Computers, Materials & Continua
DOI:10.32604/cmc.2022.023215
Article

Tech Science Press

Multi-Agent Deep Q-Networks for Efficient Edge Federated Learning Communications in Software-Defined IoT

Prohim Tam[1], Sa Math[1], Ahyoung Lee[2] and Seokhoon Kim[1,3,*]

[1]Department of Software Convergence, Soonchunhyang University, Asan, 31538, Korea
[2]Department of Computer Science, Kennesaw State University, Marietta, GA 30060, USA
[3]Department of Computer Software Engineering, Soonchunhyang University, Asan, 31538, Korea
*Corresponding Author: Seokhoon Kim. Email: seokhoon@sch.ac.kr
Received: 31 August 2021; Accepted: 15 October 2021

Abstract: Federated learning (FL) activates distributed on-device computation techniques to model a better algorithm performance with the interaction of local model updates and global model distributions in aggregation averaging processes. However, in large-scale heterogeneous Internet of Things (IoT) cellular networks, massive multi-dimensional model update iterations and resource-constrained computation are challenging aspects to be tackled significantly. This paper introduces the system model of converging software-defined networking (SDN) and network functions virtualization (NFV) to enable device/resource abstractions and provide NFV-enabled edge FL (eFL) aggregation servers for advancing automation and controllability. Multi-agent deep Q-networks (MADQNs) target to enforce a self-learning softwarization, optimize resource allocation policies, and advocate computation offloading decisions. With gathered network conditions and resource states, the proposed agent aims to explore various actions for estimating expected long-term rewards in a particular state observation. In exploration phase, optimal actions for joint resource allocation and offloading decisions in different possible states are obtained by maximum Q-value selections. Action-based virtual network functions (VNF) forwarding graph (VNFFG) is orchestrated to map VNFs towards eFL aggregation server with sufficient communication and computation resources in NFV infrastructure (NFVI). The proposed scheme indicates deficient allocation actions, modifies the VNF backup instances, and reallocates the virtual resource for exploitation phase. Deep neural network (DNN) is used as a value function approximator, and epsilon-greedy algorithm balances exploration and exploitation. The scheme primarily considers the criticalities of FL model services and congestion states to optimize long-term policy. Simulation results presented the outperformance of the proposed scheme over reference schemes in terms of Quality of Service (QoS) performance metrics, including packet drop ratio, packet drop counts, packet delivery ratio, delay, and throughput.

Keywords: Deep Q-networks; federated learning; network functions virtualization; quality of service; software-defined networking

1 Introduction

The fast-growing deployment of Internet of Things (IoT) in cellular networks has exponentially increased in massive data volumes and heterogeneous service types with the requirement of ultra-reliable low-latency communication (URLLC). By 2025, International Data Corporation (IDC) forecasts that the growth of data generated from 41.6 billion IoT devices will reach 79.4 ZB, which requires big data orchestration and network automation to be intelligent and adequate in future scenarios [1,2]. To control abundant IoT taxonomies and provide sufficient resources, machine learning and deep learning algorithms have been applied to develop smart solutions in edge intelligence for various service purposes by gathering local data for model training and testing [3,4]. Meanwhile, because IoT deployment has grown rapidly in various privacy-sensitive sectors such as Internet of Healthcare Things (IoHT), Internet of Vehicles (IoV), and Internet of People (IoP), the uses of local raw data have to be user-consented and legally authorized before being transmitted to the central cloud [5,6]. With these challenging issues, an intelligent provisioning scheme necessitates considering the security of local data privacy, communication reliability, and adequate computation resources.

Federated learning (FL) secures local data privacy, reduces communication costs, and provides a latency-efficient approach by distributing global model selection and primary hyperparameters, denoted as W_G^0, from central parameter server to local k clients for local model computation [7,8]. In t iterations, W_G^t obtains the optimal model performance by aggregation averaging of multi-dimensional local model updates in a single parameter server. However, over numerous iterations, the client and parameter server communications generate heavy traffic congestions and unreliable processes, particularly in peak hour intervals. Edge FL (eFL) partitions the iterations of round communications in two preeminent steps: (1) The local models w_k^n on n data batch from selected k participants are aggregated in optimal edge server selection, and (2) Global communications are orchestrated to transmit between edge servers and a central parameter server in an appropriate interval [9–11]. This technique reduces cloud-centric communications and improves learning precision. Therefore, a system model for offering edge aggregation servers based on specific service-learning model criticalities is applicable to enhance resource-constrained IoT environments.

Multi-access edge computing (MEC) leverages computation powers and storage capacities of the central cloud to provide a latency-efficient system, adequate Quality of Service (QoS) performance, and additional serving resources in edge networks [12,13]. 5G radio access networks (RAN) support stable connectivity and adaptability between massive users and MEC entities for driving big data communication traffics with the deployment of millimeter-Wave (mmWave), multiple-input and multiple-output (MIMO) antennas, device-to-device (D2D), and radio resource management (RRM) functions. Moreover, to extend a global view of network environments and efficiently control heterogeneous MEC entities, software-defined networking (SDN) has been adopted. An adaptive transmission architecture in IoT networks is advanced by joint SDN and MEC federation to enable an intelligent edge optimization for low-deadline optimal path selection [14]. SDN separates the data plane (DP) and control plane (CP) to enable programmable functions, which adequately control the policies, flow tables, and actions on domain resources management within RAN, core side, network functions virtualization (NFV), and MEC [15,16]. The convergence of MEC, SDN, and NFV enables the networking application programming interfaces (API), sufficient resource pools, flexible orchestration, and programmability for logically enabling resource sharing virtualization in an adaptive approach. To optimally allocate the resources and recommend the offloading decisions within NFV infrastructure (NFVI)-MEC, an intelligent agent or deep reinforcement learning approaches have a capability to apply as enablers for network automation (eNA) in order to interact with particular IoT device statuses, resource utilization, and network congestion states.

Deep Q-network (DQN) has notably been used for addressing resource allocation and computation offloading problems in massive IoT networks [17]. There are three main procedures to construct DQN-based model, including epsilon-greedy strategy, deep neural network (DNN) function approximator, and q-learning algorithm based on Bellman equation for handling Markov decision process (MDP) problem. S, A, R, and γ represent the batch of potential states, actions, rewards, and discount factors for future rewards, respectively [18,19]. In the initial step t, the agent explores by epsilon-greedy method and randomly selects an action a_t for sampling the reward $r_t(s_t, a_t)$ in order to further calculate q-value of that particular s_t. At time $t+1$, the environment feedbacks the next-state observation s_{t+1} based on the transition $p(.|s_t, a_t)$. This exploration strategy will iteratively execute until the optimal q-value and policy are defined. Algorithm design based on reinforcement principles feasibly observes scheduler states and explores rule actions to propose scheduling rules for adaptive resource management and enabling QoS provisioning scheme. Moreover, a model-free multi-agent approach feasibly tackles the heterogeneity of core backbone network for efficient traffic control and channel reassignment in SDN-based IoT networks [20].

