November-December 2019
ISSN: 0193-4120 Page No. 2416 - 2420

Additional DQN Extension Methods of Fast Convergence for the Optimal Policy

Young-Man Kwon[1], Gyu-Bong Lee[2], Dong-Keun Chung [3], Myung-Jae Lim[4]
[1]Department of Medical IT, Eulji University, Korea, ymkwon@eulji.ac.kr
[2]Department of Medical IT, Eulji University, Korea, 0409gb@gmail.com
[3]Department of Medical IT, Eulji University, Korea, tchung@eulji.ac.kr
[4]Department of Medical IT, Eulji University, Korea, lk04@eulji.ac.kr

Article Info
Volume 81
Page Number: 2416 - 2420
Publication Issue:
November-December 2019

Article History
Article Received: *5 March 2019*
Revised: *18 May 2019*
Accepted: *24 September 2019*
Publication: *12 December 2019*

Abstract
To improve the performance of DQN, Deep Mind proposed six extensions. In this paper, we propose two additional DQN extensions. The first extension applies batch normalization to the model of DQN and the second extension applies atrous convolution. We measured the performance of them 30 times on the Atari game Pong. We did the post-hoc-analysis because ANOVA analysis of them is significant at the confidence level of 95%. According to experimental results, we conclude that we can apply the proposed extensions to the vanilla DQN for better performance.

Keywords: *Atrous Convolution, Batch Normalization, Convolutional Neural Network, Deep Q-Network, Deep Reinforcement Learning*

1. INTRODUCTION

Reinforcement learning is one of the areas of machine learning such as supervised learning and unsupervised learning, that solves problems by interaction with the agent and the environment. The agent executes one of the valid actions and receives the next state and rewards for this from the environment. Based on this interaction, the agent learns a policy that maximizes the accumulated rewards.

Tabula method is the method to solve reinforcement learning problem by defining and updating the value of action by using a Q-table. However, it is difficult in the environment with many states or actions. Q-Network replaces it with an artificial neural network, that works as a function approximator with parameters. DQN (Deep Q-Network) uses deep neural network whose output is the action value [1], [2].

Many extensions of DQN have been studied to improve the performance of it [3]. Google DeepMind proposed six extensions: N-steps DQN, Double DQN, Prioritized Experience Replay, Categorical DQN, Noisy Networks and Dueling DQN. These extensions can be used independently or together. Rainbow DQN is a combination of these extensions.

In this paper, we propose two extensions for fast convergence of DQN, those are batch normalization and atrous convolution. We measure the performance of our extensions by using the Atari game Pong, where the player who gets 21 points first wins.

2. RELATED WORKS

2.1 Deep Q-Network

The structure of the vanilla DQN looks like Figure 1. In Q-learning using artificial neural network, the agent executes action to the environment and receives the next state and rewards from it (see the (a) part in Figure 1).

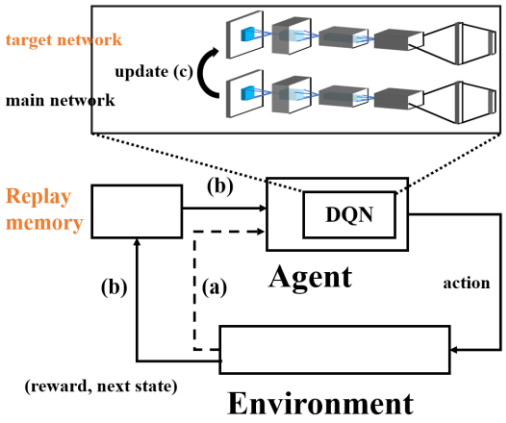

Figure 1: The structure of the vanilla DQN

November-December 2019
ISSN: 0193-4120 Page No. 2416 - 2420

In DQN, the agent is trained in the flow as (b) part of Figure 1. Deep learning algorithm such as SGD optimization assumes that training data are independent of each other, but the data of reinforcement learning are results of sequential action, so it has a strong correlation between samples. Therefore, the DQN uses the replay memory for breaking the correlation between samples (see the (b) part in Figure 1).

