Fast and Reliable Offloading via Deep Reinforcement Learning for Mobile Edge Video Computing

Soohyun Park[1], Yeongeun Kang[1], Yafei Tian[2], and Joongheon Kim[3]
[1]*School of Computer Science and Engineering, Chung-Ang University, Seoul, Korea*
[2]*School of Electronics and Information Engineering, Beihang University (BUAA), Beijing, China*
[3]*School of Electrical Engineering, Korea University, Seoul, Korea*
joongheon@korea.ac.kr

***Abstract*—In this paper, we propose an adaptive video streaming method which is inspired by deep reinforcement learning in mobile edge computing systems for autonomous driving applications. In fast moving autonomous driving applications, it is challenge to design fast and reliable video streaming (those are obtained by vision-based autonomous vehicles) task offloading. This paper handles this issue inspired by deep Q-network (DQN) which is one of the most well-known deep reinforcement learning algorithms.**

***Index Terms*—Autonomous driving, mobile edge computing, offloading, reinforcement learning, deep Q-network**

I. Introduction

Artificial Intelligence (AI) which is able to let devices or things can think and act like a human is developed and researched in many areas such as autonomous driving, unmanned systems, and robotics [1]–[3]. Especially, the autonomous driving field has been seen as a major area of research that combines AI from the use of auxiliary artificial intelligence to provide convenience to users to unmanned autonomous driving that completely excludes human intervention. Currently, further research which can prepare instantaneously and determine similarly to humans for any unexpected situations that occur during actual driving is being developed beyond the auxiliary technologies or full automation using AI [4]–[6].

In order to prepare immediately in a dynamic environment, rapid and accurate processing of large amounts of data, such as video data through cameras or data detected through sensors, is essential. For these reasons '5G', 'MEC (mobile edge computing)' technologies for high-capacity, high-speed data processing and communication were mentioned with the autonomous driving technology and as the '5G', 'MEC' technologies are progressed the autonomous driving technology also has been advanced highly.

Some video input data through a camera installed in a vehicle is used to improve perfectness of autonomous driving in such a way as detection of objects around the vehicle, recognition of traffic lights, identification of license plates, etc. When a car actually drives with autonomous driving technology, as the level of data processing required for full-autonomous driving increases, high-performance computing for video data processing will be requested. If there is a lack of internal performance of the vehicle to implement image data management and processing on the vehicle itself, it could cause serious accidents with an inappropriate judgment and unacceptable latency of autonomous vehicles. To solve these problems We propose a video data offloading & compression decision algorithm with deep reinforcement learning for optimal data processing and optimal autonomous driving through offloading in MEC environment.

II. Overall Architecture

The video data offloading algorithm proposed by this paper has 2 DQN-based decision architecture, (i) offloading decision algorithm and (ii) data compression decision algorithm. In this part, the mobile edge computing, deep reinforcement learning used in this algorithm is discussed, and the details of offloading, compression decision algorithms will be described in the next section.

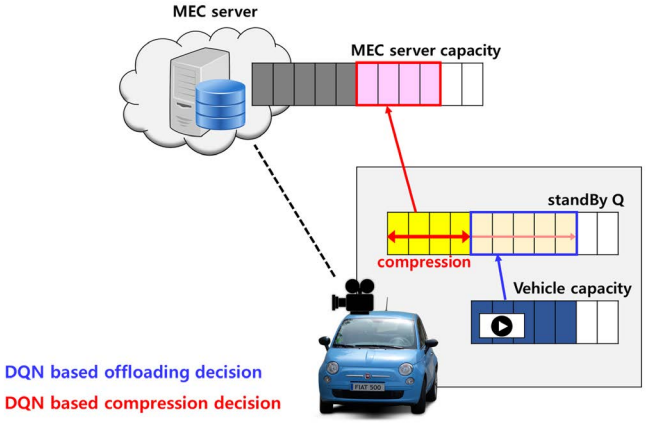

Fig. 1: overall architecture.

A. Mobile Edge Computing: Offloading

Mobile edge computing is a technology which uses an edge server with a better capacity than the local devices to

process devices' tasks or data. Because the edge server is located around wirelessly connected devices, by distributing the task to be processed on the local devices, total execution time in mobile edge computing, the sum of data transmission and processing time at the edge server, is lower than cloud computing's. And the processing burden of the device is also lower than local computing [7], [8]. When offloading is required from the terminal to the MEC server, an offloading decision is usually made according to the internal policy. Because the performance of offloading is affected by what will be offloaded and when will be offloaded, appropriate offloading decision technology is needed to achieve optimal performance [9]–[11].

B. Deep Reinforcement learning

Reinforcement learning is a way for an agent to learn action that maximizes reward that the agent can get while receiving information about the changed state at time step $t+1$ from the environment. The formula for this relationship is Q-function.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a')\right] \quad (1)$$

It returns the reward value that can be received for action in state at time step t. And Q-network is the way to learn Q-function using neural network. The expressions used in Q-network show that it minimizes a difference between the optimal value and real value (predicted value).

$$\min_{\theta} \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta)]^2 \quad (2)$$

However, since both values use the same network for Q-network, it is impossible to finally make the predicted value similar to the target value and finally the agent will learn policies in a different direction than the target. To solve this problem DQN is used. DQN has more hidden layers, can store data collected by agent in buffer and use only part of it for learning. Also, in DQN, updates of the pretend network theta do not impact to the target network. Due to these characteristics of DQN, the agent can get appropriate policies [12], [13].

