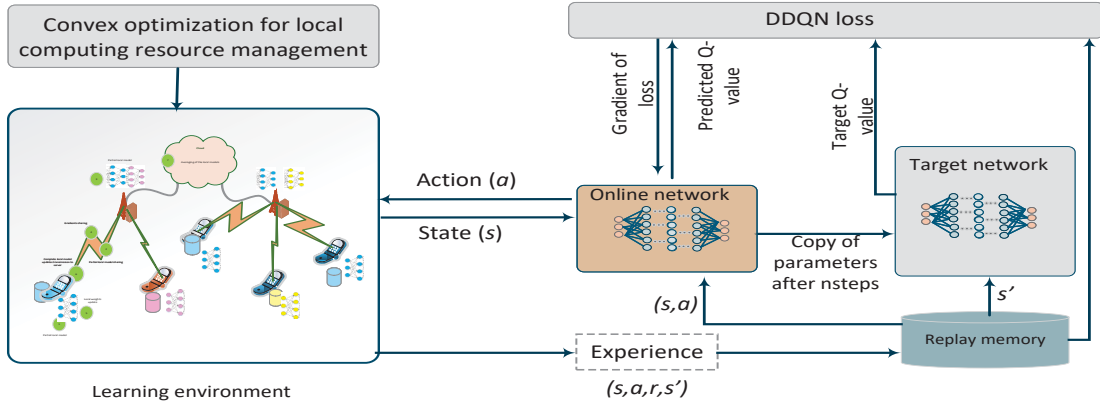


Fig. 2: Proposed solution approach.

$\frac{1}{C}$ . Maximizing the value-state function  $V_s(s, \pi_s, \pi_{-s})$  of every UE in every state yields the best policy  $\pi_s^*$ :

$$V_i(g, \pi_s, \pi_{-s}) = \mathbb{E} \left[ \sum_{t=0}^{T-1} \gamma^t \Delta_s(g_t, \pi_s(t), \pi_{-s}(t)) \mid g_0 = g \right], \quad (22)$$

where the actions of agents (not including  $s$ ) are indicated by  $\pi_{-s}$ . The discount factor is  $\gamma$ . The value function can be expressed as follows if the Markov property is taken into account:

$$V_s(g, \pi_s, \pi_{-s}) = u_s(g_t, \pi_s, \pi_{-s}) + \gamma \sum_{g' \in \mathcal{G}} P_{gg'}(\pi_s, \pi_{-s}) V_s(g', \pi_s, \pi_{-s}). \quad (23)$$

$u_s(g_t, \pi_s, \pi_{-s}) = \mathbb{E} [\Delta_s(g, \pi_s, \pi_{-s})]$ , while the state transition probability is  $P_{gg'}(\pi_s, \pi_{-s})$ . For each  $\pi_s$ , a NE exists if the following conditions are met:

$$V_i(g, \pi_s^*, \pi_{-s}^*) \geq V_s(g, \pi_s, \pi_{-s}^*), \quad \forall g \in \mathcal{G}. \quad (24)$$

We employ Q-learning in order to solve this MDP. The following formula is used to determine the optimal Q-value function  $Q_s^*(g, a_s)$ :

$$Q_s^*(g, a_s) = u_s(g_t, a_s, \pi_{-s}^*) + \gamma \sum_{g' \in \mathcal{G}} P_{gg'}(a_s, \pi_{-s}^*) V_i(g', \pi_s^*, \pi_{-s}^*), \quad (25)$$

where  $V_s(g', \pi_s^*, \pi_{-s}^*) = \max_{a_s \in \mathcal{A}_s} Q_s^*(g, a_s)$ . Now, we write (25) as follows:

$$V_s(g, \pi_s, \pi_{-s}) = u_s(g_t, \pi_s, \pi_{-s}) + \gamma \sum_{g' \in \mathcal{S}} P_{gg'}(a_s, \pi_{-s}^*) \max_{a_s \in \mathcal{A}_s} Q_s^*(g, a_s). \quad (26)$$