The replay memory stores a tuple of samples (state, action, reward, next state) during exploring or exploiting the environment. Then, the agent can obtain gradients through several samples that is sampled at random from the memory and then update the artificial neural network. Therefore, it prevents the artificial neural network from being updated to the wrong policy and helps keep learning stable.

Another problem of using Q-network is that the target value continues to change during updating. In the formula of MSE, the loss function of the vanilla Q-network is as follows.

$$MSE = (r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta) - Q(s_t, a_t, \theta))^2 \quad (1)$$

The first term is the reward r_t. It is the value received by action in the time step *t*. The second term is the maximum of Q-value after executing a_{t+1} in the next state s_{t+1}, which is the target value. The third term is the Q-value which is the value after executing the action a_t in the state s_t. When we calculate the value of second and third term using same neural network. However, training is unstable because the parameter of the target value is the same as the training parameter.

To solve this problem, DQN adds the separated target network, and it copies the weight of the main network at regular interval (see the(c) part in Figure 1). Then, the modified loss function of DQN is as follows.

$$MSE = (r_t + \gamma \max_{a} Q_{\theta^-}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t))^2 \quad (2)$$

Where, the θ^- means the parameter of the target network and the θ means the parameter of the main network.

2.2 Deep Q-Network Extensions

Google DeepMind proposed six extensions: N-steps, Double DQN, Prioritized Experience Replay, Categorical DQN, Noisy Networks and Dueling DQN.

When calculating the target value in Q-Learning, the target value is based on only the current reward. For N-steps DQN, rewards from N steps are added together and the Q function value is added only at the very end, 2-steps DQN is as in the following formula [4].

$$Q(s,a) = r_t + \gamma r_{t+1} + \gamma^2 \max_{a'} Q(s_{t+2}, a') \quad (3)$$

In the vanilla DQN, the bellman equation of target is as follows.

$$Q(s,a) = r_t + \gamma \max_{a} Q_{\theta^-}(s_{t+1}, a_{t+1}) \quad (4)$$

It has tendency to overestimate Q-value because of using max operation, which may be harmful to training performance and sometimes can lead to suboptimal policy. So, the bellman equation of Double DQN is modified as followings [5]. It solves overestimation completely.

$$Q(s,a) = r_t + \gamma \max_{a} Q_{\theta^-}(s_{t+1}, \arg\max_{a} Q_\theta(s_{t+1}, a)) \quad (5)$$

There is also extension that improve performance by modifying the sampling method. The DQN samples the tuple of the replay memory uniformly. In Prioritized Experience Replay, for the efficiency of the sample, it prioritized the samples according to the training loss [6]. The important samples are sampled more frequently, and therefore the agent learn more efficiently.

Categorical DQN predicts value as distribution rather than a single scalar value. Therefore, it uses distribution instead of value Q of bellman equation [7].

Noisy Networks and Dueling DQN are the extension that modifies model of the network. Noisy Networks adds the noise to the network weights to make exploration more efficient [8].
In Dueling DQN, the value and advantage are calculated separately and then combined only at the final layer in to the Q-value [9]. The benefit of it is to generalize learning across without imposing any change to underlying reinforcement learning algorithm. The network architecture of the Dueling DQN looks like Figure 2.

November-December 2019
ISSN: 0193-4120 Page No. 2416 - 2420

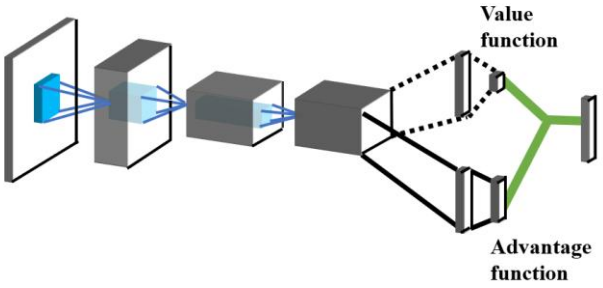

Figure 2: The architecture of the Dueling DQN model

Capitalize only the first word in a paper title, except for proper nouns and element symbols. For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [8].

