To the Rainbow

Wonseok Jung

Rainbow: Combining Improvements in Deep Reinforcement Learning

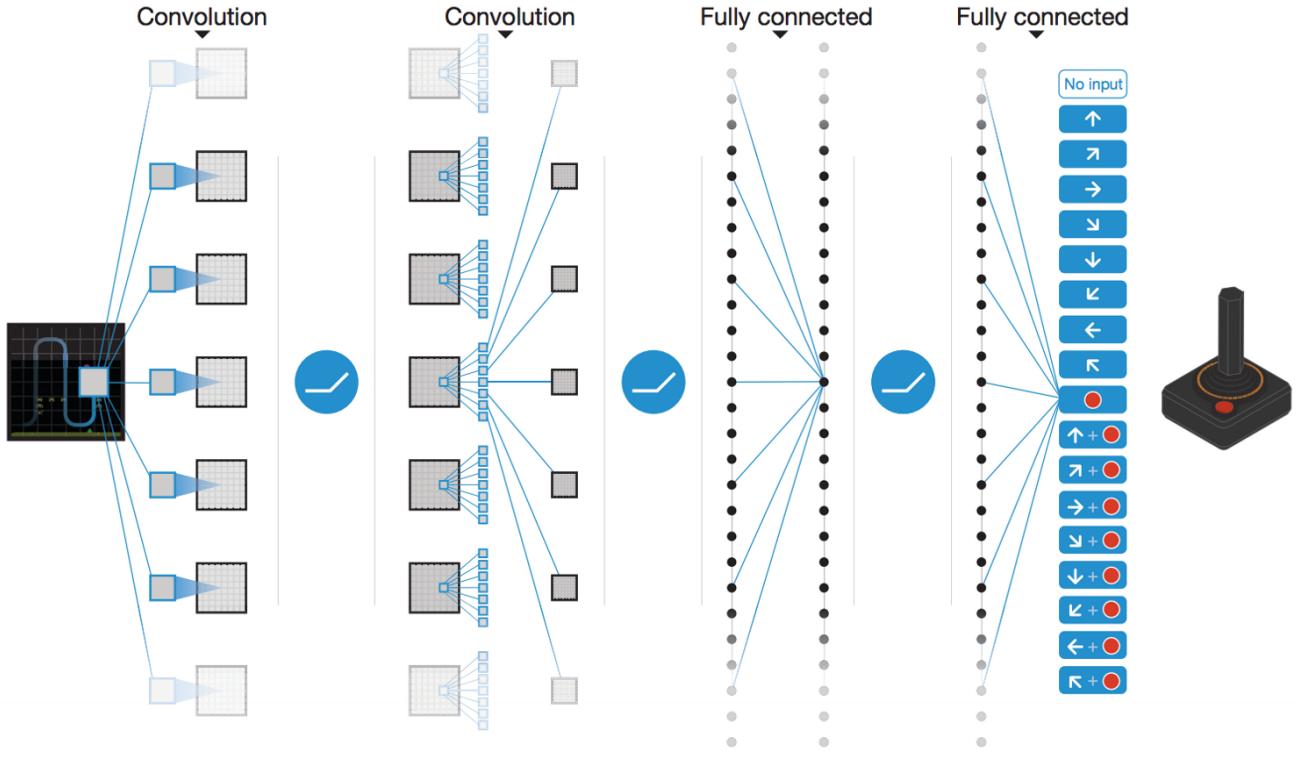
Matteo Hessel DeepMind
Joseph Modayil DeepMind
Hado van Hasselt DeepMind
Tom Schaul DeepMind
Georg Ostrovski DeepMind

Will Dabney DeepMind
Dan Horgan DeepMind
Bilal Piot DeepMind
Mohammad Azar DeepMind
David Silver DeepMind

1. Abstract

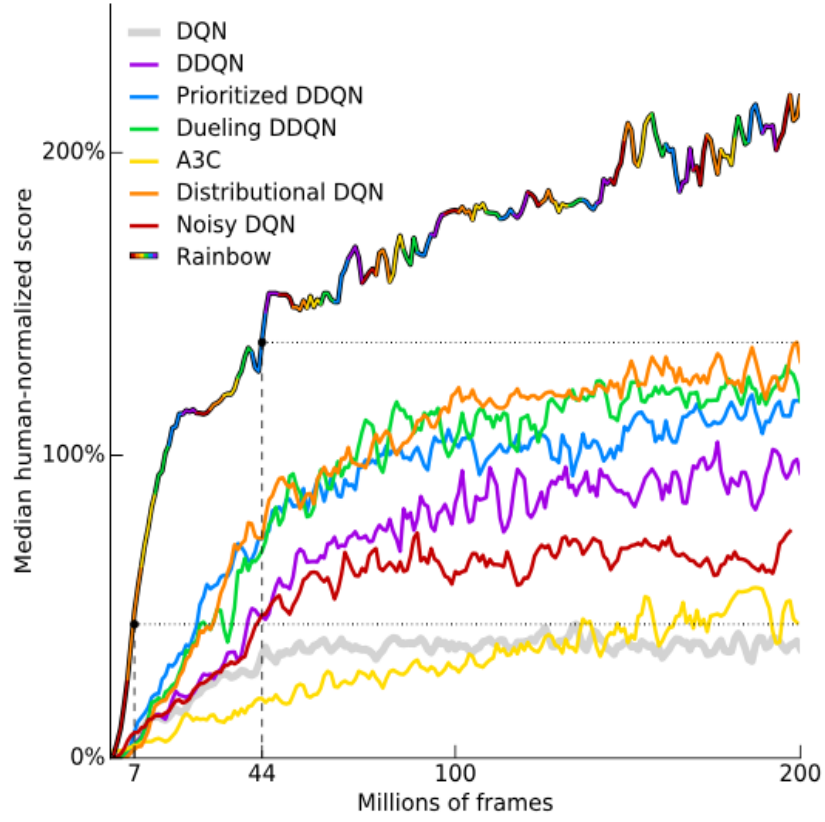
Six extensions to the DQN algorithm and empirically studies their combination