III. DQN-based Offloading and Compression Decision

A. Offloading decision algorithm

We use deep Q-network (DQN) for the offloading decision and we have designed the following:

State. The state is defined by the vehicle's capacity and standBy Q capacity

Action. The action is defined by the decision of offloading (i.e., do offloading or do local computing)

Reward. The reward is defined by the combination of energy consumption and total delay the reward as follows, i.e.,

$$R = \gamma \times R_{\text{delay}} + (1-\gamma) \times R_{energy}, \quad (3)$$

$$R_{\text{delay}} = (1 - \frac{D_{\text{DQN}}}{D_{local}}), \quad (4)$$

$$R_{energy} = \left(1 - \frac{E_{\text{DQN}}}{E_{local}}\right) \quad (5)$$

This system has two components as State, i.e., vehicle's capacity and standBy Q capacity. The standBy Q is a waiting buffer for data which is determined to offload. Action means whether to offload or not, that is expressed as 0 or 1. And the agent will get a reward for the action in each step. The reward is the combination of two aspects of the return. R_{energy} is a reward about energy consumption, R_{delay} is a reward about total execution delay and the constant γ is the weight of the two aspects for the total reward. In formula (4) and (5) D_{local}, E_{local} are the values of execution delay and energy consumption using local computing. And D_{DQN}, E_{DQN} are the values using the DQN based decision algorithm, D_{DQN} and E_{DQN} are as follows:

$$D_{DQN} = \sum_{n=1}^{N} \alpha_n * D_n^o + (1-\alpha_n) * D_n^l \quad (6)$$

$$E_{DQN} = \sum_{n=1}^{N} \alpha_n * E_n^o + (1-\alpha_n) * E_n^l \quad (7)$$

α is a value of action decision. If the data is decided to offload in the step α is 1. Finally, the determination of offloading is learned to maximize total reward 'R'.

B. Compression decision algorithm

Data determined to be offload to edge server via DQN-based offloading decision algorithm waits for transmission at standBy Q inside the vehicle. At this point, the DQN-based compression decision algorithm determines whether data is compressed or not. We have designed the compression decision algorithm as follows:

State. The states are defined by two capacity values, i.e., (i) standBy Q capacity and (ii) MEC server capacity.

Action. The actions are defined by a binary variable, i.e., (i) offload without data compression and (ii) offload with data compression.

Reward. The reward is defined by the combination of data quality (denoted by $R_{\texttt{dataQuality}}$) and waiting delay (denoted by $R_{\texttt{standByDelay}}$) as follows.

$$R = R_{\texttt{dataQuality}} - R_{\texttt{standByDelay}} \quad (6)$$

The longer the data determined to be offloading remains at standBy Q, the harder it is to ensure real-time and to satisfy the tolerance latency. Therefore, it is necessary to minimize $R_{standByDelay}$ through data compression. In determining data compression, the quality of data (video data) will also be lower than that of the original, as the size of the data decreases. For this reason, we set $R_{standByDelay}$ as a negative reward and the agent will learn to minimize waiting delay at standBy Q while maintaining video data quality high in a given situation. This algorithm will enable immediate and accurate recognition

of situations and optimal response to situations in autonomous driving technologies based on video data. Subsequent studies using embedded autonomous driving platform aim to draw meaningful results of our proposed algorithm.

IV. Conclusion and Future Work

This paper proposes an algorithm for joint offloading and compression decision making which can minimize energy consumption and total execution time while maintaining the quality of offloading data via deep reinforcement learning, i.e., deep Q-network. Based on this approach, the proposed algorithm is able to maximize the reward via optimal offloading action decisions in mobile edge computing environments.

Acknowledgment

This research was supported by the IITP/MSIT (20170001000021001, National Program for Excellence in SW) and also by National Research Foundation of Korea (2019M3E4A1080391). J. Kim is the corresponding author of this paper (joongheon@korea.ac.kr).

References

[1] L. Feng, *et. al.*, "An intelligent control system construction using high-level time Petri net and Reinforcement Learning," in *Proceedings of the IEEE International Conference on Control, Automation and Systems (ICCAS)*, Gyeonggi-do, Korea, October 2010.

[2] S. Bhagat, H. Banerjee, Z. T. H. Tse, and H. Ren, "Deep Reinforcement Learning for Soft, Flexible Robots: Brief Review with Impending Challenges," in *Robotics*, January 2019.

[3] Y. Zhang, P. Sun, Y. Yin, L. Lin, and X. Wang, "Human-like Autonomous Vehicle Speed Control by Deep Reinforcement Learning with Double Q-Learning," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, June 2018.

[4] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sunm, "A Reinforcement Learning Approach to Autonomous Decision Making of Intelligent Vehicles on Highways," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, December 2018.

[5] M. Shin and J. Kim, "Randomized Adversarial Imitation Learning for Autonomous Driving," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, Macau, China, August 2019.

[6] M. Shin and J. Kim, "Randomized Adversarial Imitation Learning for Autonomous Driving," *arXiv preprint arXiv:1905.05637*, May 2019.

[7] T. X. Tran et al., "Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, Apr. 2017

[8] Pavel Mach, Member, IEEE, and Zdenek Becvar, Member, IEEE , "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol 19, issue. 3 , thirdquarter 2017

[9] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, 2017.

[10] Zhang, Yuan, et al. "To offload or not to offload: an efficient code partition algorithm for mobile cloud computing." *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*. IEEE, 2012.

[11] J. Zhang, X. Hu, Z. Ning, E. C. . Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, 2018.

[12] Kober, Jens, J. Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey." *The International Journal of Robotics Research* 32.11 (2013): 1238-1274.

[13] Arulkumaran, Kai, et al. "Deep reinforcement learning: A brief survey." *IEEE Signal Processing Magazine* 34.6 (2017): 26-38.