Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.030984
Article

Tech Science Press

Adaptive Partial Task Offloading and Virtual Resource Placement in SDN/NFV-Based Network Softwarization

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

[1]Department of Software Convergence, Soonchunhyang University, Asan, 31538, Korea
[2]Department of Computer Software Engineering, Soonchunhyang University, Asan, 31538, Korea
*Corresponding Author: Seokhoon Kim. Email: seokhoon@sch.ac.kr
Received: 07 April 2022; Accepted: 27 June 2022

Abstract: Edge intelligence brings the deployment of applied deep learning (DL) models in edge computing systems to alleviate the core backbone network congestions. The setup of programmable software-defined networking (SDN) control and elastic virtual computing resources within network functions virtualization (NFV) are cooperative for enhancing the applicability of intelligent edge softwarization. To offer advancement for multi-dimensional model task offloading in edge networks with SDN/NFV-based control softwarization, this study proposes a DL mechanism to recommend the optimal edge node selection with primary features of congestion windows, link delays, and allocatable bandwidth capacities. Adaptive partial task offloading policy considered the DL-based recommendation to modify efficient virtual resource placement for minimizing the completion time and termination drop ratio. The optimization problem of resource placement is tackled by a deep reinforcement learning (DRL)-based policy following the Markov decision process (MDP). The agent observes the state spaces and applies value-maximized action of available computation resources and adjustable resource allocation steps. The reward formulation primarily considers task-required computing resources and action-applied allocation properties. With defined policies of resource determination, the orchestration procedure is configured within each virtual network function (VNF) descriptor using topology and orchestration specification for cloud applications (TOSCA) by specifying the allocated properties. The simulation for the control rule installation is conducted using Mininet and Ryu SDN controller. Average delay and task delivery/drop ratios are used as the key performance metrics.

Keywords: Deep learning; partial task offloading; software-defined networking; virtual machine; virtual network functions

1 Introduction

1.1 Motivation and Problem Statement

With future aspects of enabling network automation, the 3rd generation partnership project (3GPP), a global organization for standard telecommunications, released technical specifications of various use cases, which includes the procedures of gathering primary network data features, machine learning (ML) model training, and request-response interactions in the network data analytics function (NWDAF) [1]. In addition, European telecommunications standards institute (ETSI) contributes reference models and promising concepts in generic autonomic network architecture (GANA) and experiential networked intelligence (ENI) for enabling artificial intelligence (AI) algorithms, including ML/deep learning (DL) and self-organizing deployment policies [2,3]. With the preliminaries of superior testbeds and architectural systems, the decision-making and management entities of intelligent modules can be deployed within specified interfaces and network service demands which interact with operations support system (OSS)/business support system (BSS). Therefore, the confluence of network functions virtualization (NFV), software-defined networking (SDN), and edge computing allows an adaptive configuration of intelligent model policies on resource pooling adjustment, networking abstractions, and efficient slicing capabilities in multi-type Quality-of-Service (QoS) indicators with different criticality requirements [4–7].

Applying intelligent models in SDN-based architecture has been converged for multiple purposes, such as flow rules optimization, flow-aware classifications, estimation of QoS performances, and resource orchestration by enabling the interaction interfaces between classifiers and controllers [8,9]. However, in the era of the massive Internet of Things (IoT), the user equipment (UE) consists of low battery life and resource constraints for abstracting and training the intelligent collaborative learning model. Partial offloading policies can be considered for optimizing the completion time of local computation-intensive tasks [10]. With task sizes exceeding the local resource properties, the available local power and processing capacities have to consider executing a portion of the task, while another portion is offloaded to the optimal edge node. The combined local execution, offloading time, and edge computing time are primary elements to be critically forecasted and observed in order to alleviate the high possibility of task termination. Therefore, the optimal edge node selection, virtual resource placement, and task offloading scheduling are the major statement to tackle for further enhancement. To optimize this problem statement, partial offloading decisions and scheduling policy determination can be handled for energy-efficient and communication-efficient aspects by an optimization approach such as the convex technique [11,12]. However, deep reinforcement learning (DRL)-based optimization approaches are used for enabling intelligent resource allocation and offloading decisions by tackling the minimization of weighted sum completion time, consumed energy, and closed-loop resource control policies in systematic edge computing platforms [13–15].

1.2 Paper Contributions

In this paper, a working flow on deep Q-learning (DQL)-enhanced virtual resource orchestration (DQL-eVRO) policies are set on SDN/NFV-based architecture to observe the primary states, action-based orchestration, and rewards which consider the minimization of weighted sum task termination ratio. A deep neural network (DNN)-based classifier module (DNN-bCM) for delay-aware edge task computing in terms of congestion windows, link delays, and bandwidth capacities to recommend optimal edge nodes is proposed for labeling the efficiencies of available edge resources in the particular time slot. The confluence of DQL-eVRO and DNN-bCM responds to an enhancement for adaptive partial task offloading and virtual resource placement (APTOV) scheme in collaborative control softwarization, which raised one deployment scenario in learning model interactions. The simulation is conducted to indicate the performances of the proposed and reference schemes.

1.3 Paper Organization

The paper is organized as follows. Section 2 presents the system architecture of the proposed approach, including the architectural framework for the control softwarization, communication model, computation model, and primary components of the agent. Section 3 describes the algorithm flows and enhanced controller/orchestrator perspective, which is the main functional softwarization of the proposed APTOV approach in collaborative SDN/NFV-based control. Section 4 shows the simulation setup and result discussions. Section 5 presents the summary conclusion of the paper.

2 System Architecture

To estimate the optimal edge node, the DNN model can be formulated by feeding the network data collected by OpenFlow (OF) based device/resource abstractions with support modules. Within NFV-based architecture, edge servers (ES) in virtualized infrastructure manager (VIM) consist of virtualization layers that abstract the computing resources and isolate multi-pools based on different critical computing tasks. With well-classified offloading tasks, the orchestration of virtual resource placement for edge computing is considered by allocated virtual machine (VM) properties.