1.1 Paper Contributions

In this paper, the proposed system architecture is adopted to deploy multi-controller placement in NFV architecture for observing various state abstractions. Multi-agent DQNs (MADQNs) explores actions on resource placement and computation decisions for offloading w_k^n towards appropriate eFL aggregation server. Centralized controller abstracts IoT device and resource statuses to gather state spaces for the proposed adaptive resource allocation agent (PARAA). Decentralized controllers as a virtualized infrastructure manager (VIM) and VNFs are presented to abstract NFVI states for proposed intelligent computation offloading agent (PICOA). MADQNs obtain the maximum future long-term reward expectation of joint state spaces by using q-value function and DNN approximator. The optimal policy is defined before exploitation phase and obtained by a centralized controller. The proposed scheme extends MADQNs by rendering virtual network functions (VNF) forwarding graph (VNFFG) and upgrading the deficient allocation actions to approach sufficient serving MEC resource pools. The proposed controller updates the forwarding rule reactively for long-term sufficiency. An experimental simulation is conducted to illustrate the performance of proposed scheme. The custom environment and DQN agent were developed by using OpenAI Gym library, TensorFlow, Keras, and the concept of Bellman equation. To evaluate the QoS metrics in SDN/NFV aspects, Mininet and RYU SDN controller are conducted. In NFV management and orchestration (MANO), mini-nfv framework is applied on top of Mininet to develop the descriptors using TOSCA NFV template. Finally, simulation on 5G new radio (NR) networks is conducted to present an end-to-end (E2E) perspective by using ns-3, a discrete-event network simulator.

1.2 Paper Organizations

The rest of the paper is organized as follows. The system models, including architectural framework and preliminaries of proposed MADQNs components, are presented in Section 2. The proposed approach is thoroughly described in Section 3. In Section 4, simulation setup, performance metrics, reference schemes, and result discussions are shown. Section 5 presents the final conclusion.

2 System Models

2.1 Architectural Framework

In the system architecture, SDN CP allows a programmable DQN-based mechanism to observe states of the network environment via OpenFlow (OF) protocol in southbound interface (SBI), which allows the cluster head to contribute significant roles for data of IoT nodes and resource utilization collection [21]. The proposed SDN/NFV-enabled architecture for supporting MADQNs programmability and offering multiple eFL servers within NFVI-MEC environment is shown in Fig. 1. In the proposed system architecture, the centralized SDN controller communicates with NFV-MANO layer for management functions in VNF manager (VNFM) and VIM through orchestration interfaces [22]. Ve-Vnfm interface interacts between SDN controller as VNFs and VNFM for operating the lifecycle network services and resources management. Nf-Vi interface allows the controllability of NFVI resource pools for central SDN controller as a VIM [23]. To activate connectivity services between virtual machine (VM) and VNFs, Vn-Nf logical interface is used in the proposed architecture to adjust the virtual storage and computing resources based on VNFs mapping orchestration. To configure resource allocation based on optimal PARAA policy, decentralized SDN controllers as a VIM and VNFs are proposed in this scheme to formulate the parameterization of action-based VNFFG rendering for service function chaining (SFC) management system. The proposed MANO manages the VNF placement with appropriate element management system (EMS), virtual deployment unit (VDU), and VM capabilities based on allocation policy in particular congestion state spaces. The resource-constrained state observation leads to a prior for agents to adjust the backup instances with model service prioritization. After the resources are adjusted, PICOA computes the policy to advocate eFL server for local model aggregation offloading. Within multi-controllers, the flow entry installation process is configured reactively in the centralized entity. Each cluster head is commanded by OF protocol with a flow rule installation. Since proactive mode has the capability for each OF-enabled switch to set up the flow rules internally, the proposed agent controller will prioritize the reactive rule installation to ensure the proposed central policy configuration. Agent controller checks the packet flow with all the global tables and updates counters for instruction set executions. In our proposed scheme, the flow priority, hard timeout, and idle timeout are measured by the remaining MEC resources, time intervals, and criticalities of FL model services. However, if there is no match within global tables, the agent controller executes the add-flow method based on the particular state-action approximation to accordingly append datapath id, match details, actions, priority, and buffer id. With different dimensional features and scale values, SDN database entity is expected to handle the storage and preprocessing phases. For the proposed agent model, the data requiring from SDN database is uplink/downlink resource adjustment statuses, resource of eFL MEC nodes, and default core resource utilization system. With these features, the agent feasibly acquires the state observation spaces for sampling and exploring the potential actions.

2.2 Proposed DQN Components

In this context, main components of MADQNs consist of state, action, reward, and transition probability. For the hyperparameters, the values are optimized by standard parameterization for controlling the behavior of the learning model such as learning rate α, discount factor γ, epsilon ε, and mini-batch sizes q_m. In software-defined IoT networks, the local, distributed, and centralized resources for communication and computation are complex to measure thoroughly. Moreover, the observation and discrete values will be challenging to capture. Therefore, each element was assigned in percentile scales.

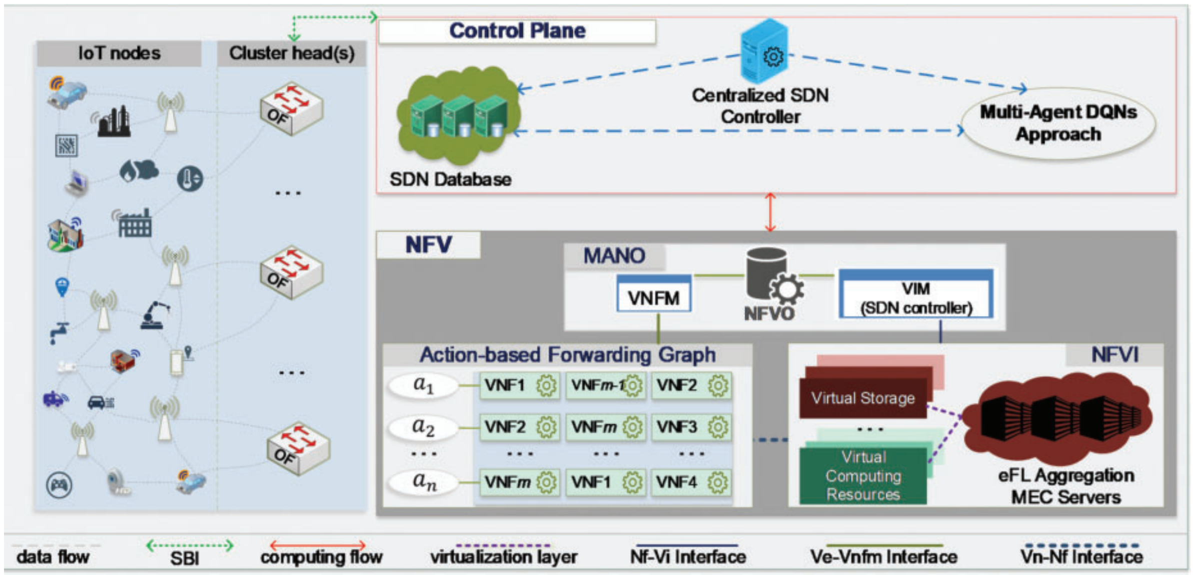

Figure 1: The system architecture for MADQNs approach and virtualization of eFL servers