2.2 Batch Normalization

Batch normalization is a technique used to improve the stability of the output of operation [10]. The idea of batch normalization is to adjust the activation values on each layer to be distributed appropriately. BN normalizes the mini-batch data by standard normal distribution (mean is zero and variance is one). Suppose we denote the mini-batch data as B = { x_1, x_2, …, x_2 }(m means the number of input data). First, we obtain the mean and variance of the data set.

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{6}$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{7}$$

Second, we normalize it for the average 0, and the variance 1.

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{8}$$

In the above formula, we used a small value epsilon whose role is to prevent the division by zero. By applying this process, data distribution can be made like normal distribution. In addition, scale and shift factor are applied to this normalized data. These factors control the gradient of normal distribution.

$$\hat{y}_i \leftarrow \gamma \hat{x}_i + \beta \tag{9}$$

Batch normalization can be applied to convolutional neural network. The following is the formula of convolutional neural network.

$$z = g(Wu + b) \tag{10}$$

Where, W and b are parameters of the network model to be learned. $g(\cdot)$ is the activation function such as ReLU. Batch normalization is applied between convolution layer and activation function, so the formula looks like following formula.

$$z = g(BN(Wu)) \tag{11}$$

Because of the shift factor of batch normalization, bias b can be ignored. By applying batch normalization, the advantage is known that we can quickly learn without being greatly affected by the initial weight.

2.3 Atrous Convolution

Atrous Convolution [11] is a method of increasing the receptive field by adding holes inside the filter. Atrous rate defines the interval of the filter to add hole.

For example, 3x3 filter with atrous rate 2 has the same view as 5x5 filter using only nine parameters in Figure 3. The dark part has only non-zero values.

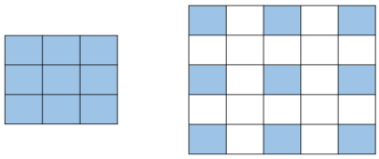

Figure 3: 3x3 filter and 3x3 filter with atrous rate 2

3. PROPOSED MODELS

In this paper, we compare the performance with the vanilla DQN and our extensions of it. The architecture of the vanilla DQN is as Figure 4.

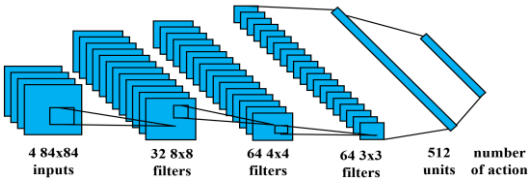

Figure 4: The vanilla DQN model of Deep Mind

We cropped 84x84 region of the Atari game screen image and made values between 0 and 1. The first hidden layer convolves 32 8x8 filters with stride 4 with the input image and applies a rectifier nonlinearity. The second hidden layer convolves 64 4 x 4 filters with stride 2, again followed by a rectifier nonlinearity. This is followed by a third convolutional layer that convolves 64 filters of 3 x 3 with stride 1 followed by a rectifier. The final hidden layer is

November-December 2019
ISSN: 0193-4120 Page No. 2416 - 2420

fully-connected and consists of 512 rectifier units. The output layer is a fully-connected linear layer with a single output for each valid action.

As a first extension of the vanilla DQN, we add batch normalization layer between the convolution layer and the activation function. So, our proposed network looks like Figure 5.

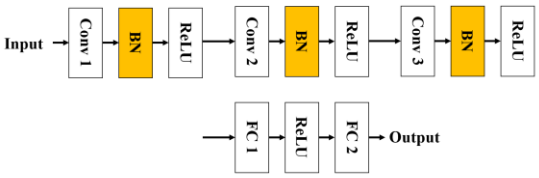

Figure 5: The architecture of the model using batch normalization

As you can see, BN normalizes output values of each convolution layer. It makes the network more stable. Also, it can be less affected by initial values and prevents problems such as vanishing and exploding gradient.

