Hindawi
Scientific Programming
Volume 2022, Article ID 8774190, 11 pages
https://doi.org/10.1155/2022/8774190

*Research Article*

# Proactive Network Fault Management for Reliable Subscribed Network Slicing in Software-Defined Mobile Data IoT Services

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

[1]*Department of Software Convergence, Soonchunhyang University, Asan-si 31538, Chungcheongnam-do, Republic of Korea*
[2]*Department of Computer Software Engineering, Soonchunhyang University, Asan-si 31538, Chungcheongnam-do, Republic of Korea*

Correspondence should be addressed to Seokhoon Kim; seokhoon@sch.ac.kr

Received 15 April 2022; Revised 2 May 2022; Accepted 6 May 2022; Published 24 May 2022

Academic Editor: Punit Gupta



Proactive network solutions (PNS) become the precise management and orchestration (MANO) in the applied artificial intelligence (AI) era. The PNS proposed to invent future mobile edge communications by predicting the fault networks for reliable slicing configurations. Furthermore, federated learning (FL) systems have been appealed to apply for critical mobile data privacy of the Internet of Things (IoT) services. Therefore, FL-based IoT communications need a precise PNS to pretend the network failures to maximize the model inference and improve end-to-end (E2E) quality of services (QoS). This paper proposed an adopted software-defined network slicing (NS) for IoT communications based on network failure prediction and resource allocations by utilizing a deep-Q-network approach (DQN). The proposed proactive reliable subscribed network slicing was based on software-defined DQN-based proactive dynamic resource allocations (SDQN-PDRA) for adaptive communication configurations. The experiment showed that the proposed approach enhanced the significant outcomes of stability, reliability, convergence time, and other communication QoS.

## 1. Introduction

In next-generation (NG) communication technology, the mitigated mobile edge computing (MEC) from the remote cloud, called mobile cloud computing (MCC), is intended to empower fronthaul computing resources and enhance NG infrastructure as a service (IaaS) for overcoming heterogeneity novelty applications, including Internet of Things (IoT), heterogeneous IoT (HIoT), Internet of healthcare things (IoHT), and Internet of Vehicles (IoV), and especially, for the time critical-communications [1–3]. MEC plays an essential role in enabling local services for 5G perspectives, which aim to provide agile response services for user devices with ultra-dense new radio (NR) services for massive user terminals. According to the enlargement of edge network infrastructure, intelligent resource management and orchestration (MANO) towards autonomous network configurations have become the critical research areas [4]. Additionally, network systems self-organizing networks (SON) must be enhanced because autonomous networks can be empowered by adopting artificial intelligence (AI) algorithms. Deep learning (DL) models were introduced to handle and improve SON perspectives, especially in distributed network areas.

DL approaches have contributed effectively to the large-scale complexity of network datasets in terms of classification, recommendation, and prediction problems [5]. DL can be applied for efficient MANO heterogeneous network resources for these reasons. Especially in IIoT applications, various IoT devices will generate a large-scale network dataset. For example, in the federated learning (FL) based IoV paradigms, each vehicle has its data cloud to store and compute privacy constraint information and share its training model and model parameters to the distributed cloud server for aggregation. The training model will be shared between MEC servers in the vehicle edge networks (VEN) [6–9]. Because of the high-speed movement of the vehicles, the exchange of information between MEC servers

is obligated to be performed with high stability and low latency [10, 11].

Moreover, to cope with massive data generated from high-speed mobility, a local cloud system must be installed for local training. To achieve reliable networking for end-to-end (E2E) communications, data integrity from the sensor is essential, and in-networking communications are required to ensure communication reliability. In FL systems, E2E round-trip communication between V2C requires ultra-reliability low latency communication (uRLLC) [12]. The software-defined routing (SDR) based on software-defined network (SDN) architecture plays a significant role as a global routing approach, which handles multipath forwarding in heterogeneous edge servers. An intelligent routing approach is essential in the VEN solution, while DL can be deployed to classify the different levels of the link statuses or reliable edge servers [12]. The adopted DL for intelligent SON with SDN architecture leverages the NG communication system towards efficient big data network solutions to empower E2E QoS and QoE based on experiential networked intelligence (ENI) [13].

DL models play essential roles in big data network data solutions based on the ENI architecture. Moreover, the intelligent softwarization network can be established with an open interface for AI infrastructure. This paper selected the DL algorithm, namely, recurrent neural network (RNN), to perform network failure prediction in the distributed edge servers for proactive network solutions. Generally, to perform network and FL system assurance, experience networking is crucial for evaluating future configuration for the coming request [14, 15]. In the distributed network, DL models empower the QoS based on the observation of QoE data points. RNN gains the top position for prediction purposes over the convolutional neural network (CNN) [14]. Meanwhile, CNN is popular and works well with image-sensing data gathered from sensor devices. In the VEN, the loading occurs with sudden fluctuation by the time spaces, whereas the RNN models are primarily based on long short-term memory (LSTM) structure with multiple gates.

Moreover, the extended reinforcement learning (RL) called deep reinforcement learning (DRL) approaches utilized DL approaches to recommend the optimal $Q$-value (state and action paired) for optimal action space selection purposes [16, 17]. DQN approaches have been comprehensively intentioned in the mobility network resource allocation, offloading, and E2E network MANO [17–19]. DRL provides the ability to adapt to natural network environments to train the agents to handle network issues [19–22]. This paper proposed an SDN-based subscribed network slicing DQN network loading adjusting when the loading metrics of the network devices exceed the defined threshold interval. SDN controller assures E2E communication reliability for model transferring between clients and aggregation servers by the global view of SDN controller with proactive fault detection and resource allocations [23–27].

The main contributions of the paper are encapsulated as follows.

We deploy ENI-based architecture for network condition predictions and resource allocations with the integrated DQN model. After that, the collected network statuses turn to the classification phase for distinguishing the distinct network conditions (e.g., MEC loading, traffic loading, path delay, etc.), which are essential for SDR rulemaking. We deploy DQN for autonomous resource management and to minimize network load fluctuations. SDN controller invests the DQN model for optimal Q-value selection, by implementing DQN in SDN architecture to explore the most appropriate action for allocating the resource to maintain the optimal network loading metrics.

We provide the E2E evaluation metrics of our proposed SDQN-PDRA with various approaches in three communication aspects, including the model convergence reliability of FL in IIoT, network stability, and QoS analytics. The intelligent network computation and configuration were based on proactive network solutions (PNS). The SDN controller was considered to handle the loading predictions and adjustment proactively.