State: in MADQNs environment, the state spaces are comprised of two main observations for PARAA and PICOA. For PARAA, the state consists of control statuses and a global functional view, including the extant maximum and minimum resources denoted as res_{max} and res_{min}, respectively. For PICOA, the state spaces are abstracted by decentralized controllers, including the maximum eFL node i capacities, cost of VNF m placement at eFL node i, and computation cost of local model w_k^n at eFL node i, denoted as res_{mec}^i, $cvnf_m^i$, and cp_w^i, respectively. The joint state observation contains two significant spaces such as uplink/downlink resource statuses and resource increment/decrement discrete adjustment in default resource utilization system, which is denoted as res_c and res_{pace}, respectively. Eqs. (1) and (2) presents the expression of state spaces. The increment/decrement level was indicated according to the positive and negative weights, denoted as ω^+ and ω^-, of a particular peak/off-peak network congestion. Based on the experience replay, optimal resource targets are defined.

$$S_{PARAA} = \{res_{max},\ res_{min},\ res_c, res_{pace}\} \tag{1}$$

$$S_{PICOA} = \{res_{mec}^i, cvnf_m^i, cp_w^i, res_c, res_{pace}\} \tag{2}$$

Action: in this environment, the batch of potential actions refers to the resource updates and SFC, which are collectively mapped by VNFFG parameterization towards virtual MEC resource pools in the NFVI entity. Numerically, the action spaces a specify the discretization operation scale of increment, decrement, and static, denoted as $A_{PARAA} \in \{0, 1, 2\}$. The percentage of applied action values is set within a restrained allocation step to reach an optimal balance of downlink and uplink capacities. The scheme forecasts computing and storage resources for processing VNF m and allocation index in serving eFL node i, denoted as (cp_m, sr_m) and ai_m^i, respectively. In the proposed system architecture, i eFL aggregation server decisions are provided in PICOA as $A_{PICOA} \in \{1, 2, \ldots, i\}$ by evaluating the task execution efficiency.

Reward: the intermediate reward in a particular time t, denoted as $r_t(s_t, a_t)$, is maximized when the agent reaches the optimal resource allocation res_{oe}, which is adaptable based on three essential conditions, including transmission intervals, experienced q-value, and the remaining resource percentile. Moreover, in IoT peak hour congestion, the resource increment requires the extra serving available resources in virtual computational blocks, denoted as res_{xt}. The output of computational capabilities from the selected action towards virtual MEC resource pools in each VNF is the main component for model aggregation completion. The reward considers the number of VNF requests and the computational costs of each VNF in that particular selected VNFFG rendering. The output of reward determines the efficiency of resource allocation and eFL server selection from actions of PARAA and PICOA agents in a defined state.

Transition Probability: different policy determines distinct transition step for sampling the next state observation. In the early stage, the randomness of transition policy allows the agents to explore the actions without specified probabilities. However, once the exploration strategy reaches an optimal goal of resource allocation rewards, epsilon-greedy policy executes the transition, denotes as $p(.|s_t, a_t)$, by performing the exploitation strategy follows the given action to its state pair. Thereafter, when the agent receives the next state spaces s_{t+1} from environment feedback, the agent will check the variation and diversity in the experience pools to enforce a particular action for that state space.

3 Multi-Agent Deep Q-Networks for Efficient Edge Federated Learning Communications

To describe the MADQNs softwarization framework with the proposed controllers towards virtual resource allocation and eFL aggregation server selection, this section delivers two primary aspects of the proposed scheme, including the algorithm flows for multi-agent in NFVI-MEC and self-organizing agent controllers for collaborative updates in NFV-enabled eFL.

3.1 Algorithm Flow for MADQNs in Proposed Environment

To optimize the policy of the model, q-table and DNN are computing in parallel behavior to support the trade-off between time-critical and precision. However, DNN acts as a central control which structures as a prime approximator. Each potential state-action pair has a q-value that accumulates in both q-table and approximated DNN output layer after the exploration strategy. With a feedforward network, numerous weight initializations, neurons, and multiple layers of perceptron, the q-value decision-making is more accurate, yet execution time is simultaneously high. To optimize a policy for a long-term self-learning environment, the randomness in exploration processes of the networking environment has to be handled. The hyperparameters are required to be well-assigned and related to the fine-grained scenario. The optimal policy for exploitation strategy as the end goal is denoted as π^*, which further expresses in Eqs. (3)–(6). Each policy interprets the agent and observation differently based on the value function and q-value function with distinct transition probability p. The required parameter consists of the beginning state resource conditions s_0, current state s_t, next state s_{t+1} observation of $[res^i_{mec}, cvnf^i_m, cp^i_w, res_c, res_{pace}]$, criticality intervals, and sample action a_t. Subsequently, the working process of MADQNs elements in NFVI-MEC environment is described in three major functional phases, including value and q-value functions, function approximator, and experience replay.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_\pi \left[\sum_{t=0}^{endT} \gamma^t r_t \mid \pi \right] \tag{3}$$

$$s_0 \sim p(s_0) \tag{4}$$

$$a_t \sim \pi(.|s_t[res^i_{mec}, cvnf^i_m, cp^i_w, res_c,\ res_{pace}]) \tag{5}$$

$$s_{t+1}[(res^i_{mec})_{t+1}, (cvnf^i_m)_{t+1}, (cp^i_w)_{t+1},\ (res_c)_{t+1}, (res_{pace})_{t+1}] \sim p(.|s_t, a_t) \tag{6}$$

The value function is computed for policy transformation and low-dimensional perspective to get the value of state s and create sample paths. It is significant to identify the resource condition at a particular time. To differentiate between each random exploration policy, the cumulative reward is a key value to maximize the expectation. Value function captures a vector of reward to follow a particular policy π for evaluating the performance of an agent by defining the expected future rewards. The value function denoted as $V^{\pi}(s[res^i_{mec}, cvnf^i_m, cp^i_w, res_c,\ res_{pace}])$ of an input state observation s into a policy π is used for returning the expected outcome following the MDP. In our proposed environment, the value function is executed in exploitation strategy following the policy. The q-value function can be expressed in order to adapt to a specific computation state, which follows the Bellman Equation to label the q-value for state-action pairs. Towards the optimal q-value $Q^*(s_t, a_t)$, the formulation with proposed state observations and action spaces in our setup environment features is presented (see Eq. (7)). The expected main requirements are the rewards of that particular state-action pair and the value of the next state that the environment ends up in, which is expressed as $s'[(res^i_{mec})', (cvnf^i_m)', (cp^i_w)',\ res'_c,\ \ res'_{pace}]$. The expectation $\mathbb{E}_{s'\sim\varepsilon}$ expresses the randomness of state s' observation. To solve for the optimal policy, the iterative update has to be executed. With known optimal policy, the best action will be chosen at state s' to maximize the q-value. However, this process is only supported in the short-period networking simulation process, but not supported in long-term sustainability and iterative execution; therefore, the function approximator comes to take place.

