# DQN Aided Edge Computing in Satellite-Terrestrial Network

Fangmin Xu(✉), Fan Yang, Chao Qiu, Chenglin Zhao, and Bin Li

Beijing University of Posts and Telecommunications, Beijing, China
xufm@bupt.edu.cn

**Abstract.** In order to support a mass of current satellite applications, it becomes a trend to integrate satellite networks with terrestrial networks, called satellite-terrestrial networks. However, traditional network protocols cannot adapt to the dynamic and complex satellite-terrestrial network. Moreover, the computing and communication capabilities of some satellites cannot meet the requirements of supporting various applications. As a result, the paper proposes an edge computing based software-defined satellite-terrestrial network architecture, which can manage network flexibly by logically centralizing network intelligence and control. Furthermore, a networking and edge computing scheme is proposed by formulating a jointly optimization problem, which is solved by using novel deep Q-learning approach. Simulation results show the effectiveness of the proposed scheme.

**Keywords:** Satellite-terrestrial networks · Software defined networking (SDN) · Networking · Edge computing · Deep-Q-learning

## 1 Introduction

Satellite network is developing fast recently and has been widely used in many fields, such as emergency assistance, navigation, remote sensing and so on [1]. Thus, variety of satellite applications are developed. However, the capability of satellites is not high enough to support the explosion of current satellite applications. So there is an emerging trend to integrate satellite networks with terrestrial networks, called satellite-terrestrial networks [2].

Traditional satellite network protocols have poor mobility, high costs and complexity, and it cannot adapt to the dynamic changing satellite-terrestrial network. Satellite-terrestrial needs a flexible and intelligent network architecture and protocols. Thus, there is a growing trend to use software-defined network (SDN) to reconstruct the satellite-terrestrial networks. SDN decoupled control and data plane, logically centralized network intelligence and state, and

Supported by the Key Program of the National Natural Science Foundation of China (61431008).


M. Jia et al. (Eds.): WiSATS 2019, LNICST 280, pp. 117–127, 2019.
https://doi.org/10.1007/978-3-030-19153-5_11

abstracted the underlying network infrastructure [3]. Feng *et al.* in [4] proposed an architecture for heterogeneous satellite-terrestrial networks, they use SDN and network virtualization to improve the flexibility of networks. Li *et al.* in [5] proposed a multi-strategy flow table management scheme, called *SAT-FLOW*. Boero in [6] constructed a SDN-based terrestrial-satellite network architecture to minimize the mean time to deliver the data of a new traffic flow from source to destination including the time required to transfer SDN control actions. Liu in [7] proposed an optimal controller and gateway placement algorithm in SDN enabled 5G-satellite integrated network which increased the network reliability and decreased the latency. It is obvious that, by abstracting and allocating networking resources dynamically, satellite-terrestrial networks can get better performance such as reliability, latency, flexibility and so on.

Moreover, traditional satellite-terrestrial network apply cloud computing to assist the satellite in computing complex tasks. For example, Huang in [8] proposed a novel layered architecture for the space-based cloud infrastructure to aid space network. However, due to the long distance and huge amount of data to be delivered, it is not economical and feasible to offload computation tasks to the cloud center. So there is an increasing trend to use edge computing to aid satellite-terrestrial network. Edge computing is a new paradigm of shifting computation resources to the network edge to improve network service performance. This brings higher network resource utilization and computation performance. Specially, our contributions are:

- Proposed an edge computing based software-defined satellite-terrestrial network architecture and formulate a jointly resource allocation problem based on it.
- Utilize deep Q-learning approach to solve the jointly resource allocation problem.

The rests of the paper is organized as follow. In Sect. 2, system description and model are presented. The joint optimization problem of networking and edge computing is formulated as a Q-learning process and solved in Sect. 3. In Sect. 4, we present the simulation results. Finally, conclusions presented in Sect. 5.

## 2 System Model

In this section, we first present the proposed architecture of edge computing based software-defined satellite-terrestrial network. Then system model is given, including network model, communication model and computing model.

