On-Policy vs. Off-Policy Deep Reinforcement Learning for Resource Allocation in Open Radio Access Network

Nessrine Hammami and Kim Khoa Nguyen
École de Technologie Supérieure (ÉTS), University of Quebec, Canada
Email: nessrine.hammami.1@ens.etsmtl.ca, kim-khoa.nguyen@etsmtl.ca

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771605

***Abstract*—Recently, Deep Reinforcement Learning (DRL) has increasingly been used to solve complex problems in mobile networks. There are two main types of DRL models: off-policy and on-policy. Both of them have been shown to have advantages. While off-policy models can improve sample efficiency, on-policy models are generally easy to implement and have stable performance. Therefore, it becomes hard to decide the appropriate model in a given scenario. In this paper, we compare an on-policy model: Proximal Policy Optimization (PPO) with an off-policy model: Sample Efficient Actor-Critic with Experience Replay (ACER) in solving a resource allocation problem for a stringent Quality of Service (QoS) application. Results show that for an Open Radio Access Network (O-RAN) with latency-sensitive and latency-tolerant users, both DRL models outperform a greedy algorithm. We also point out that the on-policy model can guarantee a good trade-off between energy consumption and users latency, while the off-policy model provides a faster convergence.**

***Index Terms*—5G, O-RAN, Resource allocation, PPO, ACER.**

I. Introduction

Reinforcement learning (RL) is an approach that studies how an agent interacts with an environment to learn to maximize an accumulated reward. To further improve the performance of this approach in solving high-complexity problems, Deep Neural Networks (DNN) have been successfully combined with RL, resulting in the powerful technique of Deep RL (DRL) [1]. There are two main types of RL models, and thus of DRL models: off-policy and on-policy. An on-policy model collects a batch of behavior and then computes and applies a policy gradient update from this data induced by the current policy of the agent [2]. On-policy models are generally stable and easy to use and implement, but they are data inefficient [2]. An off-policy model reuses samples stored in a memory replay buffer and trains a value function or a Q-function based on off-policy updates [2]. Off-policy models tend to be data efficient but unstable and often hard to use [2]. Both types of models have been shown to have advantages. Therefore, it becomes hard to decide the appropriate model in a given scenario. Recently, both of them have been increasingly employed to solve resource allocation problems in mobile networks. In [3], the authors applied DRL to build a power-efficient resource allocation framework in cloud RANs. In [4], the authors proposed a DRL approach for resource orchestration in a multi-tenant network. To solve the joint resource allocation and computation offloading optimization problem in Mobile Edge Computing (MEC), the authors in [5] used the Deep Deterministic Policy Gradient (DDPG). All these prior works applied a single DRL technique: either on-policy or off-policy. Therefore, it is hard to conclude if their DRL model is the most appropriate solution for the resource allocation problem.

To investigate the performance of on-policy and off-policy models, the specific scenario of resource allocation for real-time surveillance video is examined in the context of a dynamic slicing Open Radio Access Network (O-RAN) with latency as a Quality of Service (QoS) metric. The low cost of the wireless video surveillance cameras (VSCs) leads to an increased deployment of the real-time surveillance video application in many fields (e.g., public safety, intelligent transportation systems (ITS), etc.) [6]. This application has strict QoS requirements to maintain user satisfaction. Therefore, delivering real-time surveillance videos in resource-limited networks is a challenging task that calls for innovative solutions. In [7], the Deep Q-network (DQN) has been used to make a joint decision for task offloading, wireless channel allocation, and image compression rate selection in the context of MEC-based video surveillance network for face recognition applications. The authors in [8] studied the case of video surveillance in a construction site where computation and networking resources are often limited. They proposed an edge-based solution with a graph-assisted hierarchical DQN algorithm to guarantee reliable accuracy and low delay. In [9], a video surveillance application in ITS has been studied, and an Asynchronous Advantage Actor Critic (A3C) based solution has been proposed to solve the resource allocation and latency reduction problem. All these works [7]–[9] used a single type of DRL models without comparing the performance of the off-policy and on-policy models in the video surveillance resource allocation problem. This motivates us in this paper to answer the following question: Which type of DRL models is more efficient in solving the resource allocation problem, e.g., for real-time surveillance video?

