Joint Offloading and Streaming in Mobile Edges: A Deep Reinforcement Learning Approach

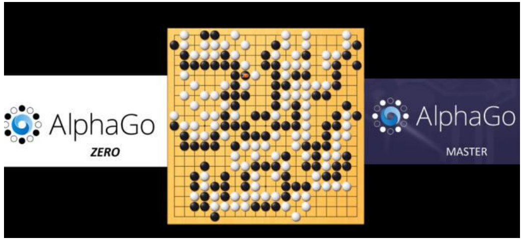
Soohyun Park, Junhui Kim, Dohyun Kwon, MyungJae Shin, and Joongheon Kim

School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea

Contents

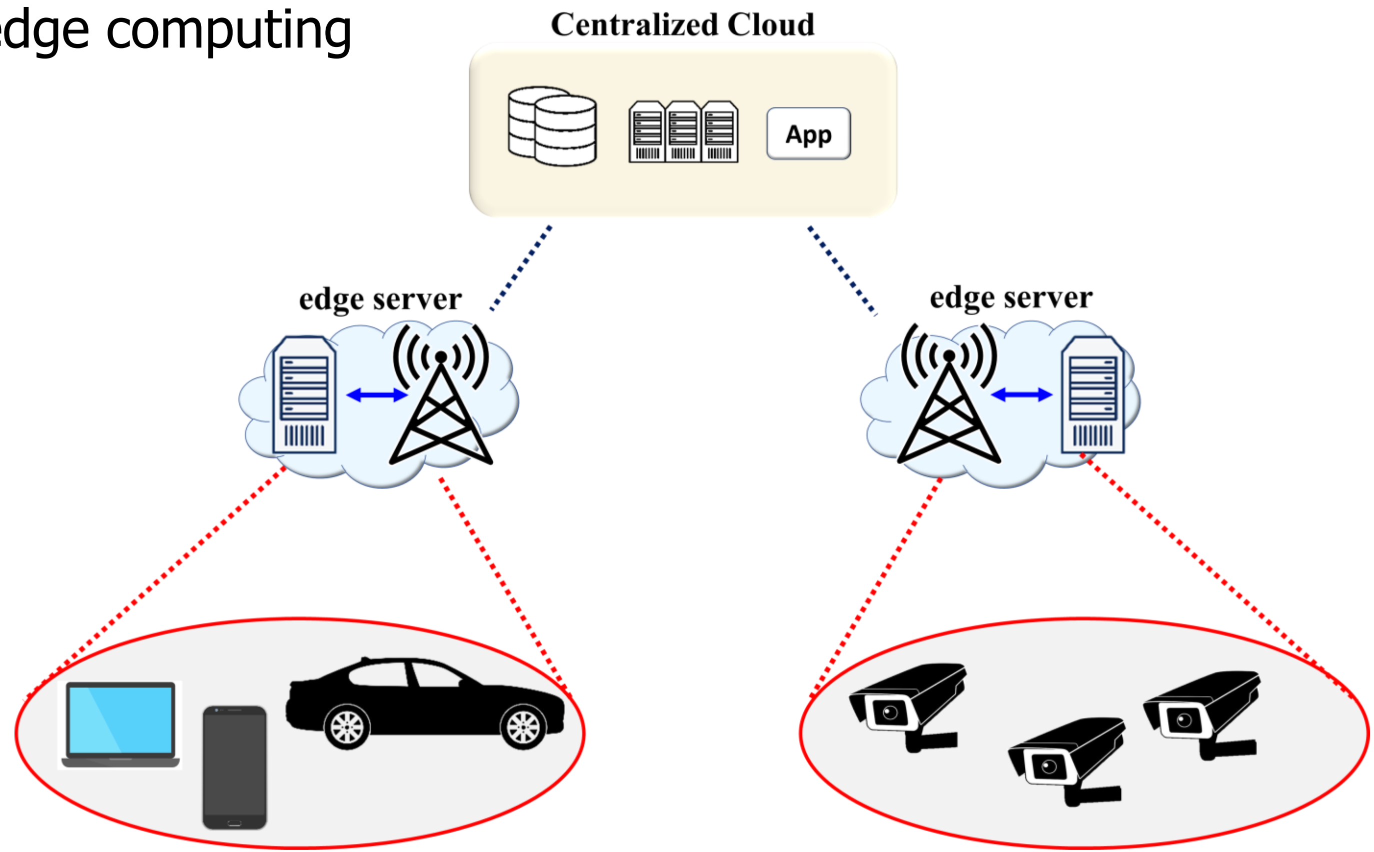
Introduction

- Sequential decision making via deep reinforcement learning
  - Intelligent system
  - Autonomous driving
  - Others – AlphaGo Zero