In the second extension, we replace the second convolution layer of DQN with an atrous convolution layer. 64 4x4 filters with stride 2 were modified to 64 4x4 filters with stride 1, adding an atrous rate 2. The stride was adjusted to 1 because the atrous rate was set to 2. It has a wider receptive field and learns features of it. The second proposed model looks like Figure 6.

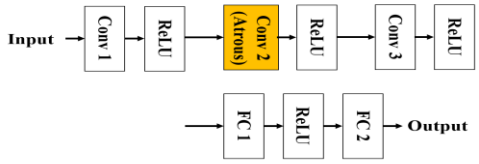

Figure 6: The architecture of the model using atrous convolution

4. EXPERIMENTAL RESULTS

We used Pong Game in Atari 2600 to measure the performance of proposed extensions. A Pong Game is a two-dimensional game imitating ping-pong, in which a player who takes 21 points first wins by passing the ball over the opponent. The hyperparameters of the model are shown in the Table I.

Table 1. Hyperparameter list of the model

Hyperparameter	Value	Description
Batch Size	32	Number of samples to be sampled from the memory
Replay memory size	100,000	Size of experience replay buffer
Target network update frequency	1,000	The frequency with which the target network is updated
Discount factor	0.99	Discount factor used in the Q-learning update
Learning rate	1e-4	The learning rate used by Adam optimizer
Replay start size	10,000	Random policy is run for this number of frames before learning starts

We experimented three models on the same environment and how many frames are required until the mean reward for last 100 episodes was 17 points (80% of 21 points) or more. We compared total reward per episode (shown in the Figure 7) and how many frames are required by average to reach 17 points in the recent 100 episodes (shown in the Figure 8. It means moving average).

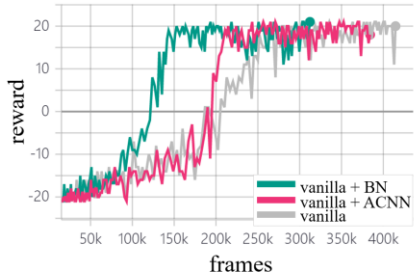

Figure 7: A sample plot of total reward per episode

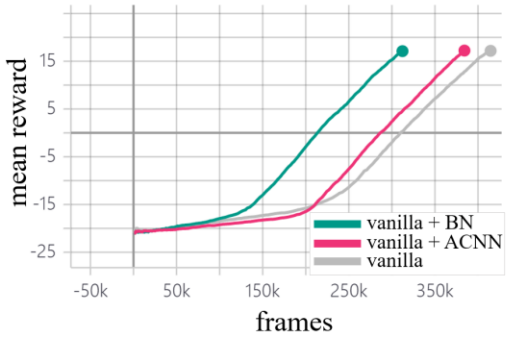

Figure 8: A sample plot of mean reward for 100 episodes

As you can see in Figure 7 and Figure 8, proposed extensions (vanilla + BN, vanilla + ACNN) shows better performance than the vanilla DQN. We made experiments 30 times by changing seed value for proposed extensions so as to be significant. The number of frames to reach 17 points on average are in the table (see Table 2). The boxplot of result is shown in Figure 9 [13-15].

Table 2. Average of frames to reach 17 points

November-December 2019
ISSN: 0193-4120 Page No. 2416 - 2420

Model	Average number of frames
Vanilla DQN	454,889
Vanilla DQN + BN	364385
Vanilla DQN + ACNN	404,092

We did ANOVA analysis to be significant between models. As a result, it was significant. So, we did post-hoc-analysis. The result between BN model and the vanilla DQN model ($p=1e-07 < 0.05$), the result between ACNN model and the vanilla DQN ($p=6e-03 < 0.05$) and the result between BN model and ACNN model ($p=2e-03 < 0.05$) showed significant difference. As a result, we conclude that we can apply the proposed extensions to the vanilla DQN for better performance.