Information about the transition probability is very difficult to find. Consequently,  $Q_s^*(g, a_s)$  is updated for Q-learning by:

$$Q_s(g, a_s) = Q_s(g, a_s) + \delta \{u_s(g, a_s, \pi_{-s}) + \gamma \max_{a'_s \in \mathcal{A}_s} Q_s(g', a'_s) - Q_s(g, a_s)\}, \quad (27)$$

where  $\delta$  denotes the learning rate, which helps in determining the  $Q$  value. The value of  $\delta$  should be selected appropriately in order for fast convergence. Additionally, for the action selection, there is a need for tradeoff in action selection. We use  $\epsilon$  greedy strategy for

selecting an action. The random action is selected with a probability  $\epsilon$  and the action is selected with maximum reward with probability  $1 - \epsilon$ . The aforementioned Q-learning method performs better for small state and action spaces. For large action spaces, Q-learning do not perform well. Therefore, there is a need for a better approach. To address this limitation, we can use deep Q-network (DQN) that uses a neural network to represent the action and state space. DQN uses two networks: (a) an online network and (b) a target network. The use of a target network is to stabilize the overall performance. In Q-network, the following loss function is used to stabilize the performance.

$$L_s(\theta) = \mathbb{E}_{g, a_s, u_s(g, a_s), g'} [(y_s^{DQN} - Q_s(g, a_s; \theta))^2], \quad (28)$$

where  $y_s^{DQN} = u_s(g, a_s) + \gamma \max_{a'_s \in \mathcal{A}_s} Q_s(g' a'_s; \theta^-)$ . Here, the term  $\theta^-$  is the target network weights. To chose the action  $a_s$ , one can use online network  $Q_i(g, a_i; \theta)$ . On the other hand, the weights of the target network are keep fixed for many rounds to provide more stability. Further, the experience replay strategy further improves the stability. The current transition is stored in a replay memory. Instead of using only the current transition, we previous transition are also considered for better results. Although DQN offers some merits, it still has a challenges of over optimistic estimation. To address this issue, we can use a double DQN (DDQN) that is based on using a different expression for  $y_s^{DQN}$ . For DDQN,  $y_s^{DDQN}$  can be given by:

$$y_s^{DDQN} = u_s(g, a_s) + \gamma Q_s \left( s', \max_{a'_s \in \mathcal{A}_s} Q_s(g' a'_s; \theta^-) \right). \quad (29)$$

Fig. 2 shows the steps of DDQN. DDQN uses both online and target network to compute optimal value  $Q_s(g, a_s; \theta)$ . Then, with the discount factor and current reward, the  $y_s^{DDQN}$  is computed. Next, we present simulation results.

#### IV. PERFORMANCE EVALUATION

We consider 5 edge server-based BSs and 20 agents in our simulation scenario. For simulations, a  $1000 \times 1000m^2$  area is taken into consideration. Agents are deployed on each device, which is created at random. In the meanwhile, the BS locations remain unchanged. The learning rates that we employ are  $\delta = 0.0001, 0.001, 0.00001$ . The number of agents and resource blocks are regarded as being equal. Each agent will receive a single resource block. The fully connected network architecture we adopt for the agent is composed of an input, two hidden layers, and an output. Additionally, a ReLU activation function is employed. The  $\epsilon$ -greedy policy is used for action. The linear selection of  $\epsilon$  takes place between 0 and 0.9.

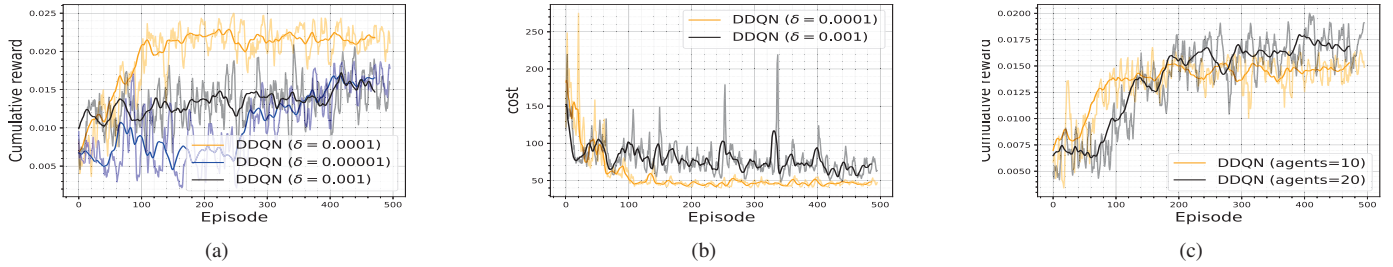


Fig. 3: (a) Cumulative reward vs. episodes for proposed DDQN using various values of learning rates, (b) cost vs. episodes for proposed DDQN using various values of learning rates, and (c) Cumulative reward vs. episodes for proposed DDQN using various number of agents.