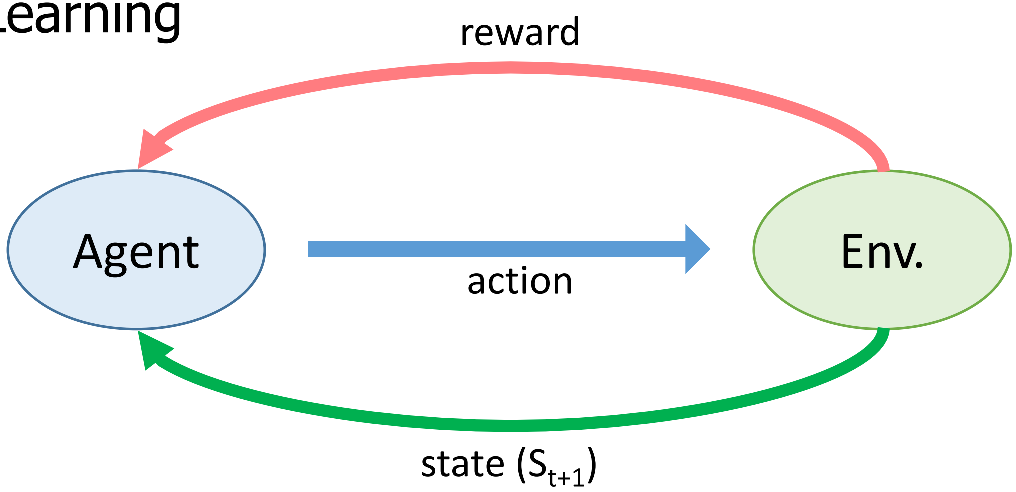


Introduction

- Mobile edge computing



- Reinforcement Learning

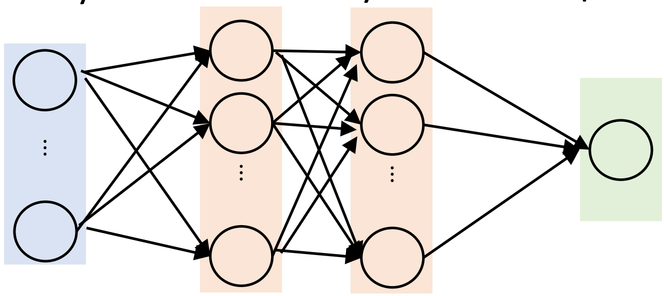


- Environment : Physical world in which the agent operates
- State : Current situation of the agent
- Action : Agent' behavior
- Reward : Goal of a problem, feedbacked form the environment

- Q-Network (Q-function approximation)

input layer hidden layers output layer

current state →



→ Action_1 : Reward_1
→ Action_2 : Reward_2
→ Action_3 : Reward_3
→ Action_4 : Reward_4

informs all of the possible actions and reward of each action

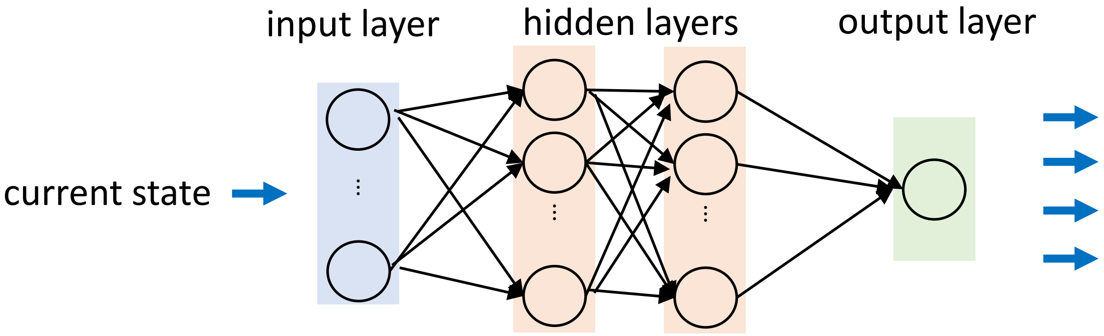
$$\min_{\theta} \sum_{t=0}^{T} \left[\underbrace{\hat{Q}(s_t, a_t|\theta)}_{Q^{pre}} - \underbrace{\left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta) \right)}_{Q^{*}} \right]^2$$

$$H(x) = Wx$$

$$\mathrm{cost}(w) = \frac{1}{m} \sum_{i=1}^{m} \left(Wx^{(i)} - y(i) \right)^2$$

which minimizes the difference between the target value and the predicted value → equal to the linear regression problem