$$Q^*(s,a) = \mathbb{E}_{s'\sim\varepsilon}\left[r + \gamma \max_{a'} Q^*(s'[(res^i_{mec})', (cvnf^i_m)', (cp^i_w)',\ res'_c,\ res'_{pace}], a')\right] \tag{7}$$

DNN estimates the functions $Q(s, a; \theta, b)$ based on biases b and weights θ on each neuron connector between each perceptron which is equivalent to peak hour and off-peak hour intervals in networking priority. The input processing of each possible state observation from time 0 to initial time t, including the resource conditions of the network environment, towards an optimal action-value selection is based on the congestion status. And weights and bias in one sample perceptron are adjusted. The rectified linear unit (ReLU) activation function is used to transform the sum of each connector weight and bias intervals from input until the output layer. Algorithm 1 presents the MADQNs flow towards the optimal resource allocation policy and eFL selection in the proposed framework. Agent execution starts with hyperparameters and parameters initialization. The total reward container per episode, particular reward in each episode, the number of episodes, discrete state spaces, q-table, the starting epsilon value, the final epsilon value, and particular epsilon-decaying value are denoted as r_e, e_r, num_e, $s_{discrete}$, q_{table}, ε^s, ε^e, and ε^-, respectively. The scheme targets the gathered state observations for applying agent learning. The joint allocation and computation costs are considered for calculating expected rewards, including the number of VNFs to attain eFL node i decision, denoted as $nvnf_i$. However, to detail the architectural stack of applied DNN, a TensorFlow-based implementation is designed to reach the optimal model with accurate parameter estimation.

The experience replay, denoted as $e_t = (s_t, a_t, r_t, s_{t+1})$, feeds the online and target networks for choosing the actions and approximating its q-value. Since there are numerous possible continuous networking states, the discrete state observations are utilized to input in the first layer as a mini-batch of resource conditions for approximating an optimal increment/decrement between uplink and downlink communications. However, if ω^+ is high, the resource utilization system is also increasingly enlarged to solve the bottleneck issues. The double dense layers with ReLU are applied to analyze the state differentiation and suitable actions to maximize the upcoming reward. For the output layer with linear, the action q-value is triggered based on the dense layer conditions. If the gradient update evaluates an unsatisfied precision, the model will be reprocessed. Until the model is accepted, the compiling process is executed with Adam optimizer and mean squared error (MSE) metric.

Algorithm 1: Pseudocode for the proposed MADQNs towards optimal action selection

```
    Require: $s\ [res^i_{mec}, cvnf^i_j, cp^i_w, res_c, res_{pace}], res_{min}, res_{max}, A, res_{oe}, res_{xt}, \omega^-, \omega^+$
    Ensure: optimal actions on allocation policies and eFL server selection from each episode for orchestrating NFVI-MEC resource pools
1:    Initialize $r_e, \gamma, \alpha, num_e, q_m, s_{discrete}, q_{table}, \varepsilon, \varepsilon^s, \varepsilon^e, \varepsilon^-$
2:    for each episode in range ($num_e$) do
3:        Initialize each episode reward $e_r$
4:        Transform the state $s$ to $s_{discrete}$
5:        while true do
6:            if random() > $\varepsilon$ then
7:                Agent selects action $a$ by $a = argmax(q_{table}[s_{discrete}])$/DNN
8:            else
9:                Agent selects random action $a$
10:           end if
11:           Calculate reward $r$ based on computation and placement costs
12.           Perform selected action $a$, then enter next-state $s'$ by $p(.|s, a)$
13:           Add the defined reward $r$ to the initialized episode reward $e_r$
14:           Transform the next-state $s'$ to the clustering chunk $s'_{discrete}$
15:           if the next-state resource $res_c$ has a stable and optimal allocation statuses do
16:               Input the maximum q-value for state-action pair
17:           else
18:               Initialize future maximum q-value by $max(q_{table}[s'_{discrete}])$
19:               Initialize current-q-value $Q_c$ for state-action pair
20:               $Q_{NEW} \leftarrow (1 - \alpha)Q_c + \alpha \cdot (Q^*(s, a) - Q_c)$ (see Eq. (7))
21:               Input new q-value $Q_{NEW}$ for the state-action pair
22:           end if
23:           Update $s_{discrete}$ to $s'_{discrete}$
24:       end while
25:       if $\varepsilon^e \geq episode \geq \varepsilon^s$ then
26:           $\varepsilon - = \varepsilon^-$
27:       end if
28:       $r_e.append(e_r)$
29:   end for
```

By applying the proposed MADQNs model, the average reward aggregation for the resource allocation environment is obtained. The average reward output of optimal resource allocation is steady for most of the episodes but remains some downward marks towards limited resource utilization, which leads to unstable management. The steady and unsteady state-action pairs are detected and needed to be significantly enhanced for avoiding high packet drop scenarios in heterogeneous local model update communications.

3.2 Self-Organizing Agent Controllers for Optimal Edge Aggregation Decisions

The implicit algorithm flow is proposed to handle the instability of MADQNs model in NFVI-MEC environment by leveraging the capabilities of the agent controllers and orchestrator. The proposed method installs flow rules for each IoT cluster head with the adjustment of uplink/downlink resource utilization priority. The orchestrator configures VNFFG descriptors following the resource allocation policy from Algorithm 1 towards eFL aggregation with optimal MEC resource pools. The proposed agent controller requires to orchestrate the flow entry tables of multiple IoT cluster heads by applying the convergence of resource allocation policy and OF controller flow stats. Each state-action (s, a) pair from MADQNs-based model is transformed into the flow configuration pair (s_f, a_f) by updating the uplink/downlink resource statuses and peak hour/off-peak hour intervals into instruction sets and priority of the entries, respectively. Based on the priority and instruction sets of each traffic flow, orchestrator gains the prior information to handle the virtual resource pool adjustment in NFVI. Fig. 2 presents the state transition of MADQNs and controller management within SDN/NFV system. When the client updates the local model, the pipeline processing is performed. Integrated PARAA and PICOA algorithms optimize the resource allocation policies and eFL aggregation MEC selections as described in Algorithm 1. Within the inspected deficient actions in training phase, the proposed scheme adjusts the policy and appends sufficient virtual resource pools for optimizing the serving capacity of the selected eFL node, as described in Algorithm 2.