### 2.1 Proposed Framework of Software-Defined Satellite-Terrestrial Network

This paper take a joint consideration of both networking and computing in software-defined satellite-terrestrial network architecture. The architecture includes data plane, control plane and application plane, as shown in Fig. 1

In the data plane, there are two types of infrastructures to provide network resources and edge computing resources. Networking resource infrastructure includes LEOs and MEOs. And computing resource infrastructure consists of MEC servers which provide computation resources for the network users. In the control plane, the control intelligence is abstracted in GEO satellites and land-based controllers. They are responsible for centralized management of the LEO/MEO satellites and MEC servers. Controllers receive the requests from devices in data plane and consider both the networking resources and MEC server status to make a decision, i.e. which LEO/MEO should server the user, and which MEC server should assist the LEO/MEO on processing raw data. In the application plane, a number of applications are provided, including remote sensing, navigation, communicating and monitoring.

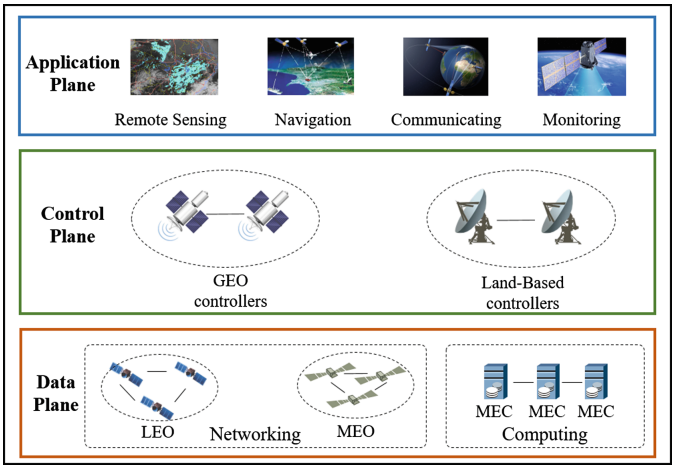


**Fig. 1.** The framework of Software-Defined Satellite-Terrestrial network.

The workflow of the architecture includes several steps. First, a user sends a request to the GEO/land-based controller, i.e. requests to identify an objects (such as vehicles or vessels) in a video/image of a certain area. Controller runs the algorithm and returns decision to the user, the decision includes which LEO satellite could the user connect to get satellite networking resource and which MEC server should aided the LEO satellite with processing raw data by complex image/video identification algorithms. Then, user connects to the selected LEO according to controller decision. LEO retrieves for the content and pass the raw data to the selected MEC server. If the MEC server is busy, then the tasks need to queue up. Till the MEC server finishes the previous tasks. MEC server begin to execute the tasks and return the result to the LEO, and then the content will be returned to the user. To visualize the description, we give the workflow diagram in Fig. 2.

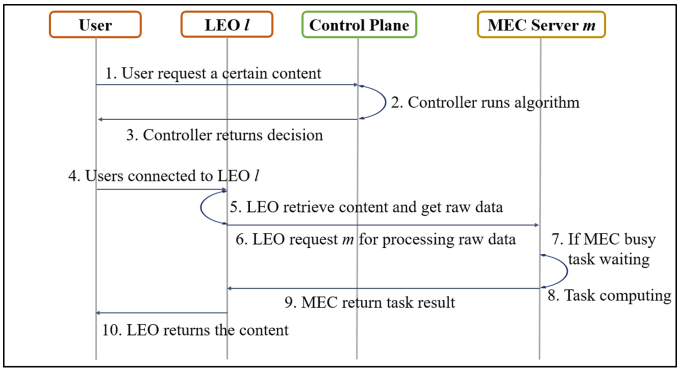


**Fig. 2.** Workflow in the proposed architecture.

### 2.2 Network Model

There are two resource pools in data layer which includes LEO/MEO satellites, and MEC servers, which are managed by the controllers in the control plane. Let $\mathcal{L} = \{1, ..., L\}$, $\mathcal{M} = \{1, ..., M\}$, $\mathcal{U} = \{1, ..., U\}$ denote the set of LEOs/MEOs, MEC servers, and users.