Continuing of the manuscript is organized as follows. Section 2 presents the related work and IIoT communication system and issues. In addition, our solution is described in Section 3. The experiment and numerical evaluation results with detailed interpretation are given in Section 4. Finally, Section 5 presents a conclusion and future work.

## 2. Related Work

Each WSN is attached to a personal edge server with a sufficient local cloud system to perform data storing, local training model, and other significant computations (see Figure 1). Whenever the local edge has insufficient resources to operate heavy traffic, there will be a high network delay with the congestion caused by the network failure [27, 28]. The FL-based system uses model transferring instead of raw data sharing [29]. FL-based network architecture reduces the amount of traffic over the network since the raw data captured from the sensor network is stored in the local cloud and performs local training [30].

The FL-based communication can be described in three layers. First, the data gathering and local training layer required to satisfy KPI obligations in terms of data integrity, clearance of the dataset, and the sensor can be sensing overdetection information that will not be utilized for model evaluation due to the PNS required reliable model inference. Moreover, some of the computation and decisions must be made inside the personal edge service to support an in-network processing conducted based on band communication over Ethernet or wireless communications.

However, the synchronization processes between local and global entities will be frequently made, while the vast local models require aggregation transfer to edge servers. At the same time, the heavy computation is not suitable for computing inside a local server, especially the missing data for model decisions. Moreover, due to the fast speed and high IoT networking, the joining radio networks consist of high failure ratios that obligate handling for uRLLC. Additionally, the optimal remote radio head (RRH) recommendations will diminish the failure ratio of joining the radio access network (RAN). The aggregation server shares its model with the server in charge of continuous computing

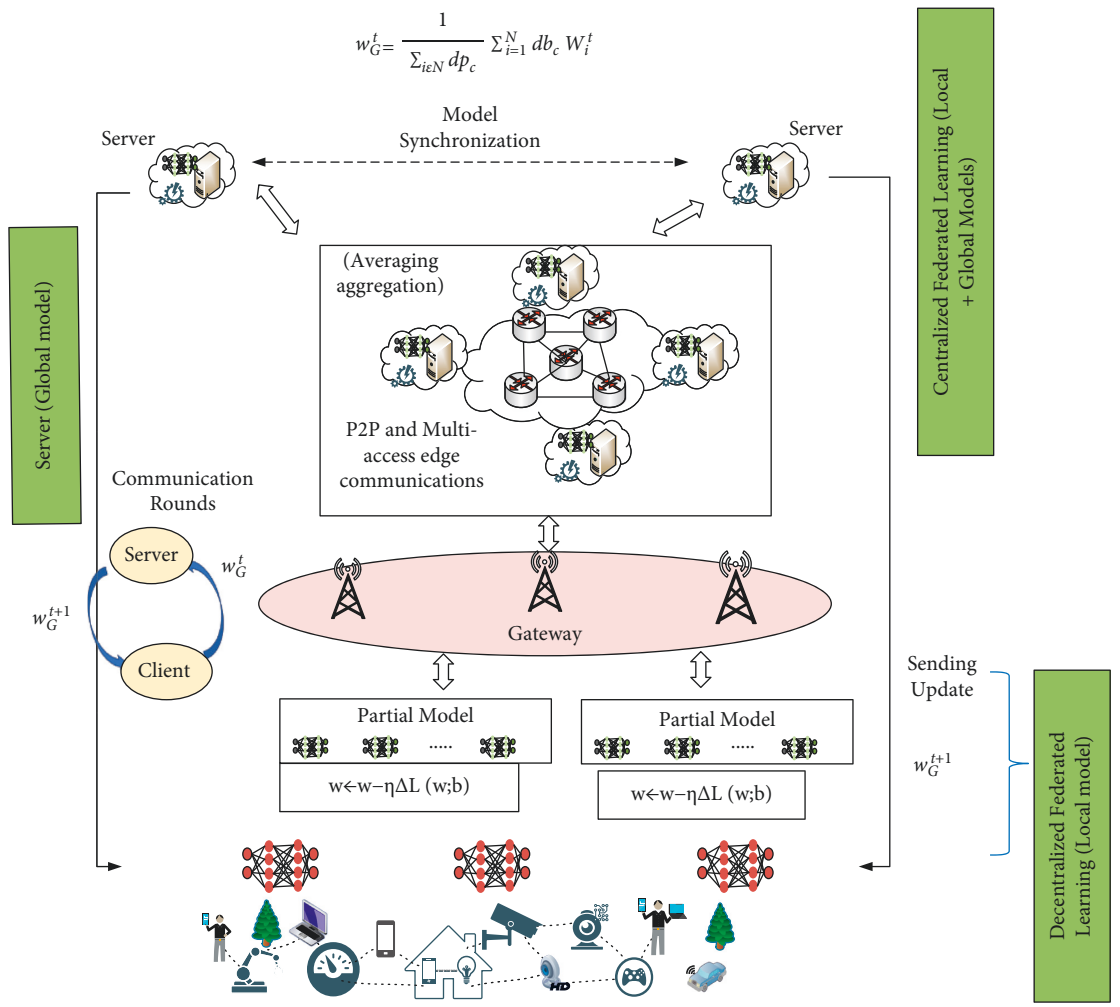


Figure 1: The FL-based IIoT environments exchange information between local and aggregation sever.

in the handover process whenever the alternative is mandatory. In the process of sharing, heavy traffic will be generated, and new route installation will be obligated.

Moreover, the rapidly converged network will become a key challenge issue when each router has a large-scale network database that takes long periods of computing processes. Therefore, PNS (advanced computation) and route installation in time critical IoT reduce delays for massive routing decisions. Additionally, the rapid routing will utilize SDN architecture since the routing computation and installation will be conducted in the control plane (CP), while the data plane (DP) takes a function of data forwarding based on the installed route [31–34]. Thus, the computation can be wholly separated and performed in advance in the CP.

*2.1. System Model.* The system model is based on the ENI architecture by converging the three primary contributors, including DQN, caching, and SDN controller. The network conditions in terms of delay, congestion window, and resource limitation are considered network loading parameters.