The equipped extra server capacities within the edge node, small base stations (SBS), are presented for assisting partial task offloading policies. Each local UE notifies the remaining energy and computing resources as a portion of state spaces for the proposed DQL-eVRO agent. In the partition phase, the existing available local capacity is a primary feature to consider before determining the edge offloading task size. In the system model, each task consists of primary variables from task profilers such as task sizes, required computing resources, and maximum available execution delay, which are denoted as a three-tuple (ρ, φ, τ), respectively [16–20]. The edge node resources are observed as state spaces for formulating the total sum of partial computing delay in local and edge entities.

2.1 Communication Model

Let $ES = \{1, \ldots k, \ldots K\}$ denote the set of NFV-based ES, which are equipped with a set of SBSs, denoted as $SBS = \{1, \ldots j, \ldots J\}$, respectively. The values of K and J are equivalent. A set of local UEs is denoted as $UE = \{1, \ldots l, \ldots L\}$, and each device l consists of resource limitation and independent computation-intensive tasks, which refers to the full local model in each learning time slot t, denoted as $\omega_{l(t)}$, that partitioned into i subtasks, denoted as $\omega_{l,i(t)}$. Let $SubT = \{1, \ldots i, \ldots I\}$ denote as the set of sequential subtask notation. With identified local computing capacities and sustainable energy, the availability of local computing is constrained, and the remaining subtasks are offloaded to selected $SBS\ j$ for computing in connected ES node k. The offloading delay, denoted as $T^{off(k)}_{l,i(t)}$, for transmitting the subtask $\omega_{l,i(t)}$ to $SBS - ES\ (j, k)$ is expressed as Eq. (1) by acknowledging the subtask size $\rho\left[\omega_{l,i(t)}\right]$ and uplink data rate $R_{l,j(t)}$ that is presented in Eq. (2). $bw_{l,j(t)}$, $p^{prop}_{l,j(t)}$, $c^{gain}_{l,j(t)}$, and N represents the allocated bandwidth, transmission propagation, channel gain, and Gaussian noise of each offloading subtask between UE and SBS at particular time slot t, respectively.

$$T^{off(k)}_{l,i(t)} = \rho\left[\omega_{l,i(t)}\right] / R_{l,j(t)} \tag{1}$$

$$R_{l,j(t)} = bw_{l,j(t)} \log_2\left(1 + \frac{p^{prop}_{l,j(t)} c^{gain}_{l,j(t)}}{N}\right) \tag{2}$$

2.2 Computation Model

In the partial offloading aspect, the local computing, offloading, and edge computing delays are critical for taking into consideration; however, the total number of subtask execution in different entities has to be acutely tackled. Eqs (3) and (4) present the primary formulation of local and edge computing delays, which are mainly based on the available resource in local UE i and allocated resources in edge node k for that particular subtask, denoted as $R_{l(t)}$ and $R_{l,i(t)}^{aloc(k)}$, respectively. A subtask execution delay in a single local and ES node k at time t is presented as $T_{l,i(t)}^{local}$ and $T_{l,i(t)}^{edge(k)}$, respectively. The task size of particular subtask i in local device l, denoted as $\varphi\left[\omega_{l,i(t)}\right]$, are considered for the computation delay. However, the sum of each subtask in various entities has to be formulated for the reward efficiencies.

$$T_{l,i(t)}^{local} = \varphi\left[\omega_{l,i(t)}\right]/R_{l(t)} \tag{3}$$

$$T_{l,i(t)}^{edge(k)} = \varphi\left[\omega_{l,i(t)}\right]/R_{l,i(t)}^{aloc(k)} \tag{4}$$

The full task computing delay at local UE l in time slot t, denoted as $T_{l(t)}^{total}$, is presented in Eq. (5) by weighted summation of each subtask with conditional statuses of local and edge offloading destination, denoted as $c_{l,i(t)}^{local}$ and $c_{l,i(t)}^{edge(k)}$, respectively. The conditional values are expressed in Eqs. (6) and (7), where the subtasks have to be offloaded to one entity, whether among edge nodes or local. $c_{l,i(t)}^{local} = 1$ if subtask i computes in local UE l; otherwise, $c_{l,i(t)}^{local} = 0$. In edge computing, subtask i requires computing in selected one edge node k. And in this aspect, multiple subtasks in each whole local task have to be scheduled, presented, and executed for offloading at both entities.

$$T_{l(t)}^{total} = \sum\nolimits_{i=1}^{I} c_{l,i(t)}^{local} T_{l,i(t)}^{local} + \sum\nolimits_{k=1}^{K} \sum\nolimits_{i=1}^{I} c_{l,i(t)}^{edge(k)} \left(T_{l,i(t)}^{off(k)} + T_{l,i(t)}^{edge(k)} \right) \tag{5}$$

$$c_{l,i(t)}^{local} \in \{0,1\} \tag{6}$$

$$\sum\nolimits_{k=1}^{K} c_{l,i(t)}^{edge(k)} = 1,\ \forall c_{l,i(t)}^{edge(k)} \in \{0,1\} \tag{7}$$

2.3 Problem Formulation

In this approach, the main objective of joint minimizing the partial offloading computation in a particular entire local task, presented in Eq. (8), is by selecting the optimal edge node k with adequate resource capacities and identifying the subtask batches for local computing based on the remaining local resources. The constraints to achieve this prime goal are presented. The total execution time of a particular task from local UE l in time slot t, $T_{l(t)}^{total}$, cannot exceed the maximum endurable termination delay, $\tau\left(\left[\omega_{l(t)}\right]\right)$, of the whole converging task profiling, which is presented in Eq. (9). The summation of every subtask size in a particular device l, that is scheduled for local or edge-k computation cannot exceed the remaining local $R_{l(t)}$ or edge-k $R_{l,i(t)}^{aloc(k)}$ resources, respectively. In Eqs. (10) and (11) present the local/edge resource constraints prerequisite. Allocated bandwidth capacities $bw_{l,j(t)}$ for each offloaded participant towards j SBS require balancing with the total bandwidth capacity bw_{j}^{total}, which is presented in Eq. (12). The influential features of edge nodes require to be gathered *via* edge-assisted information-rich module interacting with programmable controls and interfaces for optimizing the termination reduction and orchestrating the policies in adaptive agent softwarization [21–24].