### 2.3 Communication Model

We assume that the wireless channels between users and LEO/MEOs are realistic time-varying, it can be modeled as finite-state Markov channels (FSMCs). In this model, channel quality can be expressed by the received SNR $h_u^l$ of a user $u$ from LEO/MEO $l$. We split the range of $h_u^l$ into $L'$ discrete intervals, each interval is a state of Markov chain, i.e., $\mathcal{H} = \{\mathcal{H}_0, \mathcal{H}_1, ..., \mathcal{H}_{L'-1}\}$. The received SNR $h_u^l$ at time $t$ can be represented by $\mathbf{h}_u^l(t)$, where $t \in \{0, 1, 2, ..., T-1\}$. Based on a certain transition probability, the received SNR $\mathbf{h}_u^l(t)$ changes from a state to another, and the transition probability from one state $\vartheta_{\bar{s}}$ to another state $\varphi_{\bar{s}}$ at time slot $t$ could be denoted as $\gamma_{\vartheta_{\bar{s}}\varphi_{\bar{s}}}(t)$. Then, the transition probability matrix $\Upsilon_u^l(t)$ could be represented as:

$$\Upsilon_u^l(t) = [\gamma_{\vartheta_{\bar{s}}\varphi_{\bar{s}}}(t)]_{L' \times L'}, \tag{1}$$

where $\gamma_{\vartheta_{\bar{s}}\varphi_{\bar{s}}}(t) = Pr(\mathbf{h}_u^l(t+1) = \varphi_{\bar{s}} | \mathbf{h}_u^l(t) = \vartheta_{\bar{s}})$, and $\varphi_{\bar{s}}, \vartheta_{\bar{s}} \in \mathcal{H}$.

We assume that the available spectrum bandwidth of LEO/MEO $l$ is $B^l$ Hz, among which $B_u^l$ Hz is allocated to user $u$. The available backhaul capacity of LEO/MEO $l$ is $Z^l$ bps. The spectrum efficiency of user $u$ is $\nu_u^l(t)$ at time $t$. The communication rate of user $u$ associated with LEO/MEO $l$ is:

$$ComR_u^l(t) = a_u^l(t)B_u^l(t)\nu_u^l(t), \forall u \in \mathcal{U}, \tag{2}$$

$$s.t. \sum_{u \in \mathcal{U}} ComR_u^l(t) \leq Z^l, \forall l \in \mathcal{L}, \tag{3}$$

where $a_u^l(t)$ means whether user $u$ connects LEO/MEO $l$. Let $a_u^l(t) = 1$ represent user $u$ connects certain LEO/MEO $l$. Otherwise, $a_u^l(t) = 0$.

### 2.4 Computing Model

We assume $T_l = \{o_l, n_l\}$ is the computation task of LEO/MEO $l$, with the data size of $o_l$, and required CPU cycle of $n_l$.

Since we have no idea on exactly know how many computation capabilities are allocated to LEO/MEO $l$. Thus, we model the computation capabilities of MEC server $m$ allocated to LEO/MEO $l$ as a random variable $\mathcal{Q}_l^m$. The value of $\mathcal{Q}_l^m$ can be splited into $M'$ discrete intervals, i.e., $\Pi = \{\Pi_0, \Pi_1, ..., \Pi_{M'-1}\}$. The computation capability $\mathcal{Q}_l^m$ at time instant $t$ can be represented by $\boldsymbol{\mathcal{Q}_l^m(t)}, t \in \{0, 1, 2, ..., T-1\}$. Computation capability $\boldsymbol{\mathcal{Q}_l^m(t)}$ changes from one state to another based on transition possibility. Here $\varepsilon_{\propto_{\bar{s}} \varpi_{\bar{s}}}(t)$ represent the transition probability of $\boldsymbol{\mathcal{Q}_l^m(t)}$ from one state $\propto_{\bar{s}}$ to another state $\varpi_{\bar{s}}$ at time instant $t$. The transition probability matrix $E_l^m(t)$ could be denoted as:

$$E_l^m(t) = [\varepsilon_{\propto_{\bar{s}} \varpi_{\bar{s}}}(t)]_{M' \times M'}, \tag{4}$$

where $\varepsilon_{\propto_{\bar{s}} \varpi_{\bar{s}}}(t) = Pr(\boldsymbol{\mathcal{Q}_l^m(t+1)} = \varpi_{\bar{s}} | \boldsymbol{\mathcal{Q}_l^m(t)} = \propto_{\bar{s}})$, and $\varpi_{\bar{s}}, \propto_{\bar{s}} \in \Pi$.

