EasyChair Preprint

№ 780

# Deep Q-Learning Based Algorithm for Dynamic Adaptive Streaming over HTTP

Goay Fuh Yang, Wei-Tsong Lee and Hsin-Wen Wei



February 17, 2019

# DEEP Q-LEARNING BASED ALGORITHM FOR DYNAMIC ADAPTIVE STREAMING OVER HTTP

Goay Fuh Yang, Wei-Tsong Lee, Hsin-Wen Wei
Department of Electrical Engineer, Tamkang University, Tamsui, Taiwan 251.R.O.C
Goay621@hotmail.com wtlee@mail.tku.edu.tw hwwei@mail.tku.edu.tw

## ABSTRACT

As mobile technology getting more advanced, watching streaming video becomes an indispensable action in our daily life. How to maintain or provide a good quality of Experience (QoE) for the video user during bad network condition is an important and challenging issue. Therefore, we propose a Deep Q-learning based algorithm to substitute the adaptive bitrate algorithm (ABR). Our algorithm aims to learn and improve the Quality of Experience (QoE) for the video user in overall network condition. The result shows that our Deep Q-learning algorithm is able to learn and select the segment with good quality which gives the user the better (QoE).

**Keyword:** Deep Q-Learning, Quality of Experience, adaptive bitrate algorithm

## 1. INTRODUCTION

According to Cisco VNI Global IP Traffic Forecast 2017-2022[1], the internet traffic dramatically increased in recent years. This phenomenon is due to mobile technology getting more advanced and more and more people like to use mobile phone to watch video. Surfing internet video has become an indispensable action in daily life. Therefore, to maintain or improve the Quality of Experience of the video user in bad network condition has become a critical issue to be solved. In this paper, we propose a Deep Q-Learning algorithm to replace the original Adaptive bitrate control algorithm of the Dynamic Adaptive Streaming over HTTP, which is called DASH.

The Dynamic Adaptive Streaming over HTTP (DASH) [2] is an adaptive bitrate streaming technique that allows the content provider to deliver the high-quality media content through HTTP web server to the client. The media content is breaking into small file segments and part of media content information is contained in each segment. The Media Presentation file (MPD) will reallocate the segment and play in DASH player. The client has can choose the different bitrate of the segments to download. The Adaptive bitrate rule (ABR) is a technique used to adjust the quality of

the media streaming by detecting the network bandwidth and CPU capacity of Dash client. There are several types of ABR algorithms being carried out to handle some critical issue like lagging, re-buffer, and etc. There are some defective in those ABR algorithms, since most of them are model fixed. Model fixed means that they only can do with a certain limited performance which they designed. For instance, if the ABR is mainly design to play the video with a good quality, during bad network condition it will only play with high quality which might cause the rebuffer event occurs that lead to bad QoE for the video user. Hence we propose a DASH based streaming system which utilizes Deep Reinforcement learning technique to enhance the adaptive bitrate control.

Deep Reinforcement learning (DRL) is a technique that combines reinforcement learning with feed forward. Reinforcement learning is one type of machine learning and is a branch of Artificial intelligent. Markov decision process is a fundamental for DRL, DRL has an Agent that keep trial and error to get experience from the environment and finally take the action that may gain the largest reward based on the state of the environment. As the state of the environment getting more complex, the concept of deep neural network is applied to solve this situation. Hence, we get DRL that preprocess the state of the environment by backpropagation using neural network and then pass to Reinforcement learning to finish its job.

## 2. METHODOLOGY

In this section, we present the DASH-based framework used in our system. The structure of the DASH-based system is shown in Figure 1. First, the user plays the Dash player and start to watch video, the Dash player will request the Media Presentation Description(MPD) and media segment from the DASH server through HTTP protocol. While the user is watching video, the DRL Adaptation Algorithm is applied to collect the information like buffer level, available bandwidth, etc. The information will help the DRL to decide which quality of media segment to download. The DRL will choose the media segment that gains the largest reward which eventually provides a good QoE to the user. In our Dash based system, we keep most of the structure similar to original Dash framework, this give us the advantage of the original Dash framework provided. For instance, the switching quality performance can be enhanced by using the pipeline technique. Which means there are no longer needs to fill up the buffer over the desired level during switching the video quality compared to other heuristics type system.