The local server has been attached to the IoT system in federated IoT to store the sensed data from various intelligent sensor devices. The local training is conducted by splitting the local dataset between each client into minibatches of size $db$ , which are included in the set $c = \{db_1, db_2, \ldots, db_c\}$. The local trained and updated models are sent to the global servers for aggregation which can be modeled as follows:

$$w^c_{\text{update}} = \text{global } w_g \text{parameters} - \alpha \Delta MSE\left(w^c_{\text{update}}; db_c\right). \quad (1)$$

While $w_{\text{update}}$ is the model parameter update from local IoT clients $w^c_{\text{update}} \in \{w^1_{\text{update}}, w^2_{\text{update}}, \ldots, w^c_{\text{update}}\}$., local data minibatches of total client $c \in \{db_1, db_2, .., db_c\}$, and MSE is the Mean Squared Error representing the loss function for deep neural network (DNN). The local client transmits the updated model $w^c_{\text{update}}$ over the wireless networks to the aggregation server. The global server collects the up-to-date model $w^c_{\text{update}}$ from various aggregation servers for model accumulation. The global server will send the average global models to the local client. The global model $w^t_G$ can be modeled as

$$w^t_G = \frac{1}{\sum_{i\epsilon N} db_c} \sum_{i=1}^{N} db_c W^t_i, \quad (2)$$

where $w^t_i$ is the updated model at each time step $t$. $w^{t+1}_G$ is the global update summation at time $t+1$; the increasing

number of the round-trip time (RTT) communications from local to server will boost the global training accuracy. However, the number of RTT model communications will reduce the E2E transmission QoS. Generally, RTT of FL communication, communicating over the 5G communication technology, consists of poor service identity towards intelligent handling for service level agreement (SLA) that requires additional resource allocations with the high-level priority.

*2.2. Communication Overhead.* The overall overhead, which reduces the network QoS, can be expressed based on the $M/M/c/K$ queuing system. In each network node and aggregation server, the serving overheads at a particular queue interface can be determined and modeled as $M/M/1/K$ queuing system with the limitation of $K$ capacity of the server. $\rho$ is the serving ratio between the arriving task $\lambda$ and serving resource $\mu$. For $\mu > 0$, user traffic $P_0$ and $n$ user's traffic $P_n$ in the system can be measured as follows:

$$\rho = \frac{\lambda}{\mu}, \text{for } \mu > 0,$$

$$P_0 = \begin{cases} \dfrac{1-\rho}{1-\rho^{K+1}}, \text{for } \rho \neq 1, \\ \dfrac{1}{K+1}, \text{for } \rho = 1, \end{cases} \tag{3}$$

$$P_n = \begin{cases} \dfrac{(1-\rho)\rho^{n}}{1-\rho^{K+1}}, \text{for } \rho \neq 1, \\ \dfrac{1}{K+1}, \text{for } \rho = 1. \end{cases}$$

The mean waiting time $\overline{Q}$ length in the single individual edge server can be measured as

$$\overline{Q} = \begin{cases} \dfrac{\rho}{1-\rho} - \dfrac{K\left(K\rho^{K}+1\right)}{1-\rho^{K+1}}, \text{for } \rho \neq 1, \\ \dfrac{\rho(K-1)}{2(K+1)}, for\, \rho = 1. \end{cases} \tag{4}$$

Then, with the mean number of user traffic $\overline{N}$ in the single individual edge, the system can be modeled as

$$\overline{N} = \overline{Q} + (1 - P_0). \tag{5}$$

The communication overhead will be considered in the computation and communication delays. Furthermore, we denote the communication rate between serving nodes ($i$ *to* $k$ interface) in wired-based networks with bandwidth as $W$ and transmission power as $P$, the noise power between the $i$ to $k$ interface as $\partial^2_{i,k}$, and communication channel gain as $H_{i,k}$. Thus, the transmission rate from the aggregation server to another global server can be expressed as

$$TR_{1,k} = W \log_2\left(1 + \frac{P_1 H_{1,k}}{\partial_k^2}\right). \tag{6}$$

$k \in \{2, \ldots, K+1\}$ and $i \in \{2, \ldots, I+1\}$, and $i \neq k$; then, the transmission rate between $i$ and $k$ nodes can be expressed as

$$TR_{i,k} = W \log_2\left(1 + \frac{P_i H_{i,k}}{\partial_{i,k}^2}\right). \tag{7}$$

## 3. Our Solution

SDN-based DQN will allocate resource in advance for feasible flow handling for every predicted loading metric DN-based DQN appraoch performs the allocation processes and optimal parth selections (see Figure 2). The network loading metric will be under the defined threshold in the optimal state. The forwarding process of the information from the local to global servers is according to the predicated and resource adjustment schemes. SDN controller establishes the routing policy according to the comparisons of real-time observation metrics; the path with minimum metric is considered for feasible routing path. However, in the case that loading metric at each possible routing path has loading metrics more than the defined threshold (nonoptimal state), the SDN controller will be adjusted resource by attempting to query the optimal action $a^t_{i,k}$ until the observed condition reaches an optimal state. The SDN controller will select the optimal gateway to ensure the transferring of updated model parameters of local devices and the aggregated model download from the aggregation server (see Algorithm 1).

The ENI infrastructure will not be utilized in steady situations, and the SDR will be coordinated to handle the model sharing based on the current route. This method will be helpful in the stability of communication, while DP communications share similar loading metrics. However, the caching metric will not be utilized for route configuration in the frequently fluctuating communication statuses since the feasible optimal aggregation server is obligated to be defined. Furthermore, the IIoT applications are unsuitable for utilizing the restriction scheme, which restricts sending resources during the heavy loading network, increasing the waiting time at the source devices.

*3.1. Joined Environment for Resource Adjustment.* To evaluate autonomous resource allocations, the environment with varied network stability between 0 and 255 represents the loading metric randomly. A variety of entities can accomplish the IoT environment in terms of data plane conditions called state space $s_t$ and the SDN controller considers taking action $a_t$ at each time-space $t$ based on the optimal policy $\pi^*$, which provides the maximum $Q$-value. The detailed description is explained in the following.

*3.1.1. State Space.* At each time-space $t$, the SDN controller maintains the below information, which can be affected by the network conditions.