The task execution time $T_l$ at MEC server $m$ can be denoted as

$$t_l^m = \frac{n_l}{\boldsymbol{\mathcal{Q}_l^m(t)}}. \tag{5}$$

So the computation rate is:

$$CompR_l^m(t) = a_l^m(t)\frac{o_l}{t_l^m} = a_l^m(t)\frac{\boldsymbol{\mathcal{Q}_l^m(t)}o_l}{n_l}, \tag{6}$$

$$s.t. \sum_{l \in \mathcal{L}} a_l^m(t)o_l \leq O_m, \tag{7}$$

where $a_l^m(t)$ means whether LEO/MEO $l$ offloads computation tasks to certain MEC server $m$. Let $a_l^m(t) = 1$ denote LEO/MEO $l$ offloads tasks to MEC server $m$. Otherwise, $a_l^m(t) = 0$. $O_m$ is the maximum size of tasks that can execute simultaneously on MEC server $m$.

Thus, jointly optimize the networking and computation offloading strategy is a complex and high-dimensional system. Tradition approach is difficult to work out. So it is necessary to use machine learning algorithms, such as deep Q-learning, to learn the law of the system. In this way, a viable solution could be given. Thus, in this paper, we propose a deep Q-learning approach to solve computation offloading problem in satellite-terrestrial networks.

## 3 Problem Formulation

In this section, we formulate a joint optimization problem of networking and edge computation capability in satellite-terrestrial network. Then we propose a deep Q-learning approach to solve it.

### 3.1 State Space

The state space is consists of networking state, computing state. Then, state space $S(t)$ at time slot $t$ is represented as:

$$S(t) = \begin{bmatrix} \mathbf{h}_u^1(t) & \mathbf{h}_u^2(t) & ... & \mathbf{h}_u^l(t) & ... & \mathbf{h}_u^L(t) \\ \boldsymbol{\mathcal{Q}_l^1(t)} & \boldsymbol{\mathcal{Q}_l^2(t)} & ... & \boldsymbol{\mathcal{Q}_l^m(t)} & ... & \boldsymbol{\mathcal{Q}_l^M(t)} \end{bmatrix}, \tag{8}$$

### 3.2 Action Space

In this system, controller needs to decide which LEO/MEO $l$ should provide networking resource to $u$, and which MEC server $m$ should execute the computation task for LEO/MEO $l$. Thus, action space $a_u(t)$ at time instant $t$ is represented as:

$$a_u(t) = \{ComA_u(t), CompA_l(t)\}, \tag{9}$$

where $ComA_u(t)$, and $CompA_l(t)$ mean:

(1) $ComA_u(t) = [ComA_u^1(t), ..., ComA_u^L(t)]$, where $ComA_u^l(t), \forall l \in \mathcal{L}$ means whether or not LEO $l$ provide networking resource to user $u$. $ComA_u^l(t) \in \{0, 1\}$, where $ComA_u^l(t) = 0$ means LEO $l$ is not assigned to user $u$ at time instant $t$, otherwise $ComA_u^l(t) = 1$. Note that, in one time slot, only one LEO to provide user $u$ with networking resources.

(2) $CompA_l(t) = [CompA_l^1(t), ..., CompA_l^M(t)]$, where $CompA_l^m, \forall m \in \mathcal{M}$ means whether or not the LEO/MEO $l$ offload tasks to MEC server $m$. And $CompA_l^m(t) \in \{0, 1\}$, where $CompA_l^m(t) = 0$ means LEO/MEO do not offload to MEC server $m$, otherwise $CompA_l^m(t) = 1$. Note that there is only one MEC server to do the computation task for LEO/MEO $l$ at one time slot.