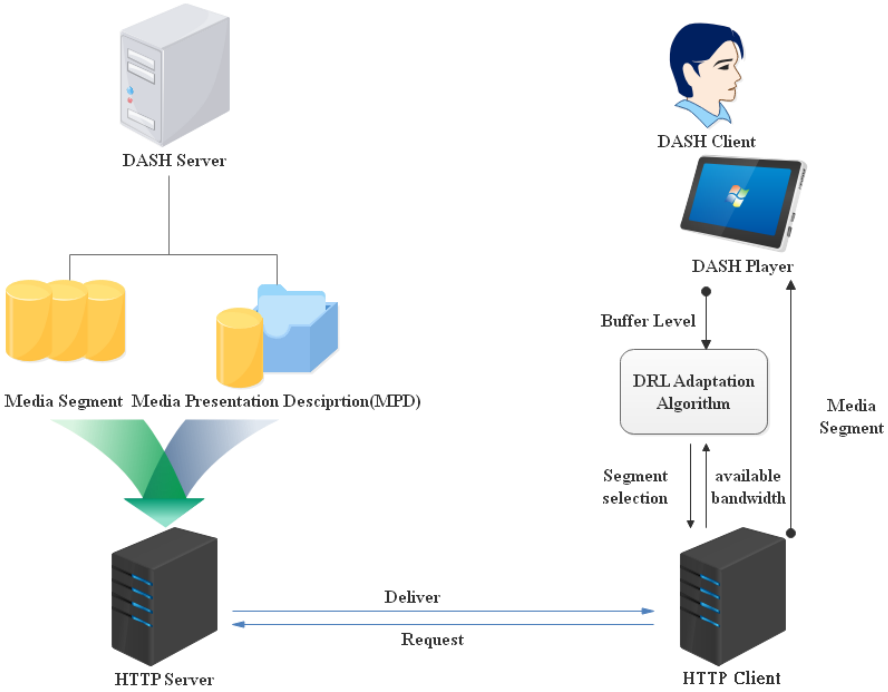


**Figure 1.** Structure of the DASH system

In this research, we carry out the concept of Deep Reinforcement Learning to adjust the video quality. The fundamental of the Deep Reinforcement learning is Markov Decision Process (MDP). According to the previous studies [3][4], the adaptive video streaming service can be modeled as MDP. A typical MDP contains action space $A$, state space $S$, and reward function $R$. The notation $A_t$ is used to indicate the action of downloading a particular quality segment of the video, where $A_t \in A$. And the reward function is defined as $r(S_t, S_{t+1}, A_t)$, where $S_t$ $and$ $S_{t+1}$ are belong to state space $S$ and are obtained from the system. Policy is a function denoted as $\delta: S \rightarrow A$ that helps the agent to decide what action to be taken based on state. Then, the action-value equation can be defined as

$$Q(S_0, A_\delta) = \sum_{S_1 \in S} P(S_0, A_\delta, S_1)[r(S_t, A_\delta, S_{t+1}) + \gamma Q(S_1, A_\delta)] \quad (1)$$

The $S_0, S_1$ indicate the one step condition transition probability of the state process $S_t$. The $\gamma \in [0,1)$ indicates the discount factor that ensures convergence the value can be set from 0 close to 1. The $P(S_0, A_\delta, S_1)$ indicates the probability of the state from $S_0$ take the action $A_\delta$ based on policy δ and move to state $S_1$. As shown in equation (1), the equation can obtain the Q-value of next state. The optimum policy can be found as

$$\delta(V) = \arg max_\delta \, Q(S_0, A_\delta) \quad (2).$$

Finally, the optimum equation can be obtained by combining equation (1) and (2) as shown in below.

$$\delta(V) = max \sum_{S_1 \in S} P(S_0, A_\delta, S_1)[r(S_t, A_\delta, S_{t+1}) + \gamma Q(S_1, A_\delta)]$$

Based on the MDP model, Watkin and Dayan have introduced Q-learning in 1992 [5]. The main concept of Q-learning is solving the MDP problem by using reinforcement learning technique. Q-learning explores the optimal reward for each state-action pair by trial and error. The common policies used in Q-learning are epsilon-greedy policy and Softmax. The Epsilon-greedy policy is a straightforward way of selecting action; it either selects random action with epsilon-greedy probability or selects the action that yields optimal reward with 1-epsilon-greedy probability. Softmax chooses the action according to the Softmax distribution of Q-value:

$$P(q_t|s_t) = \frac{e(-\frac{Q(s_t,A_t)}{\beta}}{\sum_{q \epsilon A} e(-\frac{Q(S_t,A)}{\beta}}$$

The parameter $\beta$ indicates the greediness of the algorithm. The Deep Q-learning is a RL algorithm combine with deep neural network. In our work, we use DRL to solve the problem. The Schematic diagram of DRL is shown in Figure 2. The agent obtains the Q-value from the neural network and chooses the appropriate quality segment to download as an action. Eventually, these actions taken by the agent will affect the environment and generate new state and reward for the agent.