To answer the aforementioned question, the O-RAN architecture is adopted because it is a suitable technology for DRL implementation. O-RAN has been proposed recently to provide an open, intelligent, virtualized, and fully interoperable RAN. It includes mechanisms for enabling Artificial Intelligence (AI) for more efficient network management and

orchestration [10]. The O-RAN architecture is divided into two parts: the radio part and the management part. The radio part includes the near-real-time RAN Intelligent Controller (Near-RT RIC) responsible for near-real-time control of RAN elements and resources, the Central Unit (O-CU), the Distributed Unit (O-DU) and the Radio Unit (O-RU). The management part includes the Service Management and Orchestration system (SMO) containing the non-real-time RAN Intelligent Controller (Non-RT-RIC) responsible for non-real-time control RAN elements and resources and policy-based guidance of applications in Near-RT RIC [10]. Both O-RAN RICs are suitable platforms for hosting AI workflows, in particular DRL models. These models can be trained in the O-RAN ML training host and the ML inference host will generate the actions accordingly [10].

The main contributions in this paper are summarized as follows:

- We evaluate the performance of an on-policy model, namely Proximal Policy Optimization (PPO) and an off-policy model– Sample Efficient Actor-Critic with Experience Replay (ACER) in finding an efficient allocation strategy of network resources for real-time video surveillance application.
- Unlike prior work that focuses exclusively on latency-sensitive video consumers, in this model, we consider a dynamic slicing O-RAN system composed of both types of users: latency-sensitive and latency-tolerant users.
- We formulate the problem as a Mixed-Integer Programming (MIP) model and solve it to determine the optimal bounds used to evaluate the on-policy and off-policy DRL models.

The remainder of this paper is organized as follows. Section II presents the system model based on the O-RAN architecture and formulates the resource allocation problem as a MIP model. In Section III, the DRL-based resource allocation solution is introduced and the characteristics of both models are compared. In Section IV, the performance of the two DRL models is evaluated. The paper is concluded in Section V.

II. System Model

A. System Model

As shown in Fig. 1, a slicing O-RAN system is considered, consisting of N O-RUs connected to a pool of M virtualized O-DUs (vO-DUs). vO-DUs are O-DUs implemented as virtual network functions (VNFs) running on a cloud environment. End-users with different QoS requirements can associate with a nearby O-RU and have access to the network services via wireless communication links. Two types of services differentiated by their latency are considered. The first service is the latency-sensitive service with a maximum tolerable latency constraint. The second is the latency-tolerant service. Therefore, two slices are established accordingly: a slice serving VSCs to transmit real-time videos via O-RAN to the 5G Core Network (CN) and then to a Control Center (CC) (e.g., a police station) for real-time monitoring, and a slice

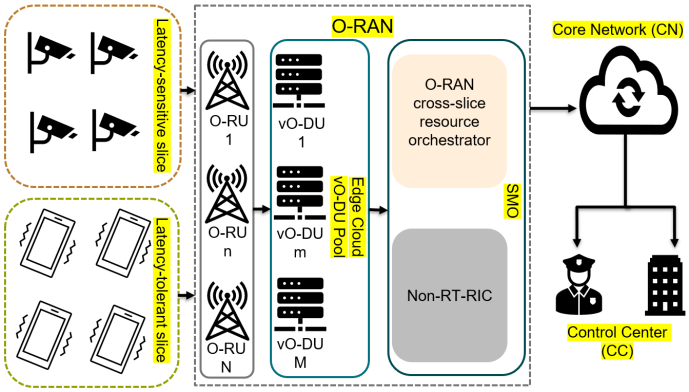

Fig. 1: System model.

serving the latency-tolerant users. These slices are managed by an O-RAN cross-slice resource orchestrator hosted by the SMO. The set of VSCs is denoted by W and the set of latency-tolerant users is denoted by V. c_w and b_v are the demands of VSC w and latency-tolerant user v in computing resources, $\forall w \in W, \forall v \in V$, respectively. The capacity of computing resources is measured by the number of CPUs. It is assumed that all CPUs are running at a fixed frequency F.