### 3.3 Reward Function

According to the work in [9], the controllers need to pay the usage of wireless spectrums in LEO/MEO $l$, and the computation fee in MEC server $m$, which are denoted as $\delta_l$ per Hz and $\eta_m$ per Joule.

In addition, the controllers charge user $u$ for the fees of accessing the networks, and using the computation resources, which are represented by $\tau_u$ per bps, and $\phi_u$ per bps.

$$
\begin{aligned}
R_u(t) &= \sum_{l\in\mathcal{L}} R_{u,l}^{comm}(t) + \sum_{m\in\mathcal{M}} R_{l,m}^{comp}(t) \\
&= \sum_{l\in\mathcal{L}} ComA_u^l(t)(\tau_u ComR_u^l(t)/\delta_l B_u^l(t)) \\
&+ \sum_{m\in\mathcal{M}} CompA_l^m(t)(\phi_u CompR_l^m(t)/\eta_m n_l e_m) \\
&= \sum_{l\in\mathcal{L}} ComA_u^l(t)(\tau_u B_u^l(t)\nu_u^l(t)/\delta_l B_u^l(t)) \\
&+ \sum_{m\in\mathcal{M}} CompA_l^m(t)(\phi_l \frac{\boldsymbol{\mathcal{Q}_l^m(t)} o_l}{n_l}/\eta_m n_l e_m) \\
&(utility/resource).
\end{aligned}
\tag{10}
$$

We define reward function $R_u(t)$ at time instant $t$ as the expected utility per resource. It is the ratio between the charging fee of using the resource and the paid fee of having the resource. The larger expected utility per resource means the higher efficiency of unit resource. Thus, $R_u(t)$ can be expressed as (10), where $e_m$ denotes the energy consumption for performing one CPU cycle.

### 3.4 Deep Q-Learning Algorithm

DQN use the interaction of agent-environment to optimize the action choosing. At one decision episode, the learning agent senses state $S(t)$ from the environment. According to a given policy, the agent chooses action $a_u(t)$ to execute. Then the environment turns to a new state $S(t+1)$, and the agent gains the corresponding reward $R_u(t)$.

In tradition Q-learning algorithm, the given policy is defined by Q-table. However. when the state space and the action space are high-dimensional. It is difficult to have all $Q(s,a)$ and put them into Q-table. Therefore, we use deep networks to evaluate $Q(s,a)$, i.e., $Q(s,a,w) \approx Q(s,a)$, where $w$ is the set of weights and biases in deep networks [10]. In each learning iteration, deep networks are trained to minimize loss function $L(w)$, so as to evaluate real $Q(s,a)$.

There are two innovations to make DQL more efficient and more robust, includes experience replay and fixed target deep networks [10]. Deep Q-learning is shown in Algorithm 1, where $\epsilon$-greedy policy is used to balance exploitation and exploration.

## 4 Simulation Results and Discussions

In this section, we evaluate the performance of the proposed algorithm in satellite-terrestrial networks. First, we present simulation settings, followed by some discussions about simulation results.

**Algorithm 1.** Deep Q-learning

Initialization:
Initialize evaluated deep networks with the set of weights and biases $w$.
Initialize target deep networks with the set of weights and biases $w'$.
**for** $k = 1 : K$ **do**
  Reset the environment with an arbitrary observation $S_{ini}$, and $S(t) = S_{ini}$.
  **while** $S(t)! = S_{terminal}$ **do**
    Select action $a_u(t)$ based on $\epsilon$-greedy policy.
    Obtain immediate reward $R_u(t)$ and next observation $S(t+1)$.
    Store the experience $(S(t), a_u(t), R_u(t), S(t+1))$ into the experience replay memory.
    Randomly sample some batches of them from the experience replay memory.
    Calculate target Q-value $Q_{target}(t)$ in target deep networks:
    if $s'$ is $s_{terminal}$
      $Q_{target}(t) = R_u(t)$,
    else
      $Q_{target}(t) = R_u(t) + \gamma_q \max_{a'} Q(s', a', w')$.
    Train evaluated deep networks to minimize loss function $L(w)$.
    Every some steps, update target deep networks.
    $S(t) \leftarrow S(t+1)$
  **end while**
**end for**

### 4.1 Simulation Settings

In this simulation, hardware environment is a GPU-based server, and this server has 8 GB 1867 MHz LPDDR3, 2 GHz Intel Core i5, and 256G memory. Software environment is Python 2.7.10 with Tensorflow 1.4.0. Note that, we use a seven-layer deep networks in the simulation.