$$\min\left(T_{l(t)}^{total}(.|k\ )\right) \tag{8}$$

$$T_{l(t)}^{total} \le \tau(\left[\omega_{l(t)}\right] \tag{9}$$

$$\sum\nolimits_{l=1}^{L} \sum\nolimits_{i=1}^{I} c_{l,i(t)}^{local} \varphi\left[\omega_{l,i(t)}\right] \le R_{l(t)} \tag{10}$$

$$\sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{i=1}^{I} c_{l,i(t)}^{edge(k)}\varphi\left[\omega_{l,i(t)}\right] \leq R_{l,i(t)}^{aloc(k)} \tag{11}$$

$$\sum_{l=1}^{L} bw_{l,j(t)} \leq bw_{j}^{total} \tag{12}$$

The edge node recommendation is used to enhance the solution efficiencies by approximating the delay-aware queuing and congestion states if local UE decides to offload several subtasks for computing in edge node k at time slot t; however, in heavy congestion hours, the classifier has a challenging procedure to identify the optimal edge for local UE according to the required experience-based capacities. Therefore, the state observations of heavy congestion interval and VM property modification in NFV-based architecture are jointly studied for adaptive closed-loop agent control in partial offloading policies [25–27].

2.4 Primary Components of Agent

With the setup environment following communication and computation models of the learning tasks, sets of states, actions, and rewards are required in this approach to be well-observed, applied as policy orchestration, and formulated as an achievement estimation in the proposed agent control, respectively. To acquire these requirements, the setup environment and three primary components of the agent, including epsilon (ε)-greedy exploration/exploitation balancing, q-learning execution, and online/target networks as an activate approximator, have to be well-interacted and cooperative correspondingly. Fig. 1 presents the overview of the interaction between each primary component in the DRL agent. In the setup environment, the support modules for observing the state of local partitioned tasks and edge capacity features are proposed and equipped for sampling a particular state $s(t)$ as an input for the agent. At the beginning phase, ε-greedy allows the exploration execution by randomly selecting a random action $a(t)$ without experience/value-based selection. For this phase, the online networks are still in the loss optimizing process, and the target networks are not yet in use. After applying $a(t)$ to the setup environment at a particular state $s(t)$, the reward $r(t)$ will be computed, and the setup environment will modify the associated attributes based on $a(t)$ values and transit to the next-state $s(t+1)$ depending on the transition probability $P(s)$ at that particular step. The action selection follows the long-term reward q-value approximation $q(t)$ and policy $\pi(t)$ at that time step.

$$y_{z}^{layer(i)} = f\left(w_{(i)} \cdot s(t)_{i} + b_{z}^{layer(i)}\right) \tag{13}$$

Batch experience at step t, denoted as $e(t)$, contains the notations of $\{s(t), a(t), r(t), s(t+1)\}$. The mini-batch divergence breaks the correlations of sequential order from the primary batch experience to improve the model efficiencies and follow the Markov decision process (MDP) statement for better independent decision-maker. $\{s(t), a(t), r(t)\}$ are used to optimize the online networks with the input features from $s(t)$ and the output from selected action $a(t)$. With defined output, the loss can be optimized in the DNN model. The output $y_{z}^{layer(i)}$ formulation in layer i of neuron z is presented in (13) by considering the activation method of combination between vectors of weight connectors $w_{(i)}$ and bias $b_{z}^{layer(i)}$ that interrelate to the state $s(t)_{i}$ in layer i. Fig. 2 presents the node of the general learning model, including the sum weight from numerous connectors, rectifier linear unit (ReLU) activation, and the output layer with sigmoid activation. To minimize the loss and update the weight optimally, gradient and Adam optimizer are used; moreover, the output layer is the sum activated weight and input state features [28–32].

3 Adaptive Partial Task Offloading and Virtual Resource Placement

In this section, the primary aspects of the proposed approach are described. The convergence of DNN-bCM and DQL-eVRO for an enhancement of APTOV scheme in collaborative SDN/NFV network softwarization is emphasized by illustrating the algorithm flows of each aspect and enhanced controller/ orchestrator perspective, which is the main functional control policy softwarization.