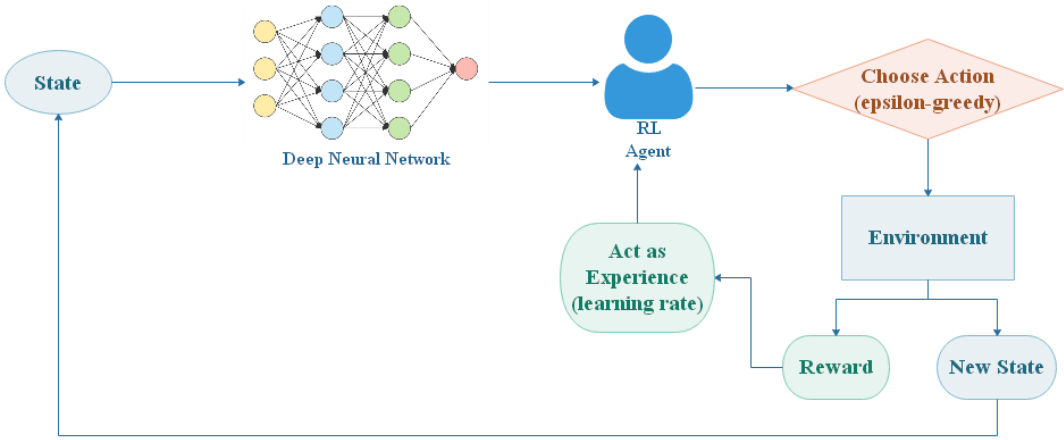


**Figure 2.** Schematic diagram of DRL

The states and rewards are the keys for the DRL to improve the QoE for the video user. Hence how to define the states and rewards is the main concern of our work. The states of environment need to be meaningful and contain enough information. Therefore, choosing what information included in the state is important. Considering that too many or few states will affect the accuracy of the RL Agent, the Buffer level $B_l$, current bandwidth $C_b$, and previous action $A_p$are included in a state $S_t = (B_l, C_b, A_t)$, since the buffer level and bandwidth have the most impact on users' experience. The action $A_t$ defined in our work is to select adequate quality segment

to download and the RL agent will choose the action based on epsilon-greedy strategy. It will choose the action that gains the most reward if the probability of 1-epsilon-greedy is high or choose the action randomly to explore another possibility as the learning experience. The definition of the reward function will directly affect the QoE of the video user. Hence, we define the reward function as the following equation.

$$reward = A_t - A_v - rb + bl + bt$$

**Table 1.** Reward Function

| Symbol | Represent | Value |
|---|---|---|
| $A_t$ | Action number | {0,1,2,3} |
| $A_v$ | Penalty for action change | {0,1,2,3} |
| $rb$ | Rebuffer event | {0,1} |
| $bl$ | Buffer level | {0,1…,24} |
| $bt$ | Buffer threshold | {-8,-7…,0} |

Table 1 shows the detail of the reward function that we defined in this paper. In the proposed reward equation, $A_t$ denotes the action number which is in range from value 0 to 4. Each value defines different action, where 0 represents the action that downloads the lowest quality segment for the player, 1 represents the action that downloads higher quality segment than 0, and so on. That is, action number 3 represents the action that download the highest quality segment. We normalize the value by divide by the highest action value that an agent can get. Hence the higher quality of the segment being chosen, the higher the positive reward gained. $A_v$ denotes the penalty for the difference between previous action and current action taken by the agent. Hence this can prevent the agent to change the video quality often and increase the stability. $rb$ denotes the rebuffer event. We define the value 1 to represent the condition that rebuffer event occurred and 0 for the normal condition. The rebuffer event is estimated by using the buffer level. When the buffer level is empty or the downloaded segment buffer is unable to fulfill the requirement of player until the next buffer segment download, then we will consider it as a rebuffer event. If the rebuffer event occurs, then there should be a negative reward, since rebuffer will introduce negative experience to users. Moreover, $bl$ denotes the buffer level. The value of buffer level is obtained from the buffer capacity of player, in which the maximum buffer level is 24 second and minimum is 0s. The higher the buffer level left the higher the value $bl$ get. Hence this will encourage the agent to get more buffer level to gain more positive reward. Finally, $bt$ denotes the threshold for the buffer level. We set the threshold value as 8 and the value $bt$ is defined by current buffer level minus 8 and we only record the negative value for $bt$, that is it will get

negative reward for the buffer level lower than 8s. This will help the agent to prevent the rebuffer event occur.