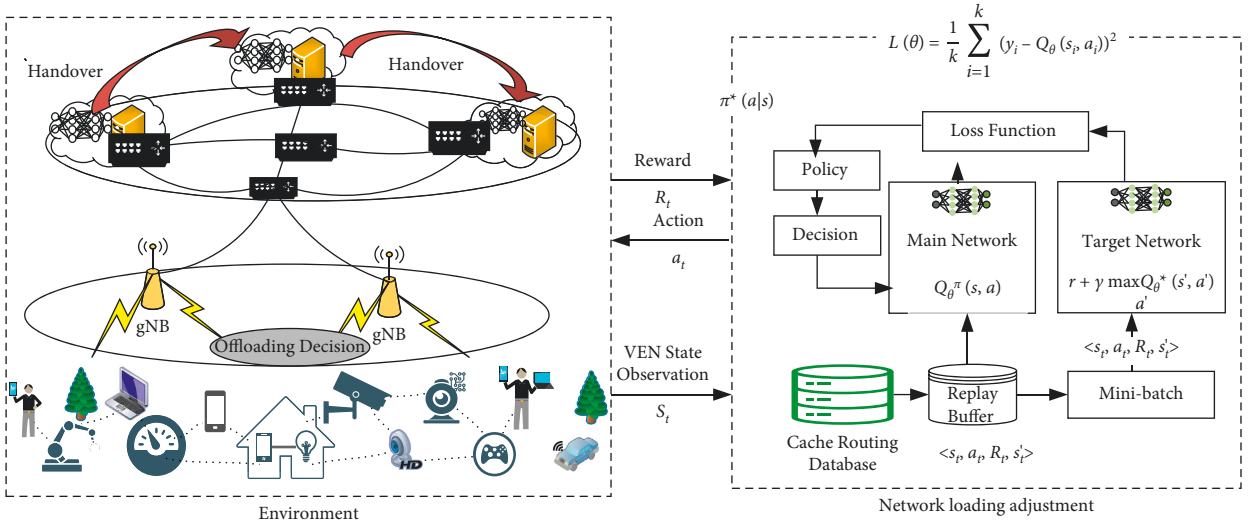


Figure 2: The proposed software-defined DQN for resource allocation after fault network predictions.

(i) $W_{i,k}$ is the bandwidth of the wired between $i$th to $k$ interfaces

(ii) $I_{i,k}$ is the number of the assigned tasks.

(iii) $\overline{N}$ is the number of the user requests at the time $t$.

(iv) $\overline{Q}$ is the average number of queue lengths at time $t$.

(v) $TR_{1,k}$ is the transmission rate of the wired between serving entity at the time $t$

Loading metrics are at interfaces between $R \in \{1, 2, 3 \ldots, R\}$ and $M \in \{1, 2, 3, \ldots, M\}$ network devices at the time $t \in \{1, 2, 3, .., \tau\}$.

So, the above information can be the entities of the system state space $S$, that is, $S \in \{W_{i,k}^\tau, I_{i,k}^\tau, N_{i,k}^\tau, Q_{i,k}^\tau, R_{i,k}^\tau\}$, due to the communication bandwidth at $i$ and $k$ interface in the wired link, will be sharing the same metric for the system state information $s_t \in S$ and can be expressed as

$$W_{i,k}^t = W_i^t = W_k^t. \tag{8}$$

*3.1.2. Action Space.* The agent takes a significant role in deciding the optimal action for $I_{i,k}$ flow requests according to the network states. In this action space, the agent will consider optimal action to meet the defined state information. We denote $A$ as our global action space at each time-space at the time $\tau \in \{1, 2, 3, .., t\}$, and $a_{i,k}^\tau$ is the action tacking at the $i$ and $k$ interfaces between $R \in \{1, 2, 3 \ldots, R\}$ and $M \in \{1, 2, 3, \ldots, M\}$ network devices. While $A$ denotes the global action space, $a_t \in A$ performs at time slot $t, a_t \in \{a_{i,k}^1, a_{i,k}^2, a_{i,k}^3, \ldots, a_{i,k}^t\}$; then $a_{i,k}^t$ is varied corresponding to the state space information.

*3.1.3. Reward Calculation.* In our system, the the agent randomly selected action at each time $t$, and the reward will be offered based on the network state information. The feedback from the environment trains the agent to determine optimal action, and the optimal action $a_t$ performs the optimal state. After $a_t$ was taken, the reward $r_t$ was immediately provided.

$$R_t = s_{\text{optimal state}}(t) - s_{\text{congested state}}(t), \tag{9}$$

$$s_{\text{optimal state}}(t) = \sum_{0<i\le t, \forall r_i=1} r_i, \tag{10}$$

$$s_{\text{congested state}}(t) = \sum_{0<i\le t, \forall r_i=-1} r_i. \tag{11}$$

Based on equations (9)–(11), $\forall r \in R_t$; the rewards metric $r$ represents the good and bad states at each time-space. Further, the reward accumulation presents the optimal network condition for entire communication periods.

*3.1.4. Optimal Policy.* The reward $R_t$ metric is corresponding to the system environment status based on the optimal action $a_t$ selection. The agent explores the optimal action for reducing the network loading metrics. Based on the experiences, the agent can select the optimal action $a_t$ at the similar network states based on the experiences. The agent chooses action according to the policy $\pi$, and the optimal policy $\pi^*$ can return the maximum sequence corresponding reward $R_t$. To maximize the $Q$-value, the action required the optimal policy $\pi^*$, and our Q-value can be obtained by utilizing Bellman optimal equation

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i | \pi, s_t = s, a_t = a\right]. \tag{12}$$

The optimal $Q$-value $Q^*(s, a)$ at state $s$ and action $a$ with the policy $\pi$ will correspond to the discount parameter $\gamma$, and the weight in $0 \le \gamma \le 1$ will reflect the agent to put on the future rewards. Moreover, for any main network $\theta$ and target

$X \in \{x_1, x_2, \ldots x_n\}$ denotes $n$ of general features as and $Y \in \{y_1,\ y_2,\ y_3 \ldots .\ y_m\}$ denotes $m$ target features in the global dataset required for client and server
(1) Initialize the synchronous $c$, $n_{epoch}$, $w_{\text{update}}$ parameters, $\alpha$, $w_g$ for aggregation server
(2) Ensure the optimal sharing path between client and server, and the routing path (see Algorithms 2)
(3) **[Aggregation Server]**
(4) for each epoch in range ($n_{epoch}$)
(5) Select lowDNN() for $c$ clients
(6) Aggregating the model $w_g$ for the next epoch by using the FedAvg algorithm [35].
(7) **end for**
(8) **[Client Server]**
(9) for each $db_c$ in (Data) do
(10) Input $\alpha$, $w_g$, and $w_{\text{update}}$ parameters
(11) Define class lowDNN(self, $\alpha$, $w_g$):
(12) for each client in $c$ do
(13) $w^c_{\text{update}} = w_g - \alpha \Delta MSE(w^c_{\text{update}})$
(14) **end for**
(15) **end for**

Algorithm 1: Exchange model between client and server.