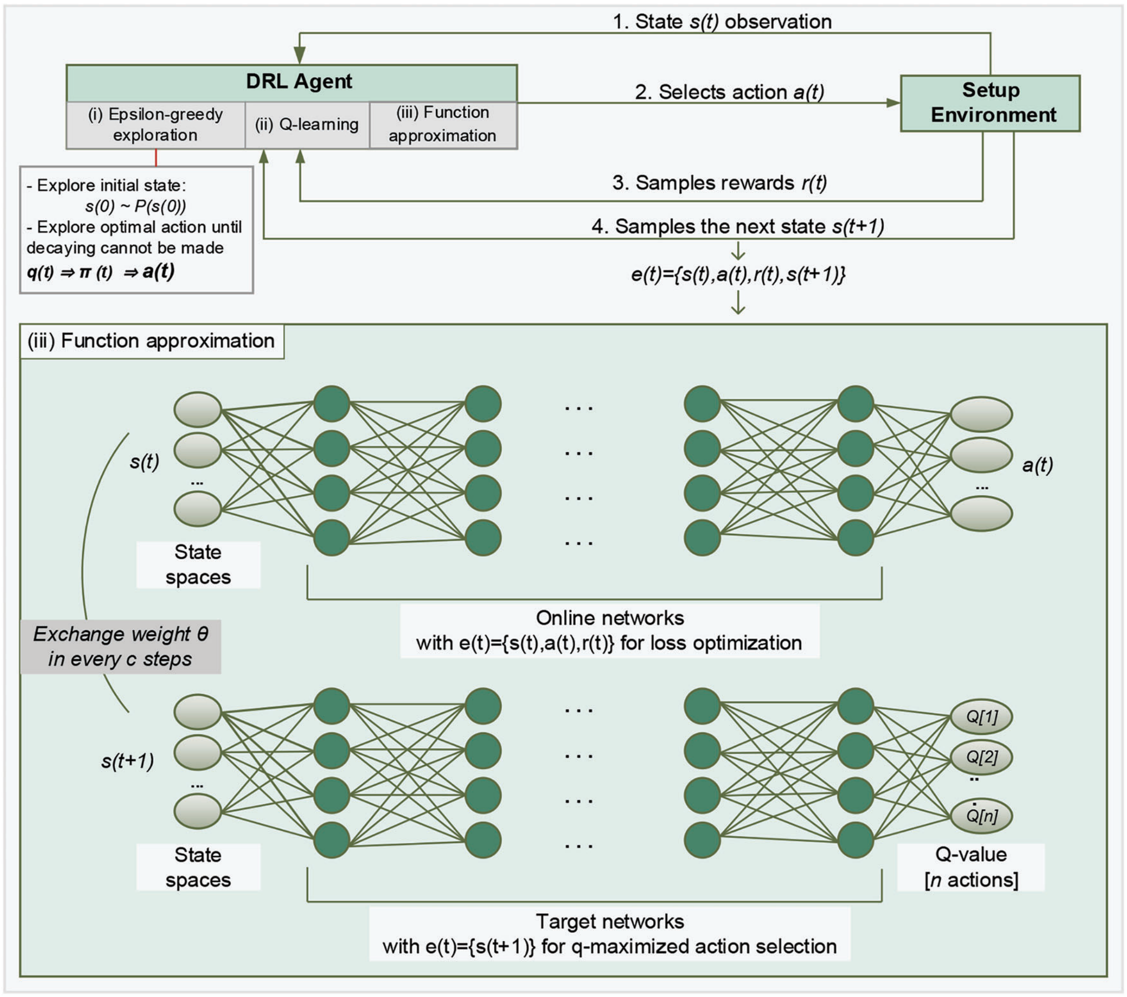
1. State s(t) observation
DRL Agent
(i) Epsilon-greedy exploration
(ii) Q-learning
(iii) Function approximation
2. Selects action a(t)
Setup Environment
3. Samples rewards r(t)
4. Samples the next state s(t+1)
- Explore initial state: s(0) ~ P(s(0))
- Explore optimal action until decaying cannot be made
q(t) ⇒ π (t) ⇒ a(t)
e(t)={s(t),a(t),r(t),s(t+1)}
(iii) Function approximation
s(t)
State spaces
a(t)
Online networks with e(t)={s(t),a(t),r(t)} for loss optimization
Exchange weight θ in every c steps
s(t+1)
State spaces
Q[1]
Q[2]
Q[n]
Q-value [n actions]
Target networks with e(t)={s(t+1)} for q-maximized action selection

Figure 1: Interaction of primary components in DRL agent

3.1 A Deep Neural Network-Based Classifier Module

In this classifier module, the primary phases to indicate the class labels of prioritization and policy-configured output for optimizing ES selection are shown. Fig. 3 illustrates the interactions between the five main stages in the control entity, including the collection, data adjustment, training/validation/testing, and recommendation phases, with the data plane. With OF-based protocol interfaces, the data plane and control plane are communicated with capabilities of device/resource gathering and programmable rule installations. The support modules are utilized in this system to observe the UE and ES information. In the collection phase, the features are accumulated for abstracting as a batch of the input dataset. The data adjustment phase carries on the procedure by detecting and eliminating the inadequate or dummy variables. With sufficient adjustment, the training labels for the learning model are obtained with specific indications of partitioned task criticality levels.

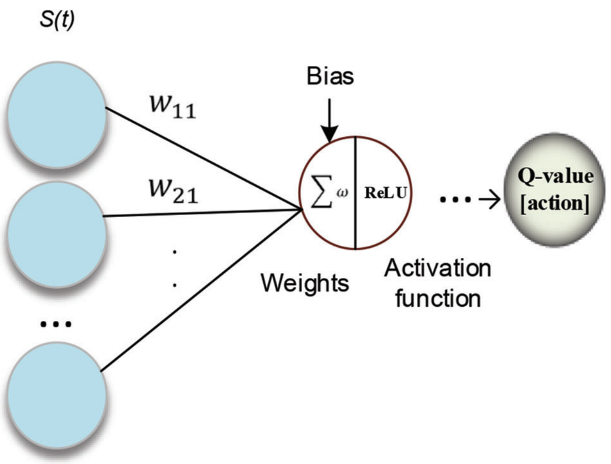

Figure 2: Learning model in a particular node

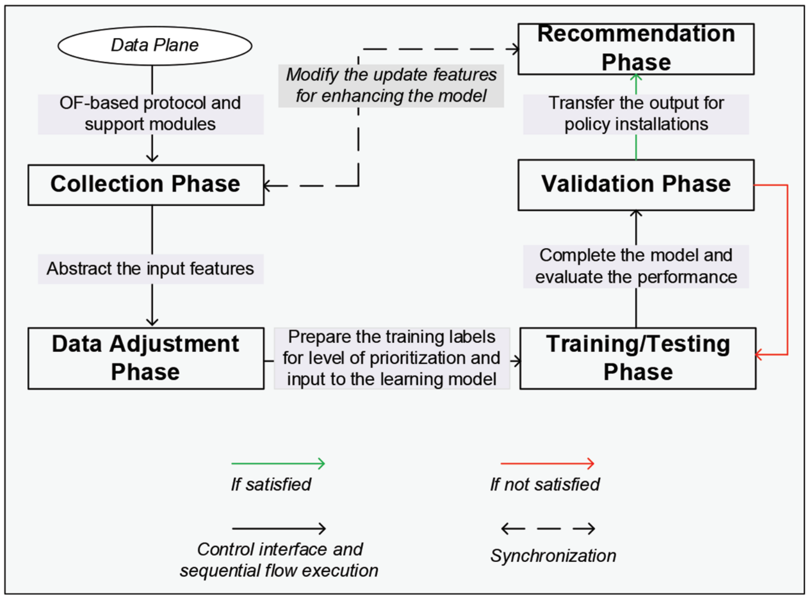
Data Plane
OF-based protocol and support modules
Collection Phase
Abstract the input features
Data Adjustment Phase
Prepare the training labels for level of prioritization and input to the learning model
Training/Testing Phase
Complete the model and evaluate the performance
Validation Phase
Transfer the output for policy installations
Recommendation Phase
Modify the update features for enhancing the model
If satisfied
If not satisfied
Control interface and sequential flow execution
Synchronization