## 3. Experiment Result

In this experiment, we assume that every segment with the same quality has a constant size. This assumption is commonly used in other relevant research [6]. In a real system, there are several types of encoding technique and parameter with a segment of varying size with different qualities. However, this factor should not significantly affect the performance of the learning algorithm. Table 2 shows the parameters that used in our work. To demonstrate the performances of the proposed algorithm we simulate the video playing under the network condition as described below. The network bandwidth for the client is 10Mbps and we apply a square wave network interference to it. The peak of the square wave network interference is 5 Mbps and the interval between two peaks is 10 second. First, we conduct the baseline result under the assumption that every video segment is streamed with highest quality 1080p. In the experiment, the video is 3 minutes long, then playing the video from the start to the end is considered as one run in the simulation. Figure 3 shows the number of rebuffer event happened under different runs in the experiment. Figure 3 shows that there are 13 times of rebuffer event occurred in average of baseline result. The training phase shows that our DRL algorithm explores the environment and learning to find the best action to get the best reward. The number of the rebuffer event occurs in the training phase decrease as the reward increase. During the training phase, we convergence the epsilon-greedy proportionally and check the mean reward to determine when should we carry out for testing phase. Figure 4 shows the mean reward for the training phase in the experiment. In the training phase, the experiment is about to play 5 runs of the video which is 15min. After the $5^{th}$ run, the mean reward does not significantly improve. Hence, the testing phase is set after $5^{th}$ run by lock the epsilon-greedy to maximum greediness because we don't need to explore anymore in the testing phase. As shown in Figure 3, the number of rebuffer event in testing phase is varying between 0-2 times; this is because there is some unforeseen interrupt in a real system that might affect the performances. For instance, the size of each segment quality is not constant.

**Table 2.** Adaptation Algorithm Parameters

| Algorithm | Parameter | Value |
|---|---|---|
| DRL | Maximum buffer size | 24s |
| | Discount factor | 0.75 |
| | Policy | Epsilon-greedy |
| | Hidden Neurons | 100 |
| | Learning rate | 0.5 |