We assume that the state of wireless channels between user $u$ and LEO/MEO $l$ can be strong (spectrum efficiency $\nu_u^l = 10$), commom (spectrum efficiency $\nu_u^l = 2$), and poor (spectrum efficiency $\nu_u^l = 0.2$), whose transition probability matrix is

$$\Upsilon = \begin{bmatrix} 0.5 & 0.4 & 0.1 \\ 0.1 & 0.5 & 0.4 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}. \tag{11}$$

We also assume the computation capability states of MEC servers are strong (computation rate $\boldsymbol{\mathcal{Q}_u^m(t)} = 50$), common (computation rate $\boldsymbol{\mathcal{Q}_u^m(t)} = 10$), and poor (computation rate $\boldsymbol{\mathcal{Q}_u^m(t)} = 1$), whose transition probability matrix is

$$E = \begin{bmatrix} 0.5 & 0.3 & 0.2 \\ 0.2 & 0.5 & 0.3 \\ 0.3 & 0.2 & 0.5 \end{bmatrix}. \tag{12}$$

The values of the rest parameters in the simulation are listed in Table 1.

**Table 1.** Parameters setting in the simulation.

| Parameters | Values | Descriptions |
|---|---|---|
| $B_u^l$ | 5 MHz | The bandwidth of LEO $l$ allocated to user $u$ |
| $\delta_l$ | 2 units/MHz | The unit paid-price of using wireless spectrums |
| $\eta_m$ | 2 unit/J | The unit paid-price of using MEC servers |
| $\tau_u$ | 10 unit/Mbps | The unit charging-price of using networking resources |
| $\phi_u$ | 10 unit/Mbps | The unit charging-price of using computing resources |
| $n_u$ | 5 Mcycles | The required number of CPU cycles to finish each computation task |
| $e_m$ | 1 J | The energy consumption for performing one CPU cycle |
| $o_u$ | 2 Mbits | The computation task size |

### 4.2 Simulation Results

Figure 3 shows the relationship between training episodes and the expected utility per resource, including networking and computing. The learning agent runs in AdamOptimizer [11] with the learning rate of 0.001, and $\epsilon$-greedy linearly decreasing from 1 to 0.1 in 500 training episodes. As shown in (10), with the increase of training episodes, the expected utility per resource increases until convergence. Finally, the expected utility reach a stable value. And with the joint consideration of networking and edge computing resources, the proposed DRL-based scheme has the better performance than only consider single aspect.

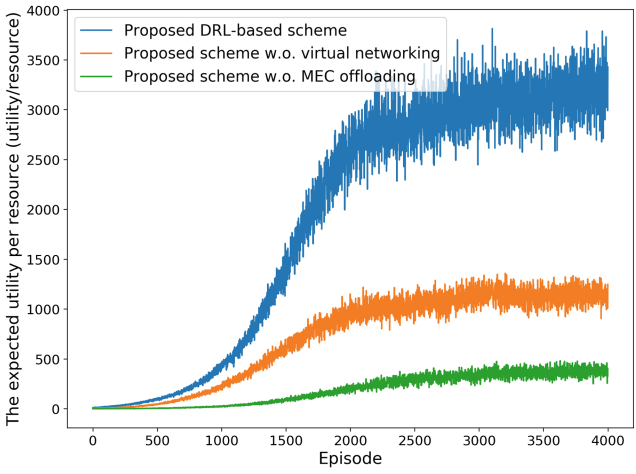


**Fig. 3.** Training curves tracking the expected utility per resource under different schemes.

Figure 4 shows the relationship between the learning loss and training steps, when the learning agent runs in AdamOptimizer with the learning rate of 0.001. At the beginning, target deep networks and evaluated deep networks have the

similar weights and biases, thus the loss values are low. Along with the training process, evaluated deep networks learn the environment and store experiences into the experience replay memory, this increase the differences between target deep networks and evaluated deep networks. Evaluated deep networks are trained by these differences to decrease the learning loss. The decrease of learning loss represents the effectiveness of the deep networks.