(1) Initial the main, target parameters, and replay buffer, $\theta, \theta\prime$, and $\mathcal{D}$, respectively
(2) Define $N$ number of episodes
(3) **for** each step in the episodes, **then**
(4) State $s_t$ observation
(5) DQNagent selects action based on the optimal policy $\pi^*(a|s)$
(6) Action $a_t$ selection and explore the next state $s\prime$ and obtain the reward $R_t$
(7) At each time slot $t$ SDN controller executes the action $a_t \in \left\{a^1_{i,k}, a^2_{i,k}, a^3_{i,k}, \ldots, a^t_{i,k}\right\}$,
(8) **if** the size of $s_t, a_t, R_t s'_t$ the size of replay buffer $\mathcal{D}$
(9) cache $s_t, a_t, R_t, s'_t$ into the replay buffer $\mathcal{D}$
(10) **else**
(11) Replace queue tail element with the current $s_t, a_t, R_t, s'_t$ as the FIFO process.
(12) **End if**
(13) Transition to next network state $(s_t \leftarrow s'_t)$
(14) Random minibatch of $k$ samples from replay buffer $\mathcal{D}$
(15) Compute the target network value:
(16) $y_t = r_t + \gamma \max_a Q^*(s'_t, a')$
(17) Compute and minimize the loss:
(18) $L(\theta) = 1/k \sum_{i=1}^{k} (y_i - Q_\theta(s_i, a_i))^2$
(19) Update the target network $\theta'$, based on the updated $\theta$:
(20) $Q^*(s, a) = r + \gamma \max_{a\prime} Q^*_\theta(s', a')$
(21) **End for**

Algorithm 2: SDQN-PDRA.

network parameter $\theta\prime$, the optimal policy $\pi^*(a|s)$ that performs maximum $Q$-value in paired state $s$ and action $a$ is denoted as

$$\pi^*(a|s) = \arg\max_a Q^\pi_\theta(s, a), \tag{13}$$

$$Q^*(s, a) = r + \gamma \max_{a'} Q^*_\theta(s', a'). \tag{14}$$

From (12), the agent explores the action policy in the obvious state to find the maximum $Q$-value. Then optimal $Q$-value is relative to the summation of rewards and discounted parameter multiply of maximum $Q$-value of next state $s'$ and action $a'$, as depicted in (14). However, when $\gamma$ is close to 1, the system will suffer from the computation delay since the agent has more chances to discover the optimal $Q$-value. Consequently, our target network $y_i$ will be written as

$$y_i = \begin{cases} r_i \text{for } \gamma = 0, \\ r + \gamma \max_{a\prime} Q^*_\theta(s', a') \text{for } 0 < \gamma < 1. \end{cases} \tag{15}$$

Then, the loss between target and actual network parameters of $k$ minibatches can be expressed as

$$L(\theta) = \frac{1}{k}\sum_{i=1}^{k}\left(y_i - Q_\theta(s_i, a_i)\right)^2 \quad (16)$$

The SDQN-PDRA is based on the resource adjustment scheme to reduce the loading metrics at each loading period (see Algorithm 2). The SDN controller adjusts the observed loading metrics based on the DQN method. We suppose each serving server can retrieve the global resource from the root or global MEC server in this scenario. The network loading metric will be under the defined threshold in the optimal state. The forwarding process of the information from the local to global servers is according to the predicated and resource adjustment schemes. SDN controller establishes the routing policy according to the comparisons of real-time observation metrics; the path with minimum metric is considered for feasible routing path. However, in the case that loading metric at each possible routing path has a loading metric more than the defined threshold (nonoptimal state), the SDN controller will be adjusted resource by attempting to query the optimal action $a_{i,k}^t$ until the network condition reaches an optimal state.

## 4. Numerical Evaluation

This section provides a precise description of the used system evaluation parameters for experiment installation in terms of experimental parameters, hyperparameters, and experiment components used to conduct E2E simulation. Furthermore, the numerical result evaluation of the prediction model, model convergence accuracy between client and server in various network conditions, and efficiency of resource adjustment based on SDQN towards the evaluation of communication QoS are performed.

*4.1. Simulation Environments.* During simulations, the captured delay was utilized to represent the real-world network loading metrics. In addition, the opened source dataset EMNIST [35] was loaded from the federated EMNIST to evaluate the converged network reliability. The EMNIST dataset was sliced to meet the number of clients for testing using the Google platform. Each client has each slice of the dataset (individual dataset) and training model, and the aggregate server utilizes the FedAvg function offered by TensorFlow Federated [35]. Moreover, the E2E evaluations were based on the simulated metrics captured from the NS3 [36] simulations (see Table 1) and the hyperparameters and the experiment components, respectively (see Tables 2 and 3).

*4.2. Results and Discussion.* The state observation during 1000 episodes in a real-world VEN environment was done and we applied our DQN approach for adjusting the state space metric to meet the determined optimal states threshold, which was set at $\forall s_t \in \{0, 1\}$ (see Figure 3).. The VEN states in natural communication consist of the average bad network state and good network state counts at 397.724 and 32.276, respectively (see Figure 3(a)). In some cases, the natural VEN environment has 0 optimal and 430 bad states (100% of observed states are in bad conditions). With the $\gamma = 0.85$, and learning rate $lr = 0.02$, our proposed resource adjustment reduced the communication overhead in VEN and reached the average optimal state count at 413.1537688 and the bad state counts of 16.84623116 (see Figure 3(b)). With this effective result, in some episodes, the proposed scheme reached 100% (430 optimal states) optimal state handling with 0% (0 bad states) of bad network state counts. Based on the notable metrics, our scheme performed the optimal state up to 96.08227%.

Four conditional simulations of the federated model experiments are conducted to emphasize FL model reliability in actual situation network routing (see Figure 4). The graph presents the remarkable outperformance of the convergence accuracy based on optimal network selected path (ONSP) over the three other possibility routing paths; for example, simple congestion of network chosen path (SCNSP), congestion of network selected path (CNSP), and heavy congestion of network selected path (HCNSP) were simulated to reflect the model reliability in network

Table 1: The experiment parameters.

| Network parameters | Values |
|---|---|
| RRH | 4 |
| Network loading | 0 to 250 |
| MEC server | 4 |
| User data rate | 20 to 72 Mbps |
| Packet size | 1024 bytes |
| P2P link | 9 Gbps |
| Interface link (P2P) delay | 2 milliseconds |
| Simulation time | 430 seconds |

Table 2: The hyperparameters' description.