- Q-Network (Q-function approximation)



Action_1 : Reward_1
Action_2 : Reward_2
Action_3 : Reward_3
Action_4 : Reward_4

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

‘$\hat{Q} \approx Q^*$’ is impossible

Because of
similar learning data & non-stationary target

As a result, it is learned in a different direction than the target

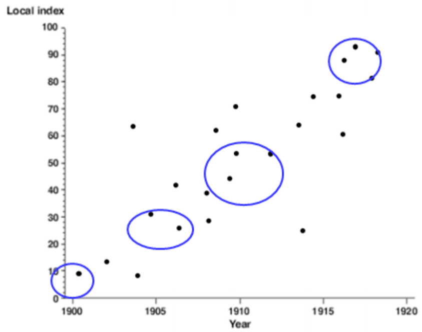
Deep Q-Network

- DQN

  ① Go deep

  ② Experience replay

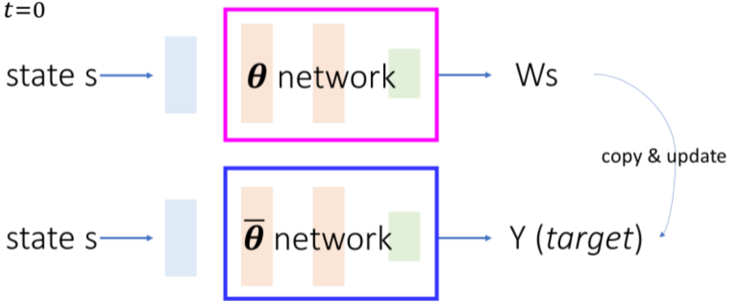
  ③ Separate target network



random sample & replay

s_1, a_1, r_1, s_2
s_2, a_2, r_2, s_3
…
s_t, a_t, r_t, s_{t+1}

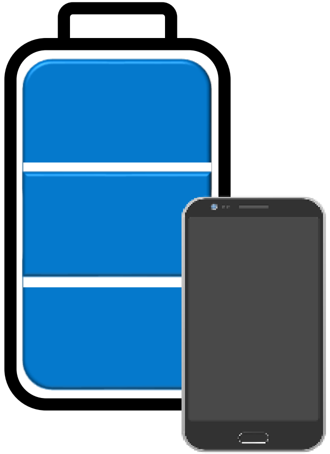
$$\min_{\theta} \sum_{t=0}^{T}\left[\hat{Q}(s_t, a_t|\theta) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\bar{\theta})\right)\right]^2$$