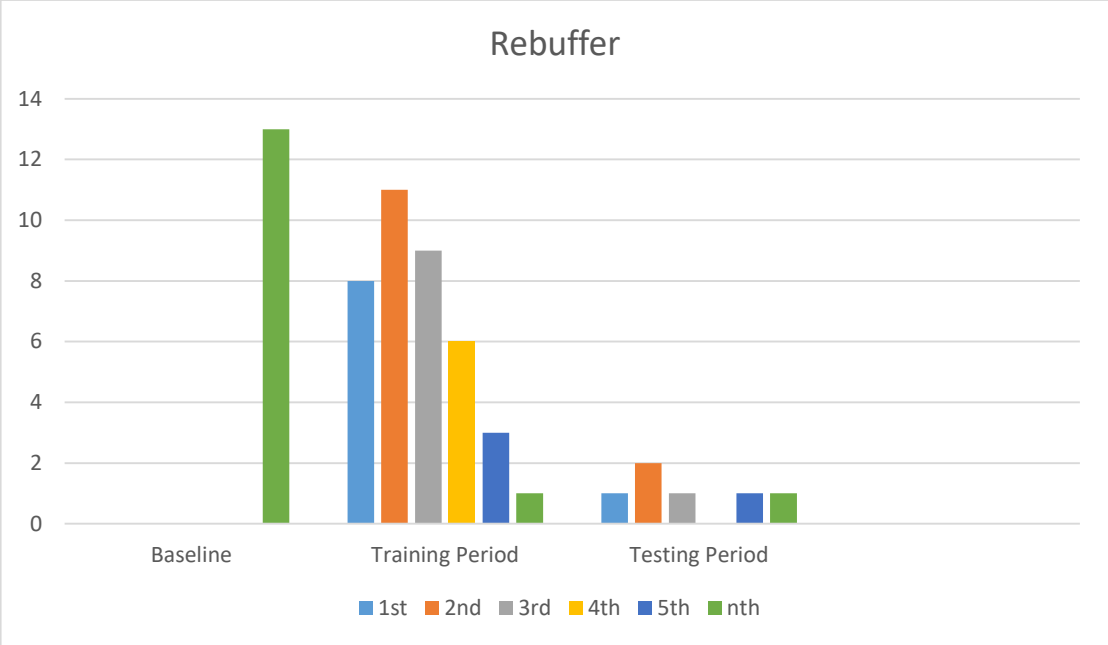


**Figure 3.** Rebuffer event

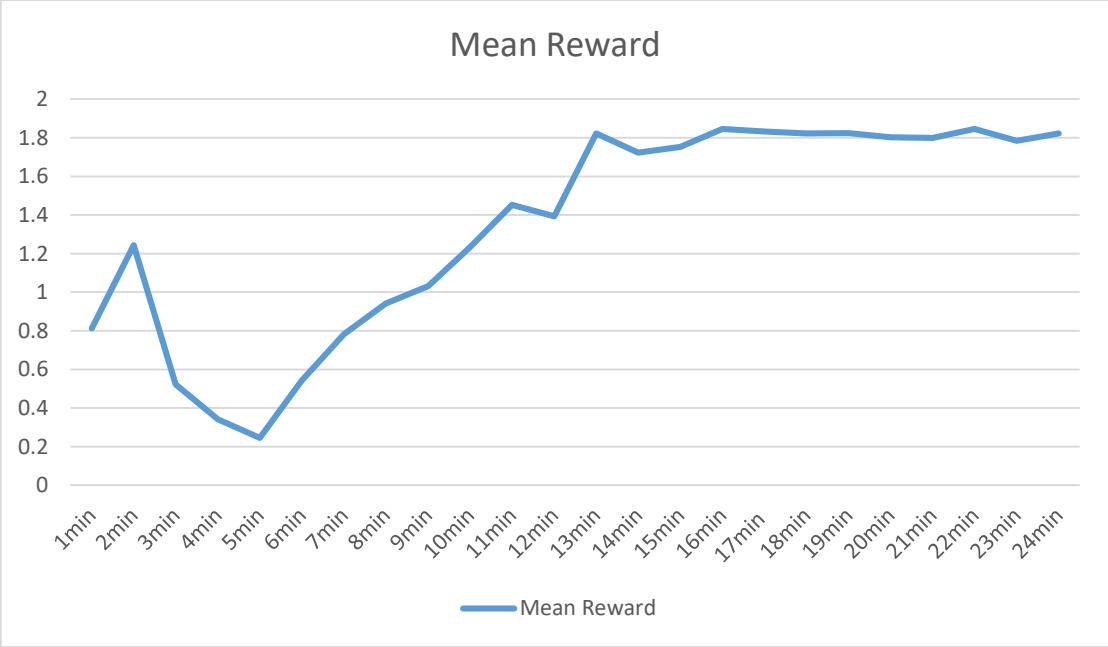


**Figure 4.** Reward for the training phase

## 4. CONCLUSION AND FUTURE WORK

We develop a DRL algorithm to improve the QoE of the video user to confront various network conditions. The number of rebuffer events has been significantly reduced in our work. In the future, we will enhance the performance of the DRL and the learning speed of the DRL.

## 5. REFERENCES


[1] Cisco,"Cisco Visual Networking Index: Forecast and Trends, 2017–2022," Cisco Public Information, 2018.

[2] Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats, ISO/IEC Standard 23009-1:2014, May 2014.

[3] R. Bellman, “A Markovian decision process,” Indiana Univ. Math. J.,vol. 6, no. 4, pp. 679–684, 1957

[4] C. Zhou, C.-W. Lin, and Z. Guo, “mDASH: A Markov decision-based rate adaptation approach for dynamic HTTP streaming,” IEEE Trans. Multimedia, vol. 18, no. 4, pp. 738–751, Apr. 2016.

[5] M. Claeys et al., “Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming,” in Proc. Adapt. Learn. Agents Workshop (ALA), St. Paul, MN, USA, May 2013, pp. 30–37.

[6] Matteo Gadaleta, Federico Chiariotti, Michele Rossi, Andrea Zanella, “D-DASH: A Deep Q-Learning Framework for DASH Video Streaming,”IEEE Transactions on Cognitive Communications and Networking