| Hyperparameters | Detailed value |
|---|---|
| Minibatch size | 32 |
| Optimizer | Adam/SGD |
| Loss function | RMSE |
| The activation function of 1st hidden layer | Tanh |
| The activation function of $n$th hidden layers | ReLu |
| The activation function of the output layer | Linear |
| Replay buffer $\mathcal{D}$ | 5000 |
| $lr$ | 0.02 |
| Discount factor $\gamma$ | $0 \le \gamma < 1$ |
| Optimal state space (threshold) | $0 \le s_t \le 1$ |
| Policy $\pi$ | BoltzmannQPolicy |

Table 3: The experiment components.

| Tools | Detailed description |
|---|---|
| OS | Ubuntu 20.04.2 LTS Rease 20.04 |
| NS3 | 30.0 |
| TensorFlow-CPU | 2.3.0 |
| Matplotlib | 3.02 |
| RAM | 16 GB |
| Program | C++, Python (version 3.7.1) |
| Keras | 2.2.4 |

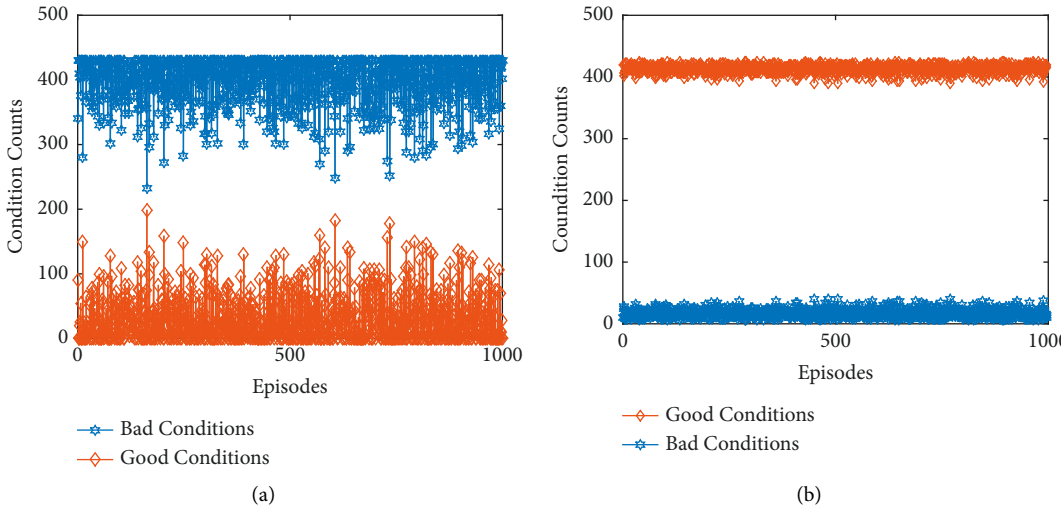


Figure 3: The state observation comparison between (a) natural network (random) and (b) loading adjustment.

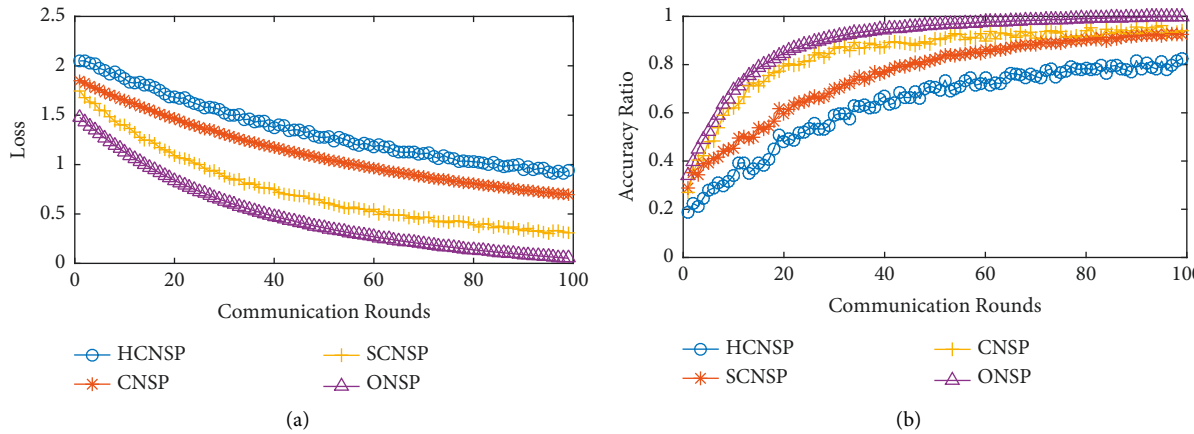


Figure 4: The convergence of model sharing performance metrics comparisons between HCNSP, SCNSP, CNSP, and ONSP network statuses represented the routing in loading periods.

environments. The mean training loss metrics in 99 communication rounds of ONSP, SCNSP, CNSP, and HCNSP is at 0.484335, 0.743309, 1.129355, and 1.354101, respectively (see Figure 4(a)). And the minimum loss of the ONSP, SCNSP, CNSP, and HCNSP is at 0.052342, 0.309779, 0.694963, and 0.94137, respectively. Based on the mean loss comparisons, the ONSP has a lessened loss metric compared to SCNSP, CNSP, and HCNSP at 0.64502%, 0.258974%, and 0.869765%, respectively. The model aggregation will rely on the network situations. The congestion environment will lead to a loss in mode sharing between aggregation servers, and it can cause low accuracy in terms of global model reliability. The E2E model reliability corresponds to the global model accuracy comparison between the ONSP, SCNSP, CNSP, and HCNSP (see Figure 4(b)). The ONSP approach reached the maximum accuracy metric at 0.998873%, while the SCNSP, CNSP, and HCNSP have the accuracy of 0.941435%, 0.925075%, and 0.825702, respectively. The ONSP enhanced the other possibility routing paths based on the numerical comparison at 0.058974138%, 0.14566234%, and 0.270408005%, respectively. Due to the ONSP delivering the optimal scheduling approach, the network loading metrics have lessened based on the proactive network configurations. Furthermore, the proposed ONSP approach will also enhance the possibility of saving the computation power in the CP.

In terms of E2E communication QoS metric evaluation, we compared our proposed integrated software-defined DQN in proactive resource allocations (SDQN-PDRA) with the other approaches, including software-defined RNN in dynamic routing (SDRDR), software-defined dynamic routing (SDDR), and software-defined experience routing (SDER). Our proposed SDQN-PDRA approach illustrated remarkable outperformed results over the other approaches, such as SDRDR, SDDR, and SDER, in terms of the packet drop counts, packet drop ratio, packet delivery ratio, and communication delay, respectively (see Figure 5).