Soohyun Park (DPS lab.)
shpark.cau@gamil.com

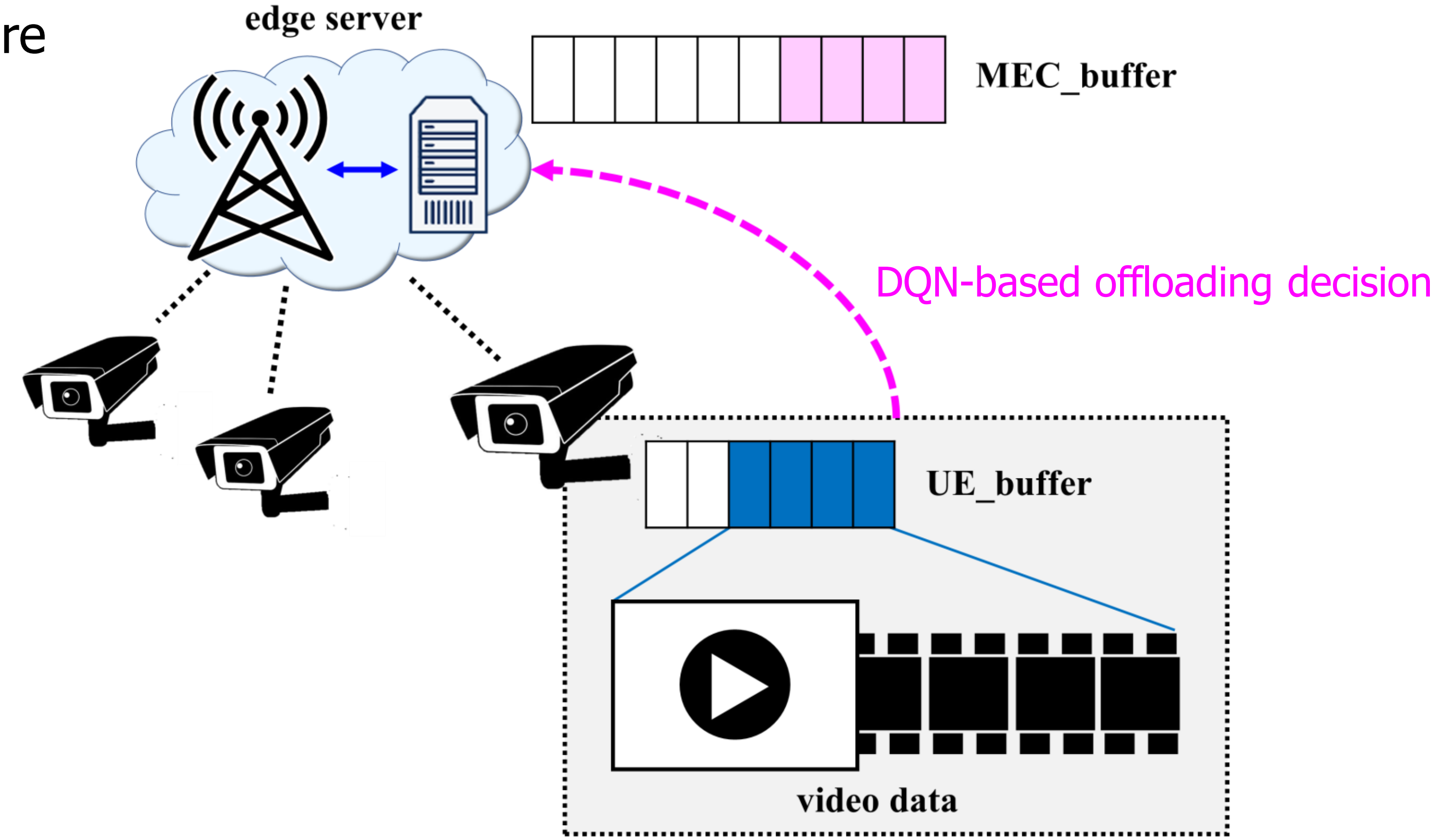
Joint streaming and Offloading

- Architecture
  - Offloading computing to edge server will be decided by a DQN based task offloading algorithm considering the energy consumption and latency.



Joint streaming and Offloading

- Architecture



- **Offloading Decision with DQN**

- State)

The system has two components, UE capacity and edge server capacity

- Action)

The action means whether to offload and is expressed as 0 or 1

- Reward)

Each agent will get a reward in each step, it is expressed as the sum of two aspects of the return.

- **Offloading Decision with DQN**

- Reward)

Each agent will get a reward in each step, it is expressed as the sum of two aspects of the return.

The reward follows the formulas

$$R = \gamma \times R_e + (1-\gamma) \times R_d,$$
$$R_e = E_{local} - E_{total},$$
$$R_d = D_{local} - D_{total},$$

γ is the weight of the two aspects for the total reward

- **Offloading Decision with DQN**

- Reward)

$$R = \gamma \times R_e + (1-\gamma) \times R_d,$$

$$R_e = E_{local} - E_{total},$$

$$R_d = D_{local} - D_{total},$$

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

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

- If $\alpha = 1$, the task of UE will be processed in the edge server
- If $\alpha = 0$, the task of UE will be processed in the UE (local computing)

- **Offloading Decision with DQN**

- Reward)

$$R = \gamma \times R_e + (1-\gamma) \times R_d,$$

$$R_e = E_{local} - E_{total},$$

$$R_d = D_{local} - D_{total},$$

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

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

- The total reward 'R' is a combination of R_e and R_d
- E_{total} and D_{total} mean the energy consumed and execution delay in the UE when all the tasks are processed in the UE based on DQN
- Finally, the determination of offloading is learned to maximize R