The system is assumed to be time-discrete with T slicing windows. At the beginning of each slicing window, the O-RAN cross-slice resource orchestrator decides and then performs the resource allocation actions. As a result, the O-RAN state is updated. During the remaining period of the slicing window, traffic is stationary and resource allocation decisions are unchanged. A resource allocation decision determines the association of an end-user to a vO-DU. Thus, the two following decision variables are defined:

$$x^t_{m,w} = \begin{cases} 1, & \text{if } c_w \text{ of vO-DU } m \text{ is allocated to VSC } w \\ & \text{during slicing window } t. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$y^t_{m,v} = \begin{cases} 1, & \text{if } b_v \text{ of vO-DU } m \text{ is allocated to user } v \\ & \text{during slicing window } t. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The CPU utilization of a vO-DU m at slicing window t is defined as:

$$z^t_m = \sum_{w=1}^{W} x^t_{m,w} c_w + \sum_{v=1}^{V} y^t_{m,v} b_v \tag{3}$$

It is assumed that the traffic of the latency-tolerant users from an O-RU to a vO-DU follows an $M/M/1$ queue with the arrival rate λ_v ,$\forall v \in V$. Thus, the total arrival rate of latency-tolerant users at a vO-DU is $\sum_{v=1}^{V} \lambda_v$.
For the simplicity of analysis, each VSC is assumed to monitor a specific area of interest $24/7$ and saves data in its FIFO queue. Also, all VSCs are identical and each VSC captures during a fixed period of time a video of a fixed size and length. All VSCs capture at a fixed rate λ and process data at a fixed

service rate μ which can be modeled as $D/D/1$ queue. It is assumed that $\mu > \lambda$ so that the queueing delay in the VSC queue is negligible. The VSC packet transmission from an O-RU to a vO-DU can be modeled as a $D/D/1$ queue with an arrival rate μ, according to a deterministic process. The total arrival rate of VSCs packets at a vO-DU is $\sum_{w=1}^{W} \mu = W\mu$.

The latency of a VSC packet w is equal to the time to transmit it and process it from O-RU n to vO-DU m. It can be calculated as the sum of the propagation delay, the queueing delay and the computing delay:

$$D_w = DP_{m,n} + DQ_{m,w} + DC_{m,w} \tag{4}$$

a) Propagation delay: The packet propagation delay from O-RU n to vO-DU m is equal to the distance between O-RU n and vO-DU m divided by the signal propagation speed:

$$DP_{m,n} = \frac{d_{n,m}}{s} \tag{5}$$

b) Queueing delay: The queueing delay experienced by a VSC packet w sent from O-RU n to vO-DU m is defined as:

$$DQ_{m,w} = \frac{1}{(z_m^t F) - (W\mu + \sum_{v=1}^{V} \lambda_v)} \tag{6}$$

$z_m^t F$ is the vO-DU m service rate and $W\mu + \sum_{v=1}^{V} \lambda_v$ is the packets arrival rate at vO-DU m.

c) Computing delay: The computing delay for processing a VSC packet w in vO-DU m using computing resources c_w is equal to the packet size divided by the vO-DU m service rate:

$$DC_{m,w} = \frac{\beta_w}{z_m^t F} \tag{7}$$

B. Problem Formulation

To evaluate the performance of the system model, the power consumption of the vO-DU pool is defined. At a slicing window t, it is calculated for an active vO-DU m as [11]:

$$P_m^t = P(0\%) + (P(100\%) - P(0\%))(2z_m^t - (z_m^t)^{1.4}) \tag{8}$$

$P(0\%)$ and $P(100\%)$ are the power consumption of the vO-DU m in the idle mode and in full load, respectively. The M vO-DUs are identical and have the same $P(0\%)$ and $P(100\%)$ values.