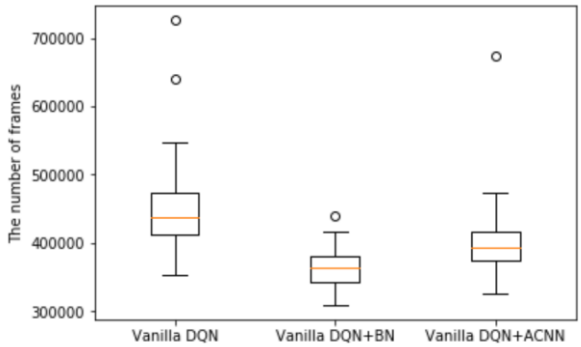

Figure 9: Boxplot of the required frames to reach 17 points on average

5. CONCLUSION

To improve the performance of DQN, Deep Mind proposed six extensions: N-steps, Double DQN, Prioritized Experience Replay, Categorical DQN, Noisy Networks and Dueling DQN. Those extensions showed great performance in Atari games.

In this paper, we additionally proposed two extensions: The first extension applies batch normalization and the second extension uses atrous convolution. We measured how many frames were required until the mean reward for 100 episodes reached 17 points. We did ANOVA analysis and post-hoc-analysis, as a result, the proposed models had significant differences from the vanilla model($p<0.05$). We conclude that we can apply the proposed extensions to the vanilla DQN for better performance.

In the future, further research is needed to confirm whether the combination of proposed extensions and existing extensions shows better performance.

ACKNOWLEDGMENT

This work was supported by R.O.K. National Research Foundation under grant NRF-2017R1D1A1B03036372 in 2019.

REFERENCES

1. V. Mnih et al. **Human-level control through deep reinforcement learning**, *Nature 518*, vol. 7540, pp. 529-533, 2015.
2. V. Mnih, et al. ***Playing Atari with deep reinforcement learning***, 2013.
3. M. Hessel et al. **Rainbow: Combining improvements in deep reinforcement learning**, *Proc. AAAI*, pp. 3215-3222, 2018.
4. R. S. Sutton. **Learning to predict by methods of temporal differences**, *Machine Learning*, vol. 3, pp. 9-44, 1988.
5. H. van Hasselt, A. Guez, D. Silver. **Deep reinforcement learning with double Q-learning**, *Proc. Association for the Advancement of Artificial Intelligence*, pp. 2094-2100, 2016.
6. T. Schaul, J. Quan, I. Antonoglou, D. Silver. **Prioritized Experience Replay**, *Proc. Int. Conf. Learning Representations*, 2015.
7. M. Bellemare, W. Dabney, R. Munos. **A Distributional Perspective on Reinforcement Learning**, *ICML*, 2017.
8. M. Fortunato et al. **Noisy networks for exploration**, *CoRR*, 2017.
9. Z. Wang, N. de Freitas, M. Lanctot. **Dueling network architectures for deep reinforcement learning**, *Proc. Int. Conf. Learning Representations*, 2016.
10. S. Ioffe, C. Szegedy. **Batch normalization: Accelerating deep network training by reducing internal covariate shift**, *Proc. Int. Conf. Mach. Learn.*, pp. 448-456, 2015.
11. P. Wang et al. ***Understanding convolution for semantic segmentation***, 2017.
12. Maxim Lapan. ***Deep Reinforcement Learning Hands-on*, Packt Publishing**, 2018.
13. Jabarullah, N.H. (2019) Production of olefins from syngas over Al_2O_3 supported Ni and Cu nano-catalysts, Petroleum Science and Technology, 37 (4), 382 – 385.
14. Hussain, H.I., Kamarudin, F., Thaker, H.M.T. & Salem, M.A. (2019) Artificial Neural Network to Model Managerial Timing Decision: Non-Linear Evidence of Deviation from Target Leverage, International Journal of Computational Intelligence Systems, 12 (2), 1282-1294.
15. Aziz, A. R., Sumantoro, I. B., & Maria, D. (2019). Total Quality Management of micro, small and medium enterprises (MSMES), and the impact to organizational culture and performance: emerging country case. Polish Journal of Management Studies, 19 (1), 32-45.