Experiments

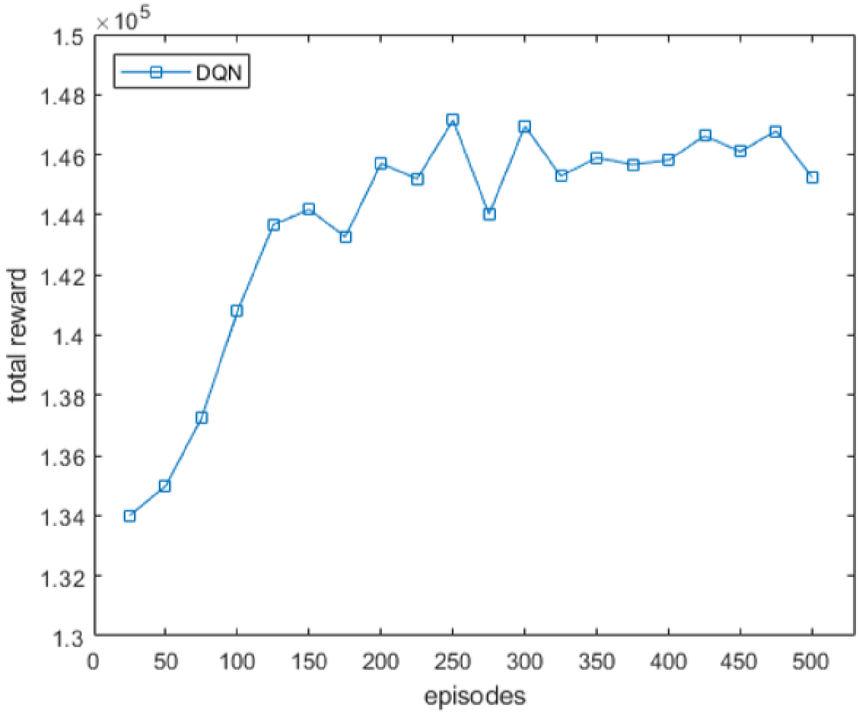


Fig. 4: Total reward versus the episode of DQN

- UE capacity : 30 ~ 50
- MEC server capacity : 800 ~ 1000
- The agent should perform 10000 tasks per episode
- Each task is selected between 0 and remained tasks uniform randomly

Experiments

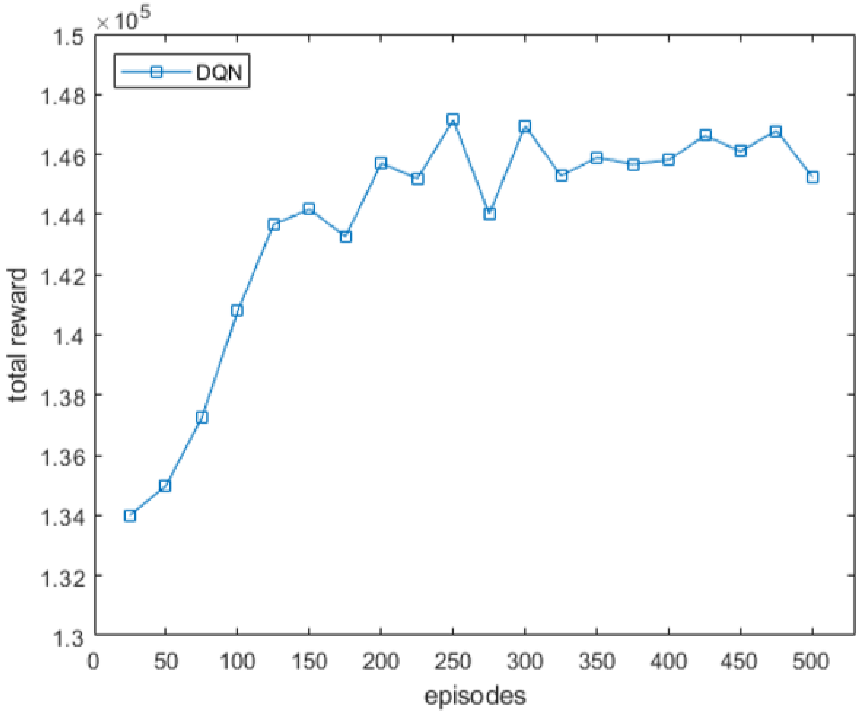
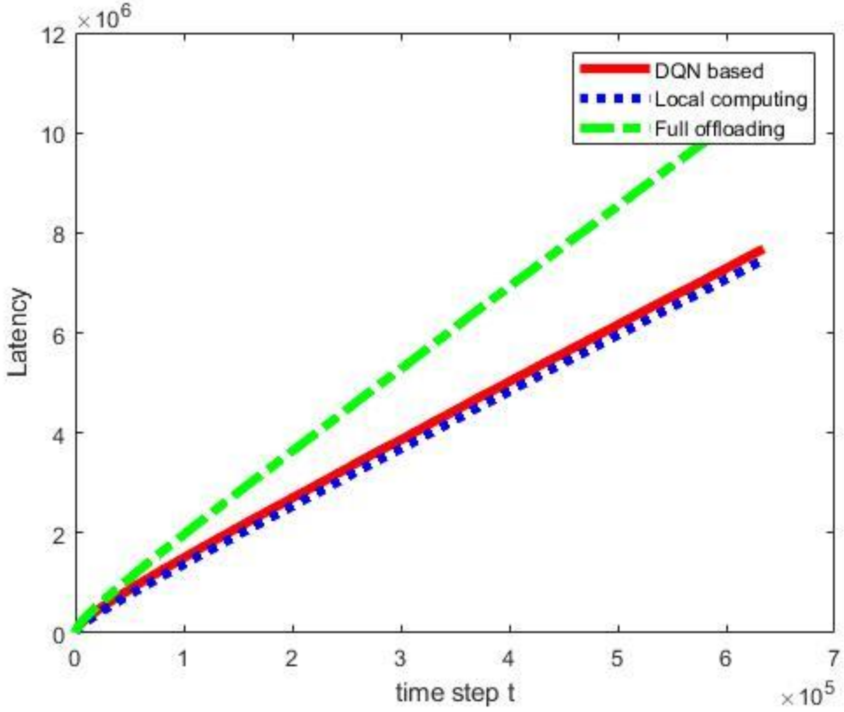