The natural network environment consists of limited network loading awareness for improving the routing experience. Thus, the local and external data sharing can be

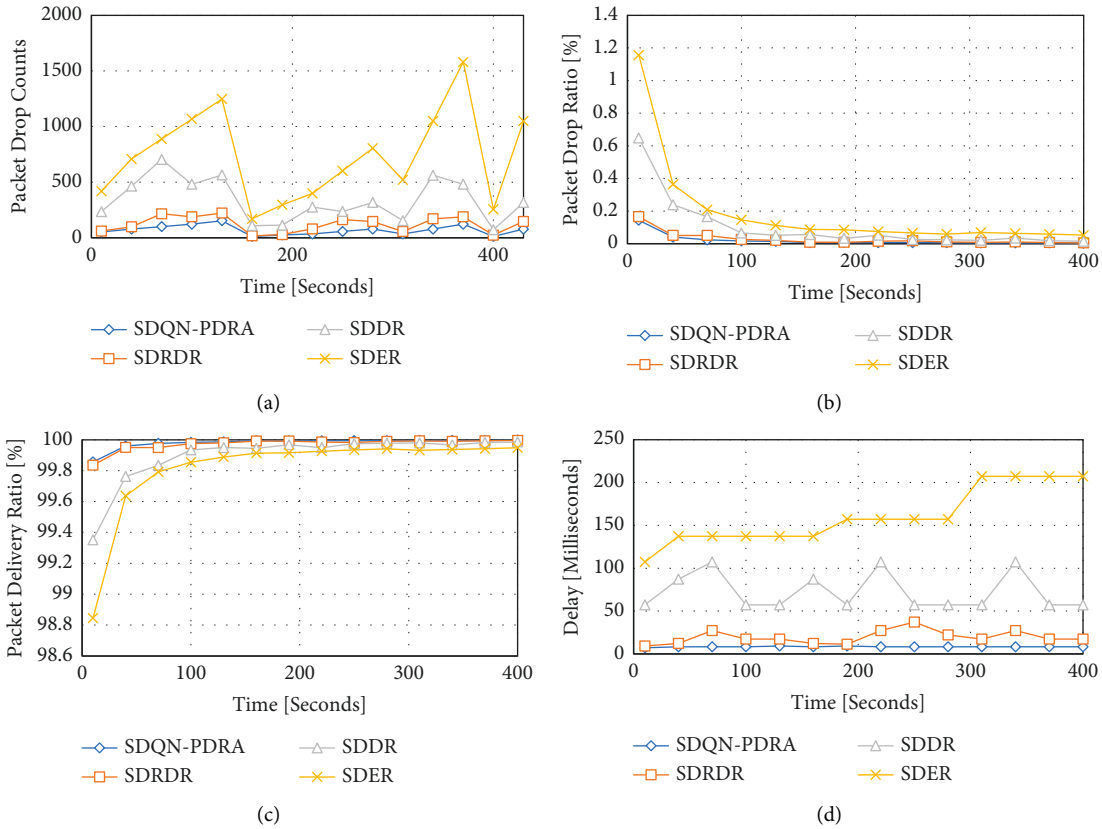


Figure 5: The presentation of E2E communication QoS metrics comparisons of proposed SDQN-PDRA with SDRDR, SDDR, and SDER in terms of (a) packet drop counts, (b) packet drop ratio, (c) packet delivery ratio, and (d) communication delay, respectively.

elaborated as static and dynamic routing protocols. Likewise, the static and dynamic routing protocols have a weakness when considering selecting the optimal path with efficiency costless. Our proposed SDQN-PDRA provided high accuracy at loading shape network prediction for the lowest cost routing and can reduce the loading state to meet the defined optimal conditions metric threshold. Our proposed SDQN-PDRA obtained the lowest packet drop counts over the SDRDR, SDDR, and SDER in mean metrics at 69, 120.2666667, 339.0666667, and 737, respectively (see Figure 5(a)).

Our proposed SDQN-PDRA obtained the lowest packet drop ratio over the SDRDR, SDDR, and SDER: 0.01925501%, 0.027260413%, 0.097242413%, and 0.176594407%, respectively (see Figure 5(b)). Therefore, the proposed SDQN-PDRA lessened the E2E communication loss between client and server in network environments based on the given graphs. For the E2E communication reliability for the communication drop ratio, see Figure 5(c). Our proposed SDQN-PDRA achieved the highest E2E communication reliability. The average E2E communication reliability of SDQN-PDRA, SDRDR, SDDR, and SDER was achieved at 99.98074499%, 99.97273959%, 9.90275759%, and 99.82340559%, respectively. Based on the presented reliability metrics, our proposed SDQN-PDRA is 0.008005403%, 0.077987403%, and 0.157339397% higher than the communication reliability metric of SDRDR, SDDR, and SDER, respectively.

The selected routing path containing high loading metric will suffer computation overhead, which can postpone the serving request and increase the waiting time of arrival traffic. Moreover, the network buffer can be limited whenever the serving rate is under the requested tasks. To cope with these issues, real-time loading resource reduction plays an essential role in improving the communication experience. The E2E communication delay between the proposed SDQN-PDRA approach and SDRDR, SDDR, and SDER approaches has been presented (see Figure 5(d)). The proposed SDQN-PDRA effectively reduced the network loading metric and selected the optimal path for route installation, and the communication delay was completely reduced for E2E data sharing. Based on the graphs, the proposed SDQN-PDRA reached the minimum average delay at 8.294905267 milliseconds, while the SDRDR, SDDR, and SDER consume a higher delay at 19.62816013 milliseconds, 71.227377 milliseconds, and 163.8931339 milliseconds, respectively. Furthermore, our proposed SDQN-PDRA quickly responded 11.33325487 milliseconds, 62.93247173 milliseconds, and 155.5982286 milliseconds faster than SDRDR, SDDR, and SDER, respectively.

## 5. Conclusion

This paper proposed an intelligent SDR based on the integrated SDN-based RNN-based traffic loading prediction and DQN for network loading adjustment (SDQN-PDRA)

for reliable FL-based IIoT. It is worth noting that IoV communication perspectives are obligated to be handled as a real-time service in distributed edge routing. Moreover, the high-speed mobility sensor will face many challenges in data processing, ultra-high mobility communication, and frequently alternative edge cloud computing. The FL-based IoT will be a large-scale distributed cloud requiring intelligent routing that effectively performs URLLC in routing and network convergence processes. The network assurance plays an essential contribution to reliable FL in IoT systems, while the reliability network systems influence the reliability of FL convergence models in terms of accuracy and decision making. Our proposed SDQN-PDRA approach prevents routing failure and adjusts the network condition to meet the optimal states. The proposed SDQN-PDRA provided remarkable contributions to IoT systems regarding IoT stability conditions and E2E communication QoS, including reliability, latency, and communication throughput. For future work, we will explore the computation cost influence on the communication overhead in routing and expand the routing environment to reflect real-world IoT communications in the 5G system. Furthermore, the SDN-based multidimensional deep-Q-network approaches will be invested in improving the autonomous routing policy.