4 Performance Evaluation

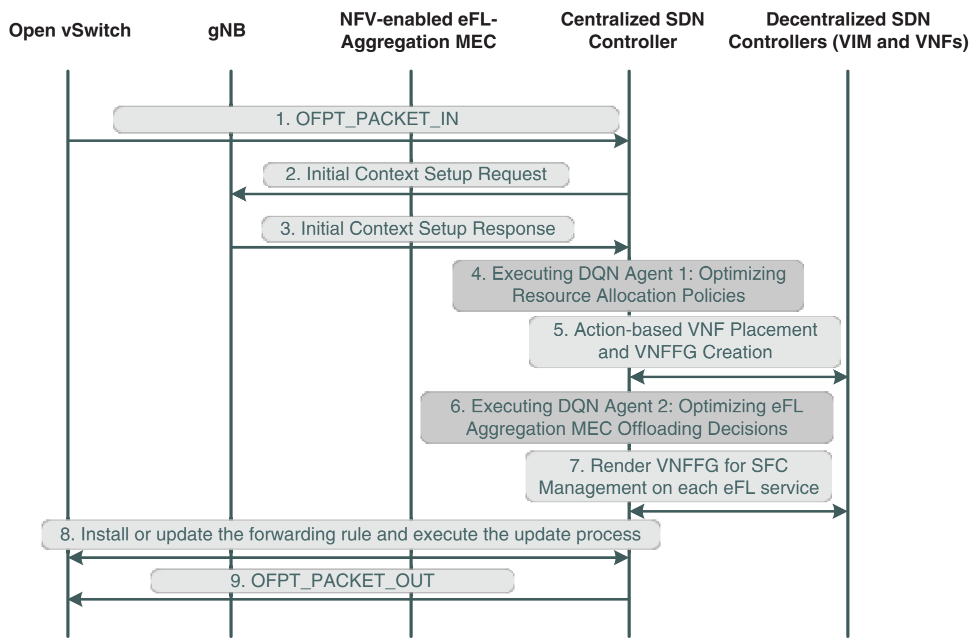
Open vSwitch
gNB
NFV-enabled eFL-Aggregation MEC
Centralized SDN Controller
Decentralized SDN Controllers (VIM and VNFs)
1. OFPT_PACKET_IN
2. Initial Context Setup Request
3. Initial Context Setup Response
4. Executing DQN Agent 1: Optimizing Resource Allocation Policies
5. Action-based VNF Placement and VNFFG Creation
6. Executing DQN Agent 2: Optimizing eFL Aggregation MEC Offloading Decisions
7. Render VNFFG for SFC Management on each eFL service
8. Install or update the forwarding rule and execute the update process
9. OFPT_PACKET_OUT

Figure 2: The state transition for proposed controllers to install forwarding rule of local model updates

4.1 Simulation Setup

To prove the theoretical approach, this section describes the three main simulation adoption environments, including MADQNs model construction, SDN/NFV control performance, and 5G NR network experiment to capture the E2E QoS performances.

By using OpenAI Gym library [24], the environment setup requires four primary functions. The initialization (init) function declares the available characteristics of state observations (see Eqs. (1) and (2)) in the setup environment. The init explores s_0 within the epsilon-greedy random exploration. The allocation step function updates the new state environment, gives a reward, and completes statutes after the agent controller performs any specific actions. Finally, whether restarting the simulation, changing the network circumstances, or starting a new episode, the reset function is used. The goal of the MADQNs model is to interact with the setup environment and choose the optimal action for a specific networking state in order to optimize eFL offloading server decisions. To train and test the models, we used TensorFlow and Keras [25,26]. Fig. 3 presents the total average rewards per 100 episodes with three α performances, including 0.01, 0.05, and 0.09. The rewards are outputted in negative numbers since the setup assigned the non-optimal reward as -0.5, which was cumulatively summed until the end of episodes. In each episode, the $Q^*(s, a)$ are gathered. In this environment setup, the optimal α is 0.09, which fluctuates around -128.3075.

Algorithm 2: Proposed self-organizing agent controllers for optimizing eFL aggregation selection

```
      Require: PARAA and PICOA decisions based on optimal actions
      Ensure: optimal flow entry installation, resource orchestration, and eFL server selection
1:    for each local model update in $t$ iteration do
2:        Transform the discrete state $s_t$ to match with the flow criteria $(s_t)_f$
3:        for each OFPT_PACKET_IN in range of cluster heads $CH$ do
4:            if no match found in local $CH$ tables do
5:                Apply optimal policies of PARAA and PICOA to bridge traffic through VNFs
6:                Transform the optimal discrete action $a_t$ to adapt with the flow stats $(a_t)_f$
7:                VNFFG creation for rendering to SFC, then, install the flow entry for execution
8:            else
9:                Execute the instruction sets of the found flow entry
10:           end if
11:           for each selected eFL server in range($i$) do
12:               Perform edge aggregation for optimal $w_i^t$
13:           end for [edge aggregation]
14:       end for [OFPT_PACKET_OUT]
15:   [Global Server]
16:       Compute averaging aggregation on $[w_1^t,\ w_2^t, \ldots w_i^t]$ for global model $W_G^{t+1}$
17:   end for [t iteration]
```

To capture the particular QoS performance metrics of the proposed controllers and NFV modules, mini-nfv on top of Mininet is used to create the data plane topology, VNF descriptors, and VNFFG descriptors. Mini-nfv supports the external SDN controller platform for experimentation. The forwarding rule installation is configured by FlowManager and RYU-based platform [27–31]. The descriptors set the VDU and VM capabilities based on selected actions from the optimal policy table. Each flow entry is configured following the forwarding graph. Fig. 4 presents the interaction of the convergence; however, the virtual links for communication perspective are still restricted for explicit fine-grained performance.