Figure 3: Interactions between the data and control planes with flow connectivity of the five primary phases: collection, data adjustment, training/testing, validation, and recommendation phase

Algorithm 1: Procedure flow of DNN-based learning model for classifier modules and recommendation orchestration

```
Require: data features $x_m$ and labels $y$ from data adjustment phase $X = \{x_1, x_2, \ldots x_m,\ y\}$
Ensure: With corresponding input features $\{x_1, x_2, \ldots x_m\}$, the learning model requires classifying
         the label $y$ as the recommended ES for offloading.
1:    import the modules as high-level operations and prediction
2:    declare the model container
3:    read the collected data as data
```

(Continued)

Algorithm 1 (continued)

```
4:      $X = list$(data)
5:      define features of $x$ and $y$
6:      define shape of the dataset by float(features.shape [0])
7:      split the training, validation, and testing set
8:      define regularization strength by learning rate $\alpha$
9:      **with** namescope of weights:
10:         random the variables of $w_{(i)}$
11:      **with** namescope of biases:
12:         random the variables of $b_z^{layer(i)}$
13:      **with** namescope of layer $i$:
14:         add neural network of matrix product of $\left[layer(i),\ w_{(i)}\right],\ b_z^{layer(i)}]$
15:      declare cumulative sum of regularized weight container
16:      **for** each weight $w_{(i)}$ **do**
17:         apply regularization on $w_{(i)}$
18:         execute the cumulative sum and append to the container
19:      **formulate** the loss function + regularized values
20:      apply gradient descent optimizer with $\alpha$ to minimize the loss
21:      **save** the model and **run** with modification
```

The priority determination is assisted by the maximum endurable termination delay $\tau(\left[\omega_{l(t)}\right])$. The dataset with adequate features and labels is used as an input dataset for the training and testing phases. When the process of learning model is complete, the validation phase is responsible for evaluating the performance precision and accuracy. If the validation returns a satisfying performance, the model output will be transferred for installing the policy and flow configurations in the recommendation phase; otherwise, the training/testing phase will be re-executed for further enhancement. The synchronization between recommendation and collection phases indicates features to be updated and improve experience-based efficiencies of the model in long-term aspects. Algorithm 1 shows the execution procedure of DNN-bCM by following the softwarized deployment aspect. The procedure covers the training/testing phase after adjusting the data by jointly presenting the sequences of regularization process, loss, gradient descent optimizer, and model saving for further modification. The optimization towards optimal ES is evaluated critically in the validation phase. The modification of update features allows the data collection to enhance for further recommendation and define the main criteria to tackle in various state possibilities within multiple congestion statuses.

3.2 Deep Q-Learning-Enhanced Virtual Resource Orchestration

To enhance the elasticity and adaptability of the proposed softwarization framework, the recommendation of an optimal edge node is converged with the DQL-enhanced VM allocation for advancing adaptive partial task computation. This section delivers the primary aspect of the allocation statement by describing the procedure of agent interaction with the proposed states, actions, and reward indication. The implicit mechanisms for agent controller/orchestrator are configured for long-term control.

In the initialization and setup environment phases, the hyperparameters and elements of the DQL-based model have to be well designed and adjusted in order to emerge with the state conditions. The elements of DQL-based approaches, q-value function, DNN online/target approximation, and experience replay for training an optimal policy of the proposed scheme are presented. To adaptively control the communication and computation resources for partial task offloading, the state observation in the proposed environment is significant to gather for the proposed agent, including the resource statuses, task information, and communication values. The collective state observations in t step are presented in (14) as a set of s_t, which will input for online networks to approximate the q-value of action. The actions in this scheme consist of the efficiency of DNN-bCM decision-maker $c_{l,i(t)}^{edge(k)}$ and $R_{l,i(t)}^{aloc(k)}$ configuration from the transmitted local models to the edge node.

$$s_t = \left\{\rho\left[\omega_{l,i(t)}\right],\ \varphi\left[\omega_{l,i(t)}\right], R_{l,j(t)},\ R_{l(t)},\ bw_{l,j(t)}, p_{l,j(t)}^{prop}, c_{l,j(t)}^{gain}\right\} \tag{14}$$

With ε-greedy strategy, the exploration phase randomly selects an action a_t to re-modify the decision and allocation capacity $\left\{c_{l,i(t)}^{edge(k)}, R_{l,i(t)}^{aloc(k)}\right\}$ for training and capturing the alternative efficiency level in the early phase. In the exploitation phase, the scheme formulates the action selection with maximum q-value output. After the action is applied to the setup environment, s_{t+1} and r_t are obtained by state transition and valuation of joint minimizing sufficiency from the partial offloading computation in a particular full local task reward formulation as $T_{l(t)}^{total}(.|k\ )$. The latency of the edge computation model is formulated by the required CPU cycles for executing the tasks over the allocated server capacities. In this scheme, the required resources are the summation of computational costs from requested VNF in a particular rendered VNF forwarding graph (VNFFG) with the offloaded partial task requirement. The existing allocation of VNF and the computational costs of particular VNF in core control are considered. s_t and a_t feed into the online networks for state input and q-value of actions denoted as $Q(a_t)$. However, to minimize the loss and optimize the model parameters θ_t, the target q-value has to be formulated with an optimal reward function. θ_t' denotes the next-step model parameters. The target networks formulate the target q-value, denoted as Y_t^q, in DQL-based agent by splitting the procedure of action selections and q-value calculation as (15) in order to handle the over-optimistic value estimation. Target θ_t is iteratively optimized and synchronized with the online networks every 1000 steps. γ represents the discount factor, which determines the weight of reward certainty in the future steps.