The objective is to associate end-users to vO-DUs with the aim to minimize the power consumption of the vO-DU pool while satisfying the QoS constraints for the different types of end-users. Therefore, the problem is formulated as follows:

$$\min_{x_{m,w}^t, y_{m,v}^t} \sum_{t=1}^{T} \sum_{m=1}^{M} P_m^t \tag{9}$$

subject to:

$$z_m^t \leq Z_m^{max}, \forall m \in M \tag{9a}$$

$$D_w < D^{th}, \forall w \in W \tag{9b}$$

$$(z_m^t F) - (W\mu + \sum_{v=1}^{V} \lambda_v) \geq 0, \forall m \in M \tag{9c}$$

Z_m^{max} denotes the maximum computing capacity of a vO-DU m. Constraint (9a) is the system stability constraint, which guarantees that the total amount of allocated computing resources in a vO-DU m cannot exceed its capacity. Constraint (9b) states that the latency experienced by each VSC packet should be less than the maximum tolerable latency for latency-sensitive slice. Constraint (9c) is the queueing stability constraint for latency-tolerant users. This computing resource allocation problem is formulated as MIP model.

III. DRL-based Computing Resource Allocation

The decision process of the computing resource allocation is modeled as a Markov decision process (MDP). The MDP is presented as a five-tuple $< S, A, P(s'|s,a), R, \gamma >$, where S is a finite state space, A is a finite action space, $P(s'|s,a)$ is the transition probability that in state $s \in S$ the action $a \in A$ leads to state $s' \in S$. $R(s,a)$ is the immediate reward of performing action a under state s and $\gamma \in [0,1]$ is the discount factor that reflects the importance of the current reward on future ones. The DRL agent in the system is the O-RAN cross-slice resource orchestrator. At a slicing window t, the DRL agent observes state $s(t)$ of state space S, tries to find the best policy and chooses the action $a(t)$ of action space A that maximizes the system reward $R(s,a)$.

a) State space: : The state space of the system includes the demands of VSCs and latency-tolerant users in terms of computing resources, and the network resource utilization. At a slicing window t, the observation of the DRL agent $s(t)$ is defined as follows:

$$s(t) = [c_1, ..., c_W, b_1, ..., b_V, z_1^t, ..., z_M^t] \tag{10}$$

b) Action space: : During each slicing window t, the DRL agent decides the vO-DU to be associated to the current end-user, therefore, an action $a(t)$ of the action space A is the index of the chosen vO-DU:

$$a(t) = m \in \{1, 2, ..., M\} \tag{11}$$

c) Reward: : In RL, the goal is to maximize the reward function, while the objective function of the MIP model (9) aims to minimize the power consumption of the vO-DU pool. Then, under state $s(t)$, the immediate reward function that guides the DRL agent to take action $a(t)$ toward optimizing the objective function can be defined as the value of the instantaneous power consumption of the vO-DU pool multiplied by a negative weight:

$$R(s,a) = -\alpha \sum_{m=1}^{M} P_m^t \tag{12}$$

α is a positive system parameter that defines a weight factor for the power consumption. To maximize $R(s,a)$, the DRL agent needs to minimize the power consumption by choosing the suitable end-users to vO-DUs associations.