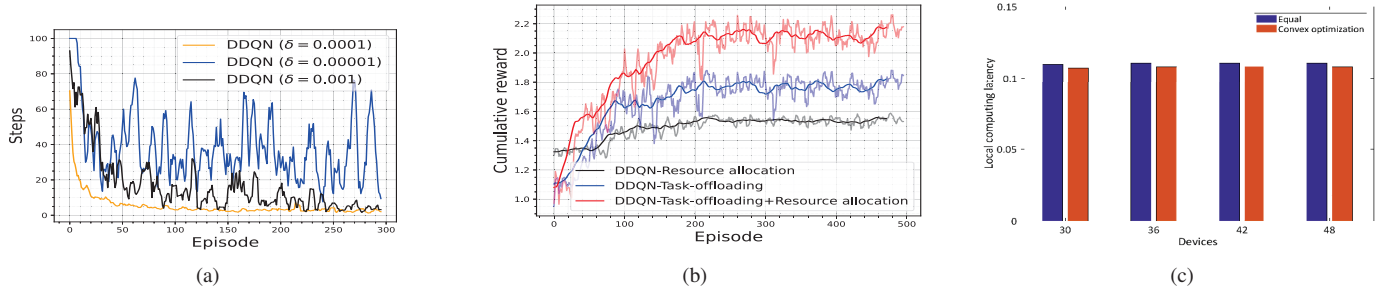


Fig. 4: (a) Steps needed to meet QoS vs. episodes for proposed DDQN using various values of learning rates, (b) steps needed to meet QoS vs. episodes for proposed DDQN using various numbers of agents, and (c) cost for proposed convex optimization scheme vs. random scheme.

Fig. 3a shows the performance of DDQN for various values of learning rate. For the learning rate 0.001, the performance is worst and does not improve with an increase in number of episodes. For the learning rates 0.0001 and 0.00001, the performance improves with an increase in number of episodes. For the learning rate 0.0001, the performance is the best. Another Fig. 3b shows the cost vs. episodes for DDQN using various learning rates. On the other hand, we study the performance of various schemes (i.e., proposed and baselines) in Fig. 3b. It is evident that the proposed scheme outperforms both baselines which shows its superior performance. Fig. 3c shows the reward vs. episodes for various numbers of agents. Again, it is evident that our proposed scheme has a stable performance for a number of agents. Now, we study the number of steps needed within episodes for all devices to reach the QoS (i.e., latency less than a threshold). It is evident from Fig. 4a that performance of DDQN for learning rate 0.0001 is better as it needs less steps to reach the QoS by all devices. Additionally, for different number of agents, the number of steps needed to fulfill QoS for various number of devices remains stable for learning rate 0.0001. Finally, we compare the performance of convex optimization-based local computing resource optimization vs. equal computing resource assignment. The convex optimization scheme outperforms the equal allocation scheme.

## V. CONCLUSION

In this paper, we have considered the optimization of resources to enable SFL. Specifically, resource allocation, task-offloading, and local computing resource optimization are considered. In addition to that, we analyzed the effect of the QoS constraint on the performance of SFL. In order to optimize resource allocation and task-offloading, we employed DDQN. Our extensive analysis showed the better and more stable nature of our proposed solution. We concluded that the framework and solution can be applied to many SFL-based practical applications.

## REFERENCES

- [1] L. U. Khan, M. Guizani, A. Al-Fuqaha, C. S. Hong, D. Niyato, and Z. Han, "A joint communication and learning framework for hierarchical split federated learning," *IEEE Internet of Things Journal*, 2023.
- [2] 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.
- [3] G. Yansong, M. KIM, S. ABUADBBA *et al.*, "End-to-end evaluation of federated learning and split learning for internet of things," in *Proceedings of 2020 International Symposium on Reliable Distributed Systems (SRDS)*, Shanghai, China, vol. 4, 2020.
- [4] X. Liu, Y. Deng, and T. Mahmoodi, "Wireless distributed learning: A new hybrid split and federated learning approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2650–2665, 2022.
- [5] V. Turina, Z. Zhang, F. Esposito, and I. Matta, "Federated or split? a performance and privacy analysis of hybrid split and federated learning architectures," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*. IEEE, 2021, pp. 250–260.
- [6] S. Otoum, N. Guizani, and H. Mouftah, "On the feasibility of split learning, transfer learning and federated learning for preserving security in its systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 7462–7470, 2022.
- [7] 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 transactions on wireless communications*, vol. 20, no. 1, pp. 269–283, 2020.
- [8] R. S. Klein, H. Luss, and U. G. Rothblum, "Relaxation-based algorithms for minimax optimization problems with resource allocation applications," *Mathematical programming*, vol. 64, pp. 337–363, 1994.
- [9] J. Nievergelt, "Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power," in *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 2000, pp. 18–35.
- [10] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, 2019.