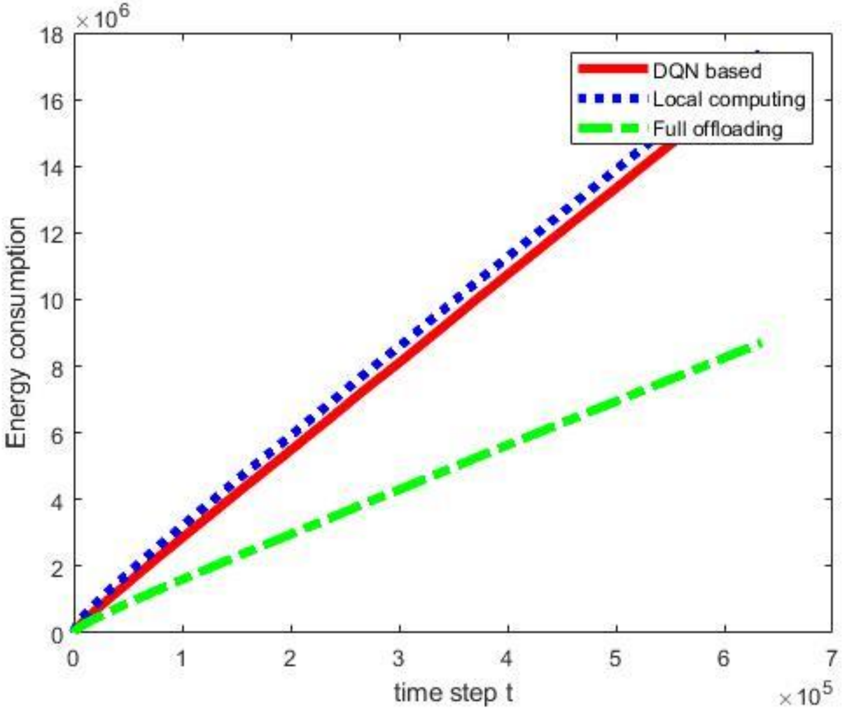
Fig. 4: Total reward versus the episode of DQN

- The total learning reward with respect to the increasing episodes

- Episode 300~500 section
- Reward doesn't increase continuously but repeats increase and decrease
- Because the task to be processed at each step is randomly selected

Experiments





Conclusion & Future work

- Conclusions
  - This paper proposes a joint dynamic video streaming and deep Q-network (DQN) based intelligent offloading method in mobile edge computing systems.
  - For utilizing video services in mobile edge networks, efficient streaming and offloading algorithms are essentially required.
  - In this paper, therefore, a new dynamic offloading algorithm with DQN is proposed.
- Future Work
  - The various performance evaluation with various settings are desired.
  - Specific streaming control algorithms can be further discussed.
    - Lyapunov optimization-based algorithms are definitely considerable.

Thank you