To perform the computing resource allocation process, the two DRL models are designed, and then their performance is compared in this scenario: PPO as an on-policy model and

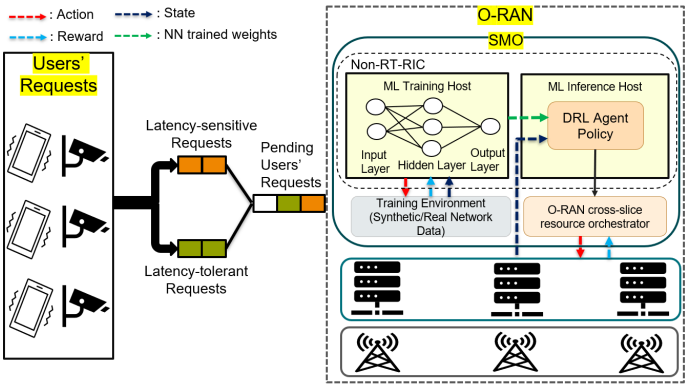

Fig. 2: Resource allocation process.

ACER as an off-policy model. Both models are model-free RL algorithms where the agent is not able to learn directly how to model the environment and is not able to predict the next state and value before taking action [12]. Instead, the agent estimates the value function of each state-action pair and derives the optimal policy [12]. Both models are also actor-critic, where two neural networks (NN) are required and the input to both actor and critic network is the state of the environment.

- The actor network: responsible for decision making and choosing the action by calculating a policy function.
- The critic network: responsible for policy evaluation and it informs the actor on how good the action was and how it should be adjusted by calculating the value function.

PPO [13] is an on-policy model that estimates the value of a policy while using it for control. It collects a small batch of experiences and uses it to update its decision-making policy by applying a policy gradient update. This batch of experiences is only useful for updating the current policy once. Thus, PPO decreases the complexity of implementation. The main advantage of PPO is solving the problem of large policy changes caused by the instability of the policy gradient process. In fact, it optimizes a clipped surrogate objective function and imposes a constraint that avoids large changes between the old policy and the updated one. Thus, it can guarantee stable policy updates.

ACER [14] is an off-policy model that makes use of a memory replay buffer to store a list of tuples (state, action, reward, next state). The stored samples are reused to train a value function based on more than one gradient off-policy update. The main advantage of ACER is sample efficiency: using each piece of sampled experience, the agent learns from sampling all the experience accumulated so far instead of learning only from recent experience.

Table I summarizes the comparison between the two DRL models.

Fig. 2 represents the workflow of the DRL-based resource allocation in the context of O-RAN architecture. According to [10], when applying RL in O-RAN, the ML training host and the ML inference host shall be co-located either as part of

TABLE I: PPO and ACER comparison

ACER	PPO
Model-free	Model-free
Policy gradient algorithm	Policy gradient algorithm
Off-policy	On-policy
Actor-Critic	Actor-Critic
Sample efficiency	Policy updates stability
Discrete, Continuous environments	Discrete, Continuous environments

Non-RT RIC or Near-RT RIC. Therefore, it is assumed that the ML training host and the ML inference host are implemented in the Non-RT RIC as part of the SMO. The DRL training process is performed in the ML training host, where the DRL agent interacts with a copy of the real environment (e.g., network simulator that imitates the behavior of the real system model, synthetic data, real network data). During each training episode, the DRL agent observes the state of the environment as in Equation (10), tries to find the best action as in Equation (11) toward maximizing the system reward as in Equation (12). Then, the resulting trained model is copied to the ML inference host as a DRL agent policy to perform dynamic resource allocation during each slicing window t. The process starts with the arrival of the end-users requests to the network. These requests are analyzed and classified into two types: latency-sensitive and latency-tolerant requests. Then, the O-RAN cross-slice resource orchestrator observes the state $s(t)$, executes the trained DRL policy to output the action $a(t)$ as a vO-DU index. Then, the required computing resources are allocated to end-users and the environment state is updated to $s'(t)$.

IV. Performance Evaluation

A. Simulation Settings

Extensive experiments are carried out on the dataset of Alibaba cluster-trace-v2018 [15]. It is an open-source, real production clusters workload traces for an 8-day-long period and 4000 machines. The dataset contains characteristics of the servers in terms of computational resources (CPU, memory, communication bandwidth, etc.) and characteristics of online tasks (i.e., latency-sensitive) and other tasks (i.e., latency-tolerant). The task trace includes the task arrival time, the task duration, and the CPU resource demand. These tasks are arranged in an ascending order based on their arrival time. The task latency is calculated using its arrival time and its duration information.