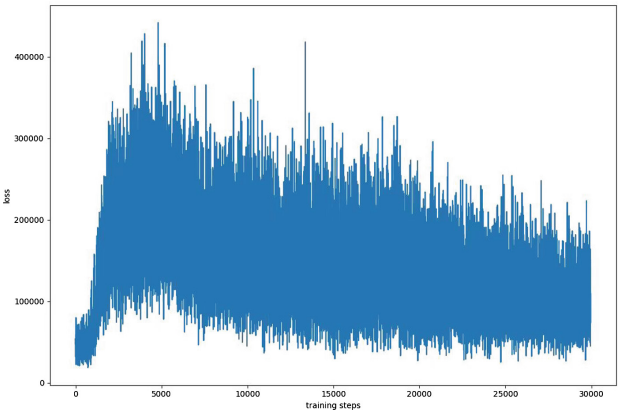


**Fig. 4.** Training curves tracking the learning loss under the proposed DRL-based scheme.

## 5 Conclusions

In this paper, we proposed an edge computing based software-defined satellite-terrestrial network architecture. Additionally, based on the architecture, we formulated the joint optimization problem of networking and edge computing resources in the network. Then we utilized a deep Q-learning approach to solve it. Simulation results showed the effectiveness and the convergence performance of our proposed scheme with different scenarios. Some future works should consider the mobility scenarios, which is used to better generalize the proposed approach.

## References

1. Kaneko, K., Nishiyama, H., Kato, N., Miura, A., Toyoshima, M.: Construction of a flexibility analysis model for flexible high-throughput satellite communication systems with a digital channelizer. IEEE Trans. Veh. Technol. **67**(3), 2097–2107 (2018)
2. Yao, H., Wang, L., Wang, X., Lu, Z., Liu, Y.: The space-terrestrial integrated network (STIN): an overview. IEEE Commun. Mag. **16**(99), 2–9 (2018)
3. Yang, B., Yue, W., Chu, X., Song, G.: Seamless handover in software-defined satellite networking. IEEE Commun. Lett. **20**(9), 1768–1771 (2016)

4. Feng, B., et al.: HetNet: a flexible architecture for heterogeneous satellite-terrestrial networks. IEEE Netw. **31**(6), 86–92 (2017)
5. Li, T., Zhou, H., Luo, H., You, I., Xu, Q.: SAT-FLOW: multi-strategy flow table management for software defined satellite networks. IEEE Access **5**, 14952–14965 (2017)
6. Boero, L., Marchese, M., Patrone, F.: The impact of delay in software-defined integrated terrestrial-satellite networks. China Commun. **15**(8), 11–21 (2018)
7. Liu, J., Shi, Y., Zhao, L., Cao, Y., Sun, W., Kato, N.: Joint placement of controllers and gateways in SDN-enabled 5G-satellite integrated network. IEEE J. Sel. Areas Commun. **36**(2), 221–232 (2018)
8. Huang, H., Guo, S., Wang, K.: Envisioned wireless big data storage for low-earth-orbit satellite-based cloud. IEEE Wirel. Commun. **25**(1), 26–31 (2018)
9. He, T.Y., Zhao, N., Yin, H.: Integrated networking, caching and computing for connected vehicles: a deep reinforcement learning approach. IEEE Trans. Veh. Technol. (2017)
10. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
11. Chilimbi, T.M., Suzue, Y., Apacible, J., Kalyanaraman, K.: Project adam: building an efficient and scalable deep learning training system. In: OSDI, vol. 14, pp. 571–582 (2014)