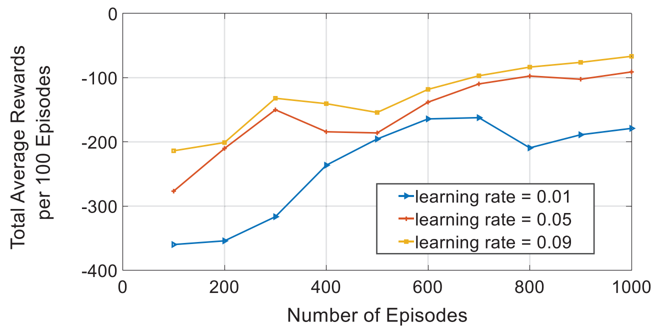

Figure 3: The total average rewards per 100 episodes within MADQNs model construction

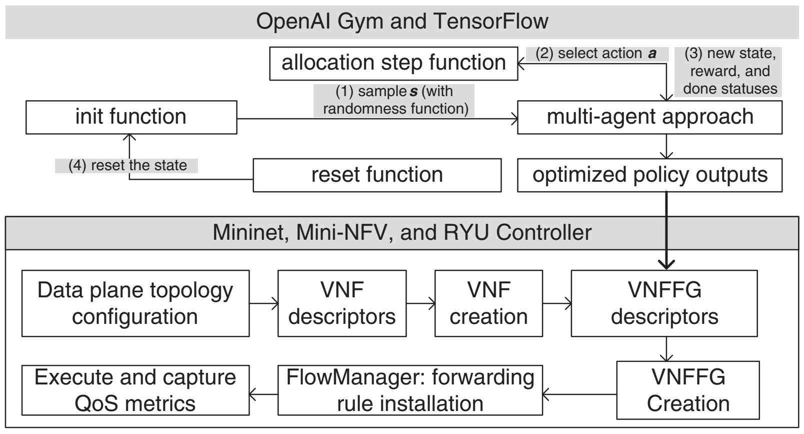
OpenAI Gym and TensorFlow
allocation step function
(2) select action *a*
(3) new state, reward, and done statuses
(1) sample *s* (with randomness function)
init function
multi-agent approach
(4) reset the state
reset function
optimized policy outputs
Mininet, Mini-NFV, and RYU Controller
Data plane topology configuration
VNF descriptors
VNF creation
VNFFG descriptors
Execute and capture QoS metrics
FlowManager: forwarding rule installation
VNFFG Creation

Figure 4: The interaction of optimal policy outputs for SDN/NFV-based control entities

A discrete-event network simulator, namely ns-3, is used in this environment to perform the E2E convergence [32–34]. The simulation was executed for 430 s, which adjusted into 4 consecutive network congestion conditions to reflect the service-learning criticalities of FL communication reliability. In this setup, there are 4 eFL nodes, and the virtual extended networks loading was configured between 0 to 250. Additionally, there are 4 remote radio heads (RRHs), and the user data rate is between 20 to 72 Mbps. The model updates will rely on the network situation, and the congestion environment will increase the loss probability between clients and aggregation servers. The congestion states lowered the model accuracy and reduced global model reliability. The payload size was set to 1024 bytes, and QoS class identifier (QCI) mechanism is set as user datagram protocol (UDP). At the core side, the point-to-point (P2P) link bandwidth was configured to 9 Gb/s, and the buffer queuing discipline was operated by random early detection (RED) queue algorithm. The default link delay of MEC was configured as 2 ms. The hyperparameters of MADQNs are prior configured to conduct the experiments with maximized output expectations in terms of computation intensity and time constraints. The learning rate α is set to 0.09 in this environment. γ, num_e, and ε values are set to 0.95, 1000, and 0.5, respectively. The main hyperparameters and parameters configuration used in overall simulation is shown in Tab. 1.

Table 1: Simulation parameters

Parameters	Specifications
Simulation time	430 s
Number of RRHs	4
Virtual extended networks loading	0 to 250
Virtual eFL MEC server	4
User data rate	20 to 72 Mbps
Payload size	1024 Bytes
P2P link bandwidth	9 Gbps
MEC link delay	2 ms
α	0.09
γ	0.95
num_e	1000
ε value	0.5

4.2 Reference Schemes and Performance Metrics

To illustrate the proposed and reference approaches in overall performances, four different resource control and eFL selection policies were simulated. The resource pools represented the capacities extraction by the proposed actions of the model. Each scheme triggered different actions, which contained the VNFFG mapping to particular virtual resources. Reference schemes were simulated in control policy for IoT congestion scenarios, including maximal rate experienced-based eFL selection (MRES), single-agent DQN-control (SADQN), and MADQNs. The proposed scheme extended PARAA and PICOA policies by enhancing the deficient actions as described in Algorithm 2.

The QoS metrics which were used to evaluate the comparison between the reference and proposed approaches are presented as follows [35,36]. *Delay* specifies the latency time of data communications from the sending node to the receiver node, including propagation, queueing, transmission, and control at the core system, which are denoted as $D^{prop}_{(J)}$, $D^{queue}_{(J)}$, $D^{tr}_{(J)}$, and $D^{ct}_{(J)}$, respectively, as described in Eq. (8). In the network simulation architecture, $J = \{1,\ 2, \ldots, j\}$ denotes the number of queueing buffers.

$$Delay = \sum_{\mathrm{J}=1}^{j} \left(D^{prop}_{(J)} + D^{queue}_{(J)} + D^{tr}_{(J)} + D^{ct}_{(J)}\right) \tag{8}$$

TP refers to the communication throughput, which expresses the successful packets delivery ratio over a given communication bandwidth *bw* (see Eq. (9)). The total, propagation, control, and processing latencies of queued *j* entities are denoted as $T^{t}_{(J)}$, $T^{prop}_{(J)}$, $T^{ct}_{(J)}$, and $T^{proc}_{(J)}$, respectively.

$$TP = \frac{\sum_{\mathrm{J}=1}^{j} \left(T^{t}_{(J)}\right) \times bw}{\sum_{\mathrm{J}=1}^{\mathrm{j}} \left(T^{t}_{(J)} + T^{prop}_{(J)} + T^{ct}_{(J)} + T^{proc}_{(J)}\right)} \tag{9}$$

The packet drop ratio in the experimental simulation is the ratio formulation between total packet lost and total packet successfully transmitted. The packet drop counts are illustrated to specifically compare in this particular experimental setup. The packet delivery ratio in the simulation environment is calculated by the subtraction between the total ratio and packet drop ratio.

4.3 Results and Discussions

The proposed agent outputted the offloading decisions of 142, 117, 371, and 370 local model updates toward 4 eFL servers, respectively. In SDN/NFV-enabled architecture, the primary consideration is the QoS metrics after installing and executing the forwarding rules [37,38]. The comparison between proposed and reference schemes is shown in Fig. 5. Within 430 s of 4 consecutive network congestion conditions, the average control delay is 8.4723 ms, which was 28.2833, 25.6824, and 11.7175 ms lower than MRES, SADQN, and MADQNs, respectively.