The two DRL solutions are compared with the optimal solution of the MIP model (9) which is obtained using Gurobi solver, and with a greedy solution that associates the task to the vO-DU with the lowest CPU usage. For PPO, both actor and critic networks have two dense layers with 64 neurons. For ACER, both networks have one dense layer with 256 neurons. All of them operate a hyperbolic tangent activation function (Tanh). Furthermore, to choose the actions, the actor network has an extra dense layer with a number of neurons equal to the action space A dimensions and a softmax activation function. The critic network has an extra dense layer with one neuron

and no activation function. For additional details, refer to the GitHub repository [16]. Table II gives a summary of the values of the main experimental parameters.

TABLE II: Main Simulation Settings

Parameter	Value
M	10
$P(0\%)$	$87W$
$P(100\%)$	$145W$
Actor learning rate	0.0003
Critic learning rate	0.001
γ	0.99
α	0.0001

B. Results

a) DRL Reward: To study the efficiency of the two proposed models, their average episodic reward (i.e., over timesteps and episodes) is first observed during the training process as illustrated in Fig. 3 and Fig. 4. As the training process progress, the performance curves of both models increase at different rates. The average episodic reward starts from low values in the early stages of learning, but then the DRL agent starts exploiting the environment to maximize the system reward. Despite a failed exploration at the beginning for ACER, its reward curve converges faster than PPO because ACER as an off-policy model is more sample efficient than PPO which needs more samples to converge to a good policy [2]. Fig. 3 and Fig. 4 show that the reward obtained by PPO is higher than the reward obtained by ACER. Thus, PPO outperforms ACER in learning an optimal policy toward maximizing the system reward.

b) DRL stability: Fig. 5 and Fig. 6 show the training performance of both models when changing their hyperparameters. The number of neurons in dense layers for the actor and critic networks is changed. Fig. 5 shows a difference in performance for ACER while the performance of PPO is more stable in Fig. 6. For 32 and 64 neurons, the average episodic reward of ACER does not converge, while for 256 neurons, its performance improves remarkably with a higher reward and a convergent curve. It is not the case for PPO which achieves almost the same performance for all NN architectures. This is because on-policy models tend to be more stable than off-policy ones which are more sensitive to hyperparameters changes [2].

c) Energy consumption under different levels of network load: Fig. 7 shows a comparison of the performance of the four models in terms of energy consumption when the number of end-users increases. The energy consumption of all models increases according to the increasing load generated by the users. The graph also shows that the two DRL models outperform the greedy baseline and consume less energy. The PPO model performs better than the ACER model and achieves a performance close to the optimal. This difference in performance becomes highly remarkable when the level of load increases and the network becomes congested.

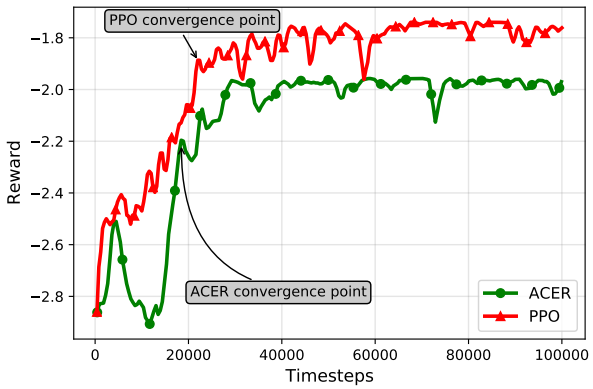

Fig. 3: Steps-reward curve for PPO and ACER.