$$Y_t^q = r_t\left(.|T_{l(t)}^{total}\right) + \ \gamma Q\left(s_{t+1}, \underset{a}{argmax}\, Q(s_{t+1}, a; \theta_t); \theta_t'\right) \tag{15}$$

However, the DQL-based model consisted of unstable training episodes with insufficient reward accumulation. In every 1000 steps, the proposed scheme formulates the collected batches for future optimization. The proposed controller and orchestrator leverage the resources with the weight-assisted offloading task. The virtual deployment unit (VDU) capabilities and properties are parameterized for configuring each VNF descriptor based on the modified weights of the network approximation. The discrete state priorities of the collected experience batch are classified for the prior adjustment of computational resource utilization. VNFFG descriptor includes the forwarding path of defined properties, policies, and VNF forwarders with connection point/virtual link capabilities for rendering into service chains. The predetermined resources in the edge server and each VNF placement are updated for intermediate reward calculation. Therefore, each virtual resource pool is defined as specific slicing with clustered model service types. The local model computation efficiency rate in each edge server is improved by instruction sets of each clustered traffic flow and virtualization extraction. The model interactivity between proposed action spaces and orchestrator is considered to express the triggering chains on gathering virtual resource pools in order to serve the partial model task communication and computation based on specific congestion states. The collection of functions that map accordingly to the virtual blocks of resources in the perspective of model actions is tackled for long-term end-to-end sufficiency. Therefore, the execution of partial task offloading feasible accesses an optimal edge server with adequate resource pools.

4 Results and Discussions

To efficiently simulate the APTOV approach, connectivity between agent output and SDN/NFV-based environment has to consider; however, with constraints of virtual interfaces, the output-configured relation is applied between DQL-based agent and mininet/mini-nfv/Ryu orchestration topology [7,23]. With partial task creation and offloading experiment, the consideration of task amounts, number of ES, virtual pool capacities, transmission bandwidth, server resources, local resources, and partial task sizes are included to evaluate the performance and proposed environment setup. Within 100 full local tasks, the partial division with the size of 450 KB is executed. In this environment, the number of ES is 5. The transmission bandwidth and server resources are 5 MHz and 15 GHz, respectively [24,33]. The transmission power is in the range of 40 mW to 100 mW. The simulation of computation and communication models for $T_{l,i(t)}^{local}$, $T_{l,i(t)}^{off(k)}$, and $T_{l,i(t)}^{edge(k)}$ was conducted for assisting the agent experiments.

With 310 slot times, the agent applied four different learning rate values for best fitting the conditional environment. The cumulative negative/positive rewards in each episode of OpenAI-Gym-based DQL agent are obtained for evaluating the agent execution. The agent used the initialization function to gather vectors of state possibilities and the approximation function to generate the exploration. The allocation function optimizes the percentage of resource placement. In each episode, the intermediate rewards have been calculated numerous times and accumulated to illustrate the model efficiencies and enhancement. With the output of the proposed APTOV agent, the controller/orchestrator configuration is made to show the QoS indicators. The outputs are configured into the descriptors corresponding to the $c_{l,i(t)}^{edge(k)}$ and $R_{l,i(t)}^{aloc(k)}$ action values.

With partial task offloading decisions and resource determination, the flow rule installation and life-cycle orchestration are made within the controller and create/update/delete functions in the VNF manager. The reference schemes are defined by indicating the differences between various policies for offloading and resource adjustment in terms of experience-based, resource-maximized, and single DNN-based approaches, which are denoted as EB-A, RM-A, and DNNB-A, respectively. The primary features of congestion windows, link delays, and allocatable bandwidth capacities in the dataset are gathered by simulation software, which is collected at each ES corresponding with each UE. The data was inputted into the DL mechanism to extract the features and optimize the model parameters. The train, validation, and test sizes are adjusted to 70%, 15%, and 15%, respectively. The loss and accuracy of model training/inference are evaluated before configuring the class labels in the DQL-eVRO. By using TensorFlow/Keras platform, the overall accuracy of the final learning model reached 99.86%. The experience-based approach offloaded the partial tasks based on the latency-maximized experience within the batch container comparison, which adjusts the resources corresponding to experience replay. The resource-maximized approach considered the resource allocation policies, which supervised placing the adjustable available resources at every congestion condition. For single DNN-based approaches, the resource features and local task information are gathered; however, the edge recommendation is fixed, and the VM capacities have challenging functionality to adaptively place and map in high congestion states.

Fig. 4 illustrates the graphical delay comparison between APTOV and reference approaches. In terms of delay, the proposed approach considered the congestion state by deploying the setup environment and proposed agent with the capability of configuring allocation actions to orchestrate the communication resource in peak-hour intervals and accelerate the offloading process. In this aspect, the delay-critical evaluation determined the partial offloading success rate without fully taking risks of high termination drops. Within SDN/NFV-based softwarization architecture, the elastic computing resource mapping and programmable rule installation enhance the optimal edge selection for resource-efficient and latency-efficient objectives. APTOV achieved an average delay of 9.4612 ms within 310 sconds experiment, which is 95.6222, 81.9085, and 30.2743 ms lower than EB-A, RM-A, and DNNB-A, respectively.