## Data Availability

The data and finding are included within the article.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

This work was funded by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2020R1I1A3066543) and BK21 FOUR (Fostering Outstanding Universities for Research) (no. 5199990914048). In addition, this work was supported by the Soonchunhyang University Research Fund and by the Bio and Medical Technology Development Program of the National Research Foundation (NRF) funded by the Korean government (MSIT) (no. NRF-2019M3E5D1A02069073).

## References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of things: challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[2] P. Shantharama, A. S. Thyagaturu, N. Karakoc, L. Ferrari, M. Reisslein, and A. Scaglione, "LayBack: SDN management of multi-access edge computing (MEC) for network access services and radio resource sharing," *IEEE Access*, vol. 6, Article ID 57545, 2018.

[3] P. K. Thiruvasagam, A. Chakraborty, and C. S. R. Murthy, "Resilient and latency-aware orchestration of network slices using multi-connectivity in MEC-enabled 5G networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2502–2514, 2021.

[4] 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.

[5] A. H. Sodhro, S. Pirbhulal, and V. H. C. de Albuquerque, "Artificial intelligence-driven mechanism for edge computing-based industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4235–4243, 2019.

[6] Y. Ye, S. Li, F. Liu, Y. Tang, and W. Hu, "EdgeFed: optimized federated learning based on edge computing," *IEEE Access*, vol. 8, Article ID 209191, 2020.

[7] L. U. Khan, M. Alsenwi, I. Yaqoob, M. Imran, Z. Han, and C. S. Hong, "Resource optimized federated learning-enabled cognitive Internet of things for smart industries," *IEEE Access*, vol. 8, Article ID 168854, 2020.

[8] X. Lu, Y. Liao, P. Lio, and P. Hui, "Privacy-preserving asynchronous federated learning mechanism for edge network computing," *IEEE Access*, vol. 8, Article ID 48970, 2020.

[9] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021.

[10] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han, and C. S. Hong, "Network slicing: recent advances, taxonomy, requirements, and open research challenges," *IEEE Access*, vol. 8, Article ID 36009, 2020.

[11] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1227–1240, 2020.

[12] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: a cloud-edge based framework," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 35–44, 2020.

[13] S. Math, P. Tam, and S. Kim, "Reliable federated learning systems based on intelligent resource sharing scheme for big data Internet of things," *IEEE Access*, vol. 9, Article ID 108091, 2021.

[14] L. Wang and D. Xu, "Resource allocation in downlink SWIPT-based cooperative NOMA systems," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 1, pp. 20–39, 2020.

[15] W. Saeed, Z. Ahmad, A. I. Jehangiri, N. Mohamed, A. I. Umar, and J. Ahmad, "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.

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

[17] J. Baek and G. Kaddoum, "Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1041–1056, 2021.

[18] S. A. Mohammed, A. R. Mohammed, D. Cote, and S. Shirmohammadi, "A machine-learning-based action recommender for network operation centers," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2702–2713, 2021, 2021.

[19] B. Li, R. Zhang, X. Tian, and Z. Zhu, "Multi-agent and cooperative deep reinforcement learning for scalable network automation in multi-domain SD-EONs," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4801–4813, 2021.

[20] B. Dab, N. Aitsaadi, and R. Langar, "Q-learning algorithm for joint computation offloading and resource allocation in edge cloud," in *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 45–52, Arlington, VA, USA, April 2019.

[21] S. Malektaji, A. Ebrahimzadeh, H. Elbiaze, R. H. Glitho, and S. Kianpisheh, "Deep reinforcement learning-based content migration for edge content delivery networks with vehicular nodes," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3415–3431, 2021.

[22] Z. Liu, X. Chen, Y. Chen, and Z. Li, "Deep reinforcement learning based dynamic resource allocation in 5G ultra-dense networks," in *Proceedings of the IEEE International Conference on Smart Internet of Things (SmartIoT)*, pp. 168–174, Tianjin, China, August 2019.

[23] F. Xu, F. Yang, S. Bao, and C. Zhao, "DQN inspired joint computing and caching resource allocation approach for software defined information-centric Internet of things network," *IEEE Access*, vol. 7, Article ID 61987, 2019.

[24] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.

[25] X. Wang, Y. Zhang, R. Shen, Y. Xu, and F.-C. Zheng, "DRL-based energy-efficient resource allocation frameworks for uplink NOMA systems," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7279–7294, 2020.

[26] G. M. S. Rahman, T. Dang, and M. Ahmed, "Deep reinforcement learning based computation offloading and resource allocation for low-latency fog radio access networks," *Intelligent and Converged Networks*, vol. 1, no. 3, pp. 243–257, 2020.

[27] E. Kim and S. Kim, "An efficient software defined data transmission scheme based on mobile edge computing for the massive IoT environment," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 2, pp. 974–987, 2018.

[28] R. Mu and X. Zeng, "A review of deep learning research," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 4, pp. 1738–1764, 2019.

[29] J. Xue and Y. An, "Joint task offloading and resource allocation for multi-task multi-server NOMA-MEC networks," *IEEE Access*, vol. 9, Article ID 16152, 2021.

[30] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 235–250, 2020.

[31] V. Jain and B. Kumar, "Optimal task offloading and resource allotment towards fog-cloud architecture," in *Proceedings of the 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 233–238, Noida, India, January 2021.

[32] P. Babarczi, "Resilient control plane design for virtual software defined networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2557–2569, 2021.

[33] C. Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in NOMA heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, Article ID 16001, 2020.

[34] Z. A. Arain, X. Qiu, L. Zhong et al., "Stochastic optimization of multipath TCP for energy minimization and network stability over heterogeneous wireless network," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 1, pp. 195–215, 2021.

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, and Z. Chen, "Tensorflow: large-scale machine learning on heterogeneous distributed systems," 2016, https://arxiv.org/abs/1603.04467.

[36] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds., Springer, Berlin, Germany, 2010.