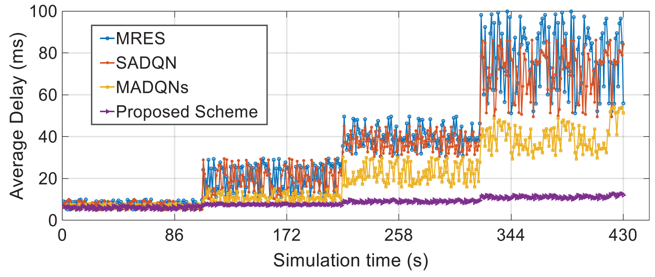

Figure 5: Comparison of average delay between proposed and reference schemes in SDN/NFV model

In E2E simulation, the emphasis of FL model reliability in real-time routing networks was considered. Fig. 6a depicted the average delays of E2E communications in the edge cloud systems. The data communication between the aggregation servers were utilized the IP network communications. The graph presented the comparisons between the proposed and reference methods with various possibilities of forwarding paths. The proposed scheme performed an average delay of 12.8948 ms, which was 64.3321, 150.9983, and 169.9983 ms lower than MADQNs, SADQN, and MRES, respectively. The proposed scheme distinguished the loading metrics of every possible serving MEC server. The predicted metrics represented the loading statuses of MEC server; therefore, the MEC, which has the lowest loading metrics, will be considered as an optimal server for serving incoming local model update requests. *TP* comparison is presented in Fig. 6b, which illustrated the notable outperformance over other approaches. The proposed scheme, MADQNs, SADQN, and MRES reached an average throughput of 659.0801, 113.7167, 50.8434, and 47.2032 bps, respectively. The proposed scheme utilized the integrated multi-agent to predict the optimal route with the lowest loading metrics for efficient eFL offloading. The average packet drops ratio of the proposed scheme significantly reached 0.0284% within 430 s simulation, which is 0.1068%, 0.1482%, and 0.1446% lower than MADQNs, SADQN, and MRES, respectively. In contrast, the proposed, MADQNs, SADQN, and MRES schemes achieved a closing packet delivery ratio of 99.9965%, 99.9853%, 99.9501%, and 99.9384%, respectively. Figs. 6c, and 6d show the graphical comparison of packet drop ratio and packet delivery ratio, respectively. Moreover, within a particular simulation setup, the packet drop counts of the proposed scheme reached a total of 1309 packets, which was 4083, 9746, and 10847 packets lower than MADQNs, SADQN, and MRES, respectively.

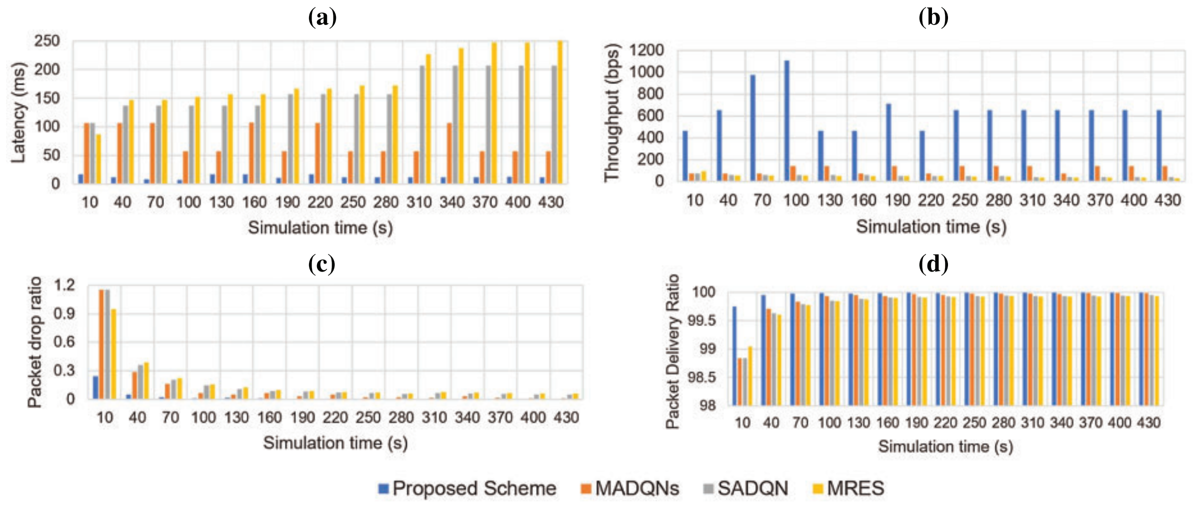

Figure 6: Comparison of (a) E2E average delay, (b) throughput, (c) packet drop ratio, and (d) packet delivery ratio between proposed and reference schemes in E2E communication perspective

MADQNs deployed the control policies of both deficient and efficient output episodes. The downlink and uplink transmission are strongly congested in heavy multi-dimensional model updates, while multiple virtual MEC is offloaded and reallocated deficiently. To gain unoccupied resource pools for QoS assurances, the proposed scheme extended MADQNs and considered the optimal resource pools for high mission-critical FL model traffics, which covers the networking states with over-bottleneck peak hour circumstances. While the extant communication and computation resources are used, the proposed controllers and orchestrator advance the positive weights ω^{+} to accelerate the serving resources from NFVI. The conditional configuration and orchestration trigger a flexible serving backup instance capacity.

In the congested FL communication networks, the local model w_k^n updates and global model W_G distributions have to transmit through long-time queueing before entering the ingress buffer of the routing or switching devices. During the heavy loading networks, the waiting time of the incoming packets can be expired and discarded before forwarding to another network. Therefore, an eNA of optimal resource allocation and sufficient eFL aggregation server offloading is applicable for enhancing reliability. In proposed scheme framework, the transmission from w_k^n to eFL node i was enhanced to aggregate reliable w_i models based on the proposed PARAA and PICOA policies. The aggregation averaging procedures between edge w_i and parameter server were executed in appropriate intervals or off-peak hours. Furthermore, the proposed approach is capable of alleviating the communication overhead for both computation and communication latency since the proposed method determined the optimal network interface with minimum cost for updating model parameters during congested situations. The proposed scheme considered the serving cost of joint entities which are efficient to each serving path. The queuing system of each SDN entity at the DP network and VNF entities were handled separately. The computation overhead and queuing system in CP were considered. Therefore, the proposed scheme avoided the data forwarding overhead from high computation intensity. In the proposed system, SDN controller was scheduled for the optimal path local model computation with adequate requests. Based on the comparisons, the proposed scheme significantly handled the routing congestion in FL communications in order to meet the criteria of URLLC key performance indicators.

5 Conclusion

This paper proposed a multi-agent approach, including PARAA for optimizing virtual resource allocation and PICOA for recommending eFL aggregation server offloading, in order to meet the significance of URLLC for mission-critical IoT model services. SDN/NFV-enabled architectural framework for controlling the proposed forwarding rules and virtual resource orchestration is adopted in software-defined IoT networks. MADQNs model interacted with the gathered state observations and contributed a collection of exploration policies for sampling the allocation rules under the expansion of edge intelligence. To obtain deficient policies, the proposed algorithms targeted weak episodes with low aggregated rewards of optimal learning rate hyperparameter. The proposed agent controller outputs a setup of long-term self-organizing flow entry with sufficient computation and communications resource placement. The optimal actions are used to correspondingly configure the VNFFG descriptors and map towards adequate virtual MEC resource pools with four experimental congestion states. The simulation was conducted in three main aspects. Based on the validation, the proposed scheme contributed a promising approach for achieving efficient eFL communications in future massive IoT congestion states.