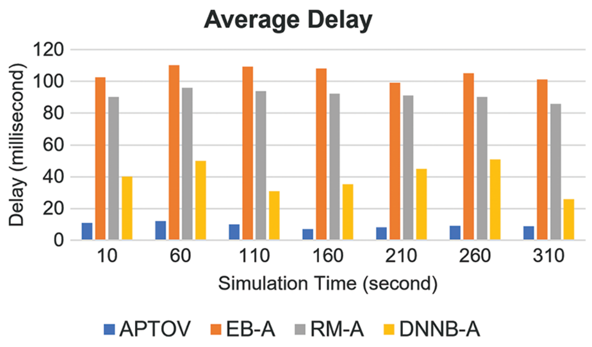

Figure 4: Comparison of average delay

Fig. 5 presents the partial task delivery ratio of APTOV and reference approaches within 310 s of simulation. This performance metric covers the successful rates of subtask $\omega_{l,i(t)}$ uploading towards $SBS - ES\ (j,k)$. By optimal offloading edge recommendation, the communication model with sufficient allocated bandwidth, propagation power, and channel gain could perform the distribution effectively. APTOV used a DQL-enhanced controller and orchestrator to primarily assist the DNN-bCM decisions and improve the VM capacity placement in heterogeneous multi-dimensional model updates. With sufficient reward-maximized action selection, APTOV delivers the offloading task efficiently through OF-based interfaces and support modules for long-term agent control. With a global view of data infrastructure topology, partial offloading decisions can be determined with various options to maximize the end-to-end completion time. APTOV achieved an ending delivery ratio of 99.98%, which is 1.7862%, 0.98%, and 0.38% higher than EB-A, RM-A, and DNNB-A, respectively. Fig. 6 shows the graphical comparison of the drop ratio between proposed and reference schemes. The high drop ratio allowed the full task termination and insufficient uses of network and computation resources. With APTOV control and orchestration in SDN/NFV-based approach, the drop ratio is alleviated for future massive intelligent edge application scenarios, where partial task offloading is deployed, particularly for model training/ inference in edge intelligence scenarios.

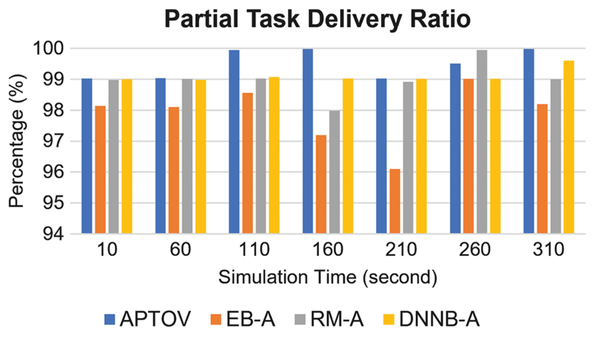

Figure 5: Comparison of partial task delivery ratio

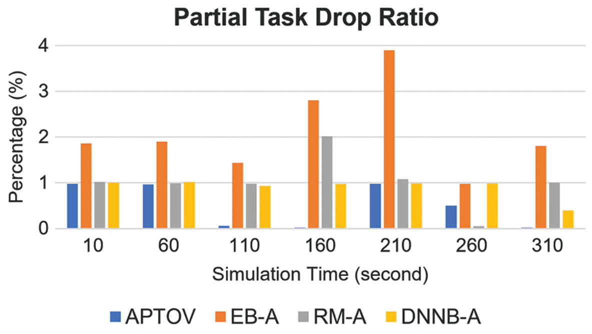

Figure 6: Comparison of partial task drop ratio

5 Conclusion and Future Work

In this paper, an adaptive decision with efficient virtual resource allocation for partial task offloading is presented by collecting and adjusting the data within leveraged edge computing and SDN/NFV-based network softwarization. The system model for training/testing the optimal edge recommendation in DNN-bCM considered the interaction between local devices, ES, and SDN/NFV-based control with proposed DQL-eVRO to tackle the alternative offloading destination with a modified flow rule setup. The confluence between DNN-bCM and DQL-eVRO introduced the APTOV approaches for adaptive task offloading and efficient resource mapping, followed by the reward formulation by minimization models of partial task computation delay in local, offloading delay towards selected edge node, and computation delay in recommended ES. The proposed simulation correspondingly configured and orchestrated the agent actions to enhance the QoS performances for time-sensitive networking applications, particularly in high congestion and peak-hour conditions. DQL-based approach aimed to offer the self-managing closed-loop control in network virtualization and softwarization environment.

In future studies, the virtual link connectivity for mininet/mini-nfv/Ryu and DQL-based agents will be further extended. Moreover, the scheduling and partition procedure with task profiler information will be further studied by using double deep-q-networks and artificial neural networks to meet the ultra-reliable low-latency QoS requirements for mission-critical applications.

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] T. K. Le, U. Salim and F. Kaltenberger, "An overview of physical layer design for ultra-reliable low-latency communications in 3GPP releases 15, 16, and 17," *IEEE Access*, vol. 9, pp. 433–444, 2021.

[2] R. Chanparadza, T. B. Meriem, B. Radier, S. Szott and M. Wodczak, "SDN enablers in the ETSI AFI GANA reference model for autonomic management & control (emerging standard), and virtualization impact," in *2013 IEEE Globecom Workshops (GC Wkshps)*, Atlanta, GA, USA, pp. 818–823, 2013.

[3] Y. Wang, R. Forbes, U. Elzur, J. Strassner, A. Gamelas *et al.,* "From design to practice: ETSI ENI reference architecture and instantiation for network management and orchestration using artificial intelligence," *IEEE Communications Standards Magazine*, vol. 4, no. 3, pp. 38–45, 2020.

[4] X. Yin, B. Cheng, M. Wang and J. Chen, “Availability-aware service function chain placement in mobile edge computing,” in *2020 IEEE World Congress on Services (SERVICES)*, Beijing, China, pp. 69–74, 2020.

[5] T. Rausch and S. Dustdar, “Edge Intelligence: The convergence of humans, things, and AI,” in *2019 IEEE Int. Conf. on Cloud Engineering (IC2E)*, Prague, Czech Republic, pp. 86–96, 2019.

