Paper Reading No.16

Rainbow: Combining Improvements in Deep Reinforcement Learning

Sheng Lian

October 2019

1 Brief Paper Intro

- ***Paper ref:*** AAAI 2018, http://arxiv.org/abs/1710.02298
- ***Authors:*** Matteo Hessel et al, from DeepMind
- ***Paper summary:*** Several independent improvements to the DQN algorithm have been made by the RL community. This paper examines six extensions to the DQN algorithm and empirically studies. Also, detailed ablation experiments have been researched. their combination.
- ***Reading motivation:*** This article is somewhat similar to the summary of DQN.

2 Extensions to DQN

DQN has shown it's effectiveness on many tasks, but several limitations of this algorithm are now known, and many extensions have been proposed. Here we will briefly introduce DQN and it's six representative extensions that address distinct concerns.

Original DQN It's unrealistic to use Q-Table to store the value of each state action pair in Q-learning when the state and action space are too large. So, DQN is proposed to obtain optimal Q value by updating the neural network.

In RL, under a given policy π, the true value of an action a in a state s is

$$Q_\pi(s,a) \equiv \mathbb{E}\left[R_1 + \gamma R_2 + \ldots | S_0 = s, A_0 = a, \pi\right] \tag{1}$$

For learning a parameterized value function $Q\left(s, a; \boldsymbol{\theta}_t\right)$. The parameter θ can be updated by

$$\theta_{t+1} = \theta_t + \alpha \left(Y_t^{\mathrm{Q}} - Q\left(S_t, A_t; \boldsymbol{\theta}_t\right)\right) \nabla_{\boldsymbol{\theta}_t} Q\left(S_t, A_t; \boldsymbol{\theta}_t\right) \tag{2}$$

and the Y_t^Q is defined as

$$Y_t^{\mathrm{Q}} \equiv R_{t+1} + \gamma \max_a Q\left(S_{t+1}, a; \boldsymbol{\theta}_t\right) \tag{3}$$

In **DQN** situation, DQN introduced target network and experience replay to training, where target network's parameter θ^- are copied from online network's θ periodically. The formulation goes as

$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max_a Q\left(S_{t+1}, a; \boldsymbol{\theta}_t^-\right) \tag{4}$$

Double DQN [5] addresses an overestimation bias of Q-learning by decoupling selection and evaluation of the bootstrap action. In Double Q-learning, two value functions are learned to update value functions. θ is used for choosing the greedy policy and the θ' is used for determining the value. The Double Q-learning's error goes as

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \operatorname{argmax} Q\left(S_{t+1}, a; \boldsymbol{\theta}_t\right); \boldsymbol{\theta}'_t\right) \tag{5}$$

It's easy to migrate double Q-learning to DQN. DDQN [5] propose to choose the greedy policy according to the online network, but using the target network to estimate its value. So the target Y is replaced by

$$Y_t^{\text{DoubleDON}} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a}{\operatorname{argmax}} Q\left(S_{t+1}, a; \boldsymbol{\theta}_t\right), \boldsymbol{\theta}_t^{-}\right) \tag{6}$$

Here, the target network θ_t^- for the evaluation of the current greedy policy, while the θ_t is used for choosing policy.

Dueling DQN [6] helps to generalize across actions by separately representing state values and action advantages.

In the original DQN, the neural network directly outputs the Q value of each action, and the Q value of each action of Dueling DQN is determined by the state value V and the advantage function A, that is, Q = V + A. So, the calculating of Q value is as follows:

$$q_\theta(s, a) = v_\eta\left(f_\xi(s)\right) + a_\psi\left(f_\xi(s), a\right) - \frac{\sum_{a'} a_\psi\left(f_\xi(s), a'\right)}{N_{\text{actions}}} \tag{7}$$

Here, the advantage A is it: For a particular state, the difference between the value that can be obtained by an action and the average value that can be obtained by that state.

The difference between DQN and Dueling DQN is indicated in Figure 1.

Prioritized experience replay [4] improves data efficiency by replaying more often transitions from which there is more to learn. In the traditional

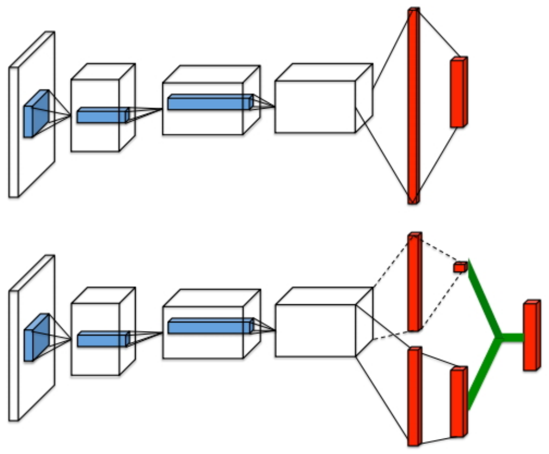

Figure 1: A popular single stream Q-network (top) and the dueling Q-network (bottom).