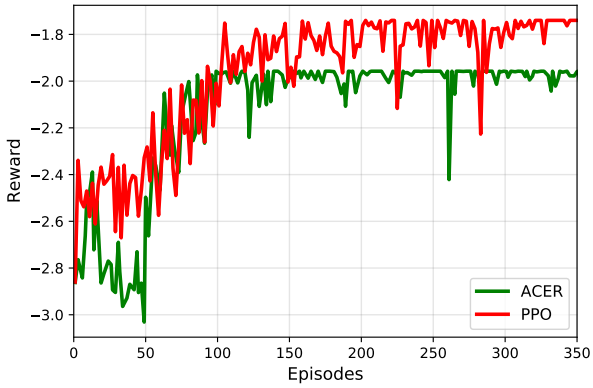

Fig. 4: Episodes-reward curve for PPO and ACER.

d) Trade-off between users latency and energy consumption: Fig. 8 shows the results of the trade-off between users latency and energy consumption for the four models. The users latency increases as the energy consumption increases. The two DRL models outperform the greedy baseline which shows the highest energy consumption and users latency. PPO learns to guarantee a trade-off between users latency and energy efficiency better than the ACER model.

V. Conclusion

In this paper, a slicing O-RAN system is explored and latency is considered as a QoS metric. To address the problem of resource allocation, two DRL models are proposed: an on-policy model, PPO, and an off-policy model, ACER. It is concluded that the overall on-policy model performance is superior to the off-policy model. In addition to its simple implementation and performance stability, PPO can learn an optimal policy that guarantees a good trade-off between users latency and energy consumption, while the sample efficiency of ACER can produce a faster convergence.

Acknowledgment

The authors thank Mitacs, Ciena, and ENCQOR for funding this research under the IT13947 grant.

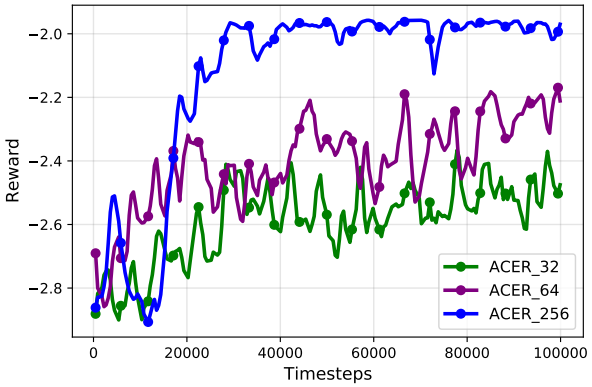

Fig. 5: ACER reward for different NN architectures.

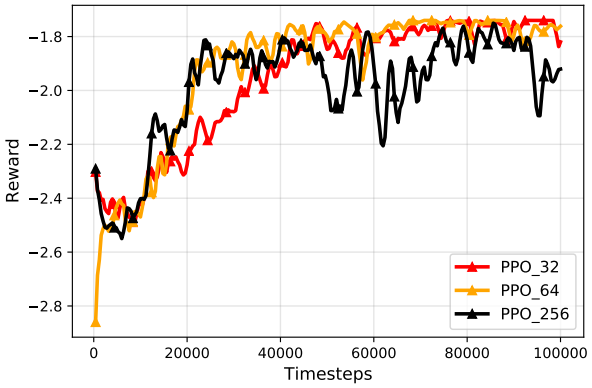

Fig. 6: PPO reward for different NN architectures.

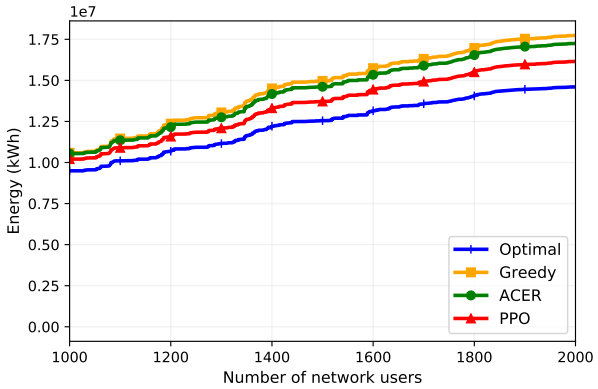

Fig. 7: Energy consumption.