[6] W. Zhuang, Q. Ye, F. Lyu, N. Cheng and J. Ren, “SDN/NFV-empowered future IoV with enhanced communication, computing, and caching,” *Proceedings of the IEEE*, vol. 108, no. 2, pp. 274–291, 2020.

[7] P. Tam, S. Math, C. Nam and S. Kim, “Adaptive resource optimized edge federated learning in real-time image sensing classifications,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 10929–10940, 2021.

[8] G. Zhao, H. Xu, Y. Zhao, C. Qiao and L. Huang, “Offloading tasks with dependency and service caching in mobile edge computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 11, pp. 2777–2792, 2021.

[9] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez *et al.,* “A survey on machine learning techniques for routing optimization in SDN,” *IEEE Access*, vol. 9, pp. 104582–104611, 2021.

[10] Z. Kuang, L. Li, J. Gao, L. Zhao and A. Liu, “Partial offloading scheduling and power allocation for mobile edge computing systems,” *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, 2019.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K: Cambridge Univ. Press, 2004. [Online]. Available at: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf.

[12] 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.

[13] K. Haseeb, I. Ahmad, I. I. Awan, J. Lloret and I. Bosch, “A machine learning SDN-enabled big data model for IoMT systems,” *Electronics*, vol. 10, no. 18, pp. 2228, 2021.

[14] H. Park and Y. Lim, “Deep reinforcement learning based resource allocation with radio remote head grouping and vehicle clustering in 5G vehicular networks,” *Electronics*, vol. 10, no. 23, pp. 3015, 2021.

[15] Y. Y. Munaye, R. -T. Juang, H. -P. Lin, G. B. Tarekegn and D.-B. Lin, “Deep reinforcement learning based resource management in UAV-assisted IoT networks,” *Applied Science*, vol. 11, no. 5, pp. 2163, 2021.

[16] H. Li, F. Fang and Z. Ding, “DRL-assisted resource allocation for NOMA-MEC offloading with hybrid SIC,” *Entropy*, vol. 23, no. 5, pp. 613, 2021.

[17] X. Han, X. Meng, Z. Yu and D. Zhai, “A dynamic adjustment method of service function chain resource configuration,” *KSII Transactions on Internet and Information Systems*, vol. 15, no. 8, pp. 2783–2804, 2021.

[18] G. Zhao, H. Xu, Y. Zhao, C. Qiao and L. Huang, “Offloading tasks with dependency and service caching in mobile edge computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 11, pp. 2777–2792, 2021.

[19] Z. Jin, C. Zhang, G. Zhao, Y. Jin and L. Zhang, “A context-aware task offloading scheme in collaborative vehicular edge computing systems,” *KSII Transactions on Internet and Information Systems*, vol. 15, no. 2, pp. 383–403, 2021.

[20] I. A. Elgendy, W. -Z. Zhang, Y. Zeng, H. He, Y. -C. Tian *et al.,* “Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile iot networks,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2410–2422, 2020.

[21] O. Adamuz-Hinojosa, J. Ordonez-Lucena, P. Ameigeiras, J. J. Ramos-Munoz, D. Lopez *et al.,* “Automated network service scaling in NFV: Concepts, mechanisms and scaling workflow,” *IEEE Communications Magazine*, vol. 56, no. 7, pp. 162–169, 2018.

[22] M. Abu-Alhaija, N. M. Turab and A. Hamza, “Extensive study of cloud computing technologies, threats and solutions prospective,” *Computer Systems Science and Engineering*, vol. 41, no. 1, pp. 225–240, 2022.

[23] P. Tam, S. Math, A. Lee and S. Kim, “Multi-agent deep Q-networks for efficient edge federated learning communications in software-defined IoT,” *Computers, Materials & Continua*, vol. 71, no. 2, pp. 3319–3335, 2022.

[24] K. Xiao, Z. Gao, C. Yao, Q. Wang, Z. Mo *et al.,* “Task offloading and resources allocation based on fairness in edge computing,” in *Proc. IEEE Wireless Communications and Networking Conf. (WCNC)*, Marrakesh, Morocco, pp. 1–6, 2019.

[25] R. Amiri, M. A. Almasi, J. G. Andrews and H. Mehrpouyan, “Reinforcement learning for self organization and power control of two-tier heterogeneous networks,” *IEEE Transactions on Wireless Communications*, vol. 18, no. 8, pp. 3933–3947, 2019.

[26] 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.

[27] J. Pei, P. Hong, K. Xue and D. Li, “Resource aware routing for service function chains in SDN and NFV-enabled network,” *IEEE Transactions on Services Computing*, vol. 14, no. 4, pp. 985–997, 2021.

[28] Z. Lv and W. Xiu, “Interaction of edge-cloud computing based on SDN and NFV for next generation IoT,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5706–5712, 2020.

[29] I. Kalphana and T. Kesavamurthy, “Convolutional neural network auto encoder channel estimation algorithm in MIMO-OFDM system,” *Computer Systems Science and Engineering*, vol. 41, no. 1, pp. 171–185, 2022.

[30] C. Pham, D. T. Nguyen, N. H. Tran, K. K. Nguyen and M. Cheriet, “Optimized IoT service chain implementation in edge cloud platform: A deep learning framework,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 538–551, 2021.

[31] K. Praveen kumar and P. Sivanesan, “Cost optimized switching of routing protocol scheme for IoT applications,” *Computer Systems Science and Engineering*, vol. 41, no. 1, pp. 67–82, 2022.

[32] C. She, C. Sun, Z. Gu, Y. Li, C. Yang *et al.,* “A tutorial on ultrareliable and low-latency communications in 6G: Integrating domain knowledge into deep learning,” *Proceedings of the IEEE*, vol. 109, no. 3, pp. 204–246, 2021.

[33] X. Wu, “A dynamic QoS adjustment enabled and load-balancing-aware service composition method for multiple requests,” *KSII Transactions on Internet and Information Systems*, vol. 15, no. 3, pp. 891–910, 2021.