DQN experience pool, the selection of batch data for training is random, without considering the sample priority relationship. But in fact the value of different samples is different, this paper give each sample a priority and sample according to the priority of the sample.

Multi-step Learning as used in A3C [3], is an effective way to boost the learning speed. The original DQN uses the current instant reward R and the next moment's estimated value as the target value. In the early stage when the model parameter is not steady, the target value also face large deviation. So the learning speed is relatively slow. A3C shifts the bias-variance trade-off and helps to propagate newly observed rewards faster to earlier visited states.

Distributional DQN In DQN, the network's output is the expected estimate of the state-action value Q. This expectation actually ignores a lot of information. Using only the expected value, we can not see the risks behind the action. Distributional Q-learning [1] learns a categorical distribution of discounted returns, instead of estimating the mean.

Noisy Nets [2] solve the problem on limitations of exploring using ϵ-greedy policies. NoisyNet is another policy for adding agent's ability of exploration. This method increases the exploration ability of the model by adding noise to the model's parameters, which goes as

$$y = (b + \mathbf{W}x) + \big(b_{noisy} \odot \epsilon^b + (\mathbf{W}_{noisy} \odot \epsilon^w)\, x\big) \tag{8}$$

3 The Integrated Agent: Rainbow

This paper find that the above mentioned methods are indeed largely complementary, and integrate all the aforementioned six components into a single integrated agent, called Rainbow. For fusing the above mentioned algorithm, the loss function was first changed according to Multi-step learning, double DQN and Distributional RL. Also, this paper combine the multi-step distributional loss with double Q-learning by using the greedy action in S_{t+n} selected according to the online network as the bootstrap action a^*_{t+n}, and evaluating such action using the target network. Also, this paper modified the model's architecture according to dueling network with return distributions. Then they replace all linear layers with their noisy equivalent.

4 Experiments

See Fig 2, this paper compare the integrated agent (rainbow colored) to DQN (grey) and six published baselines. Rainbow shows it's superior performance.

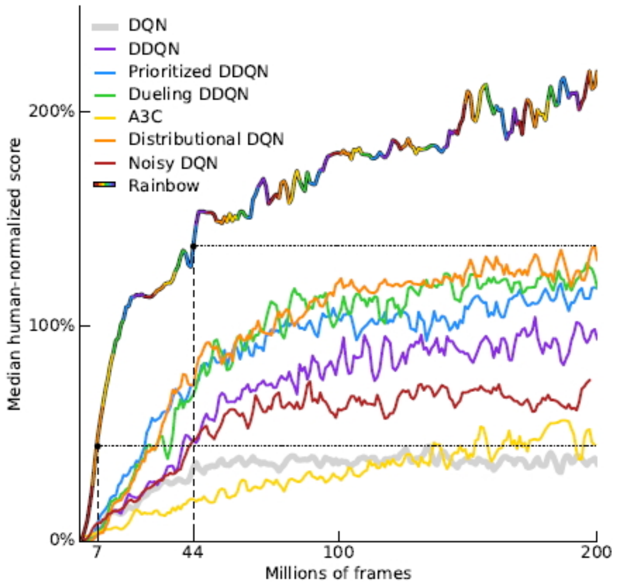

Figure 2: Median human-normalized performance across 57 Atari games.

Also, as an AAAI paper, detailed ablation study are conducted. Here we only show Fig. 3. This figure shows the effectiveness of different extension to DQN.

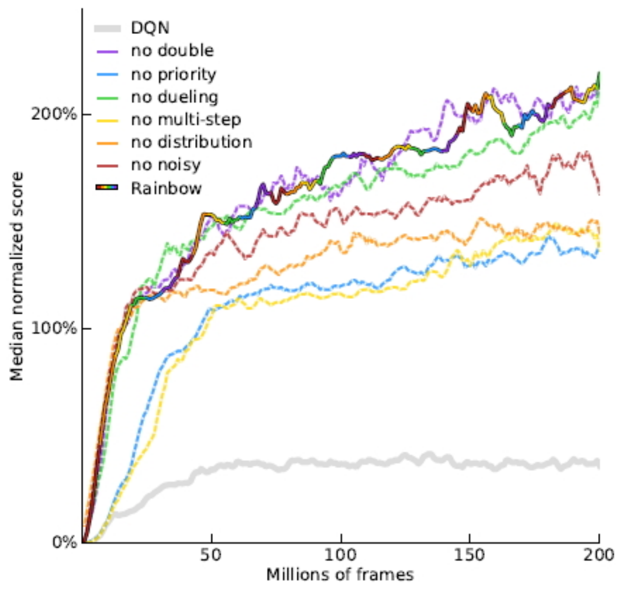

Figure 3: comparison of Rainbow to DQN (gray) and to six different ablations (dashed lines).

5 Conclusion

This paper covers too much content that I didn't understand too much. Many issues in this note is based on some blogs. I hope to have new gains when I read this paper again next time.

References

[1] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org, 2017.

[2] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[3] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[4] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[5] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[6] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.