Funding Statement: This work was funded by BK21 FOUR (Fostering Outstanding Universities for Research) (No. 5199990914048), and this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2020R1I1A3066543). In addition, this work was supported by the Soonchunhyang University Research Fund.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] F. Hussain, S. A. Hassan, R. Hussain and E. Hossain, "Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1251–1275, 2020.

[2] D. Reinsel, J. Gantz and J. Rydning, "The digitalization of the world: From edge to core," in *IDC White Paper*, Seagate Inc., Framingham, MA, USA, vol. 1, pp. 1–28, 2018.

[3] S. Kim and D.-Y. Kim, "Adaptive data transmission method according to wireless state in long range wide area networks," *Computers, Materials & Continua*, vol. 64, no. 1, pp. 1–15, 2020.

[4] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 38–67, 2020.

[5] B. Custers, A. Sears, F. Dechesne, I. Georgieva, T. Tani *et al., EU Personal Data Protection in Policy and Practice*. Heidelberg, BE, DEU: Springer, 2019. [Online]. Available: https://doi.org/10.1007/978-94-6265-282-8.

[6] W. Saeed, Z. Ahmad, A. I. Jehangiri, N. Mohamed, A. I. Umar *et al.,* "A fault tolerant data management scheme for healthcare internet of things in fog computing," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 1, pp. 35–57, 2021.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of the 20th Int. Conf. on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, vol. 54, pp. 1273–1282, 2017.

[8] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang *et al.,* "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[9] X. Mo and J. Xu, "Energy-efficient federated edge learning with joint communication and computation design," *Journal of Communications and Information Networks*, vol. 6, no. 2, pp. 110–124, 2021.

[10] Y. Ye, S. Li, F. Liu, Y. Tang and W. Hu, "EdgeFed: Optimized federated learning based on edge computing," *IEEE Access*, vol. 8, pp. 209191–209198, 2020.

[11] J. Ren, G. Yu and G. Ding, "Accelerating DNN training in wireless federated edge learning systems," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 219–232, 2021.

[12] D.-Y. Kim, S. Kim and J. H. Park, "A combined network control approach for the edge cloud and LPWAN-based IoT services," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 1, 2020. [Online]. Available: https://doi.org/10.1002/cpe.4406.

[13] Z. Li and Q. Zhu, "An offloading strategy for multi-user energy consumption optimization in multi-MEC scene," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 10, pp. 4025–4041, 2020.

[14] X. Li, D. Li, J. Wan, C. Liu and M. Imran, "Adaptive transmission optimization in SDN-based industrial internet of things with edge computing," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1351–1360, 2018.

[15] S. Shahzadi, F. Ahmad, A. Basharat, M. Alruwaili, S. Alanazi *et al.,* "Machine learning empowered security management and quality of service provision in SDN-NFV environment," *Computers, Materials & Continua*, vol. 66, no. 3, pp. 2723–2749, 2021.

[16] D.-Y. Kim and S. Kim, "Network-aided intelligent traffic steering in 5G mobile networks," *Computers, Materials & Continua*, vol. 65, no. 1, pp. 243–261, 2020.

[17] W. Chen, X. Qiu, T. Cai, H.-N. Dai, Z. Zheng *et al.,* "Deep reinforcement learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1659–1692, 2021.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness *et al.,* "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] N. Yuan, C. Jia, J. Lu, S. Gua, W. Li *et al.,* "A DRL-based container placement scheme with auxiliary tasks," *Computers, Materials & Continua*, vol. 64, no. 3, pp. 1657–1671, 2020.

[20] T. Wu, P. Zhou, B. Wang, A. Li, X. Tang *et al.,* "Joint traffic control and multi-channel reassignment for core backbone network in SDN-IoT: A multi-agent deep reinforcement learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 231–245, 2021.

[21] "OpenFlow switch specifications," Open networking foundation, 2014. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.3.4.pdf.

[22] "Network functions virtualisation (NFV); ecosystem; report on SDN usage in NFV architectural framework," White Paper, ETSI, Sophia Antipolis, France, 2015. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_nfv-eve005v010101p.pdf.

[23] "Network functions virtualisation (NFV) release 2; management and orchestration; architectural framework specification," White Paper, ETSI, Sophia Antipolis, France, 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/006/02.01.01_60/gs_NFV006v020101p.pdf.

[24] G. Brockman, V. Cheung, L. Petterson, J. Schneider, J. Schulman *et al.,* "OpenAI gym," arXiv preprint arXiv: 1606.01540, 2016.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen *et al.,* "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprint arXiv: 1603.04467, 2016.

[26] F. Chollet, "Keras," 2015. [Online]. Available: https://github.com/fchollet/keras.

[27] B. Lantz, B. Heller and N. McKeown, "A network in a laptop: Rapidprototyping for software-defined networks," in *Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, New York, NY, USA, 2010. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466.

[28] "Ryu," Faucet Organisation. [Online]. Available: https://github.com/faucetsdn/ryu.

[29] J. Castillo, "Mini-nfv framework," 2018. [Online]. Available: https://github.com/josecastillolema/mini-nfv.

[30] H. Babbar, S. Rani, M. Masud, S. Verma, D. Anand *et al.,* "Load balancing algorithm for migrating switches in software-defined vehicular networks," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 1301–1316, 2021.

[31] J. Ali, G.-M. Lee, B. Roh, D. K. Ryu and G. Park, "Software-defined networking approaches for link failure recovery: A survey," *Sustainability*, vol. 12, no. 10, 2020.

[32] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*, Berlin, Heidelberg: Springer, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_2.

[33] J. Ali and B. Roh, "An effective hierarchical control plane for software-defined networks leveraging TOPSIS for end-to-end QoS class-mapping," *IEEE Access*, vol. 8, pp. 88990–89006, 2020.

[34] S. Math, P. Tam and S. Kim, "Intelligent real-time IoT traffic steering in 5G edge networks," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3433–3450, 2021.

[35] J. Ali, B. Roh and S. Lee, "Qos improvement with an optimum controller selection for software-defined networks," *PLoS ONE*, vol. 14, no. 5, pp. 1–37, 2019.

[36] P. Tam, S. Math and S. Kim, "Intelligent massive traffic handling scheme in 5G bottleneck backhaul networks," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 3, pp. 874–890, 2021.

[37] J. Ali and B. Roh, "Quality of service improvement with optimal software-defined networking controller and control plane clustering," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 849–875, 2021.

[38] M. Beshley, N. Kryvinska, H. Beshley, M. Medvetskyi and L. Barolli, "Centralized QoS routing model for delay/loss sensitive flows at the SDN-ioT infrastructure," *Computers, Materials & Continua*, vol. 69, no. 3, pp. 3727–3748, 2021.