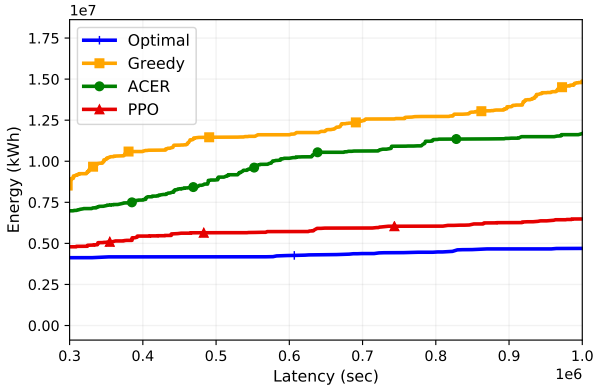

Fig. 8: Trade-off between energy consumption and users latency.

REFERENCES

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

[2] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. *ArXiv*, abs/1706.00387, 2017.

[3] Zhiyuan Xu, Yanzhi Wang, Jian Tang, Jing Wang, and Mustafa Cenk Gursoy. A deep reinforcement learning based framework for power-efficient resource allocation in cloud rans. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[4] Xianfu Chen, Zhifeng Zhao, Celimuge Wu, Mehdi Bennis, Hang Liu, Yusheng Ji, and Honggang Zhang. Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 37(10):2377–2392, 2019.

[5] Juan Chen, Huanlai Xing, Zhiwen Xiao, Lexi Xu, and Tao Tao. A drl agent for jointly optimizing computation offloading and resource allocation in mec. *IEEE Internet of Things Journal*, pages 1–1, 2021.

[6] Guanglun Huang, Baoxian Zhang, Zheng Yao, and Cheng Li. Stochastic joint rate control and resource allocation for wireless video surveillance. *Computer Networks*, 190:107904, 2021.

[7] Yang Kunpeng, Hangguan Shan, Tengxu Sun, Roland Hu, Yingxiao Wu, Lu Yu, Zhaoyang Zhang, and Tony QS Quek. Reinforcement learning-based mobile edge computing and transmission scheduling for video surveillance. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2021.

[8] Zhongxing Ming, Jinshen Chen, Laizhong Cui, Shu Yang, Yi Pan, Wei Xiao, and Lixin Zhou. Edge-based video surveillance with graph-assisted reinforcement learning in smart construction. *IEEE Internet of Things Journal*, pages 1–1, 2021.

[9] Xiantao Jiang, F Richard Yu, Tian Song, and Victor CM Leung. Intelligent resource allocation for video analytics in blockchain-enabled internet of autonomous vehicles with edge computing. *IEEE Internet of Things Journal*, pages 1–1, 2020.

[10] O-RAN ALLIANCE. ORAN-WG2.AIML.v01.00: O-RAN Working Group 2 AI/ML workflow description and requirements, Technical Report. 2019.

[11] Ning Liu, Zhe Li, Jielong Xu, Zhiyuan Xu, Sheng Lin, Qinru Qiu, Jian Tang, and Yanzhi Wang. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 372–382. IEEE, 2017.

[12] Rongpeng Li, Zhifeng Zhao, Qi Sun, I Chih-Lin, Chenyang Yang, Xianfu Chen, Minjian Zhao, and Honggang Zhang. Deep reinforcement learning for resource management in network slicing. *IEEE Access*, 6:74429–74441, 2018.

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[14] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *ArXiv*, abs/1611.01224, 2017.

[15] Alibaba production cluster data v2018, 2018. [online] Available: https://github.com/alibaba/clusterdata/tree/master/cluster-trace-v2018/.

[16] Nessrine Hammami. On-policy vs. off-policy deep reinforcement learning for resource allocation in open radio access network, 2021. [online] Available: https://github.com/nessry/ON-OFF-DRL.