2.Introduction



강화학습에서 최근 복잡한 sequential decision-making problems을 해결한 것은 Deep Q-Networks algorithms으로 부터 시작하였다.

2.2 Performance



57개의 Atari game에서 DQN의 성능비교를 해본 결과, Rainbow DQN의 performance가 가장 좋았다.

3. Background

- 강화학습에서 agent는 reward를 maximize하기위한 action을 선택한다.
- 각 time step $t = 0, 1, 2, \dots$ 마다 environemt는 state S_t를 제공한다.
- Agent는 action A_t를 선택한다.
- Environment는 A_t 받고 Reward R_{t+1}, 다음 state S_{t+1}을 제공한다.

3.1 Agent and Evironment

- 이런 interaction을 Markov Decision Process 혹은 MDP라고 한다.
- 이 MDP는 $< S, A, T, R, \gamma >$ 의 Tuple이다.
- S : finitie set of states
- A : finite set of actions
- $T(s, a, s')$: transition function ,
  $P[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $r(s, a)$: reward function, $E[R_{t+1} \mid S_t = s, A_t = a]$
- $\gamma \in [0, 1]$: discount factor

3.1 Agent and Environment

- Policy π : probability distribution over actions for each States
- Agent는 주어진 polcy π를 따라 action을 선택한다.
- Agent는 action을 선택하며 State마다 reward를 collect 한다.

$$G_t = \sum_{k=0}^{\infty} \gamma_t^{(k)} R_{t+k+1}$$

- Agent는 위의 discounted return을 최대화 하기 위한 policy를 찾는다.

3.1 Value-based reinforcement learning

- Agent learns an estimate of the expected discounted return or value
- Given state : $v^{\pi}(s) = E_{\pi}[G_t \mid S_t = s]$
- Given state-action pair : $q^{\pi}(s, a) = E_{\pi}[G_t \mid S_t = s, A_t = a]$

3.1 Obtain new policy from state-action value function

- action values에 대하여 $\epsilon - greedily$ 하게 action을 선택한다.
- $greedy$ action : 가장 높은 value를 받는 action을 선택 , $(1 - \epsilon)$
- non $greedy$ action : ϵ의 확률로 action을 random하게 선택한다.
- 위의 policy로 인해 agent는 explortion을 하며 새로운 policy를 찾는다.

3.2 Deep reinforcement learning and DQN

- DQN : deep network와 reinforcement learning이 결합한 알고리즘
- CNN을 이용하여 state S_t의 action value를 approximation
- Agent는 $\epsilon - greedily$ 하게 action을 선택
- Transition $(S_t, A_t, R_{t+1}, \gamma_{t+1}, S_{t+1})$을 replay memory buffer에 추가

3.2 Deep reinforcement learning and DQN

- Loss를 minimize한다.
  - $(R_{t+1} + \gamma_{t+1} max_{a'} q_{\overline{\theta}}(S_{t+1}, a^{'}) - q_{\theta}(S_t, A_t))^2$
  - time step t는 replay memory에서 randomly picked
  - θ : Online network의 parameter(Back-propagated only into parameter θ)
  - $\overline{\theta}$: Target Network의 parameter(Periodic copy of the online network)

4. Extension to DQN

- Double DQN
- Prioritized experience replay
- Dueling network architecture
- Multi-step bootrap targets
- Disributional Q-learning
- Noisy DQN

4.1 Double DQN

- Q-learning은 maximization하는 부분때문에 overestimation 문제 발생
- 이로 인해 Learning 효율이 떨어진다.
- Double Q-learning : Target과 evaluation을 분리
    i. Target : maximization performed for the bootstrap
    ii. selection of the action from its evaluation
- DQN과 combine
  $(R_{t+1} + \gamma_{t+1} q_{\bar{\theta}}(S_{t+1}, argmax_{a'} q(St) - q_\theta(S_t, A_t))^2$

4.2 Prioritized replay

- DQN은 replay buffer에서 uniform하게 sampling하여 학습한다.
- 학습이 더 필요한 transitions을 sample 하는 것이 더 이상적인 방법일 것이다.
- Prioritized experience replay samples transitions with probability p_t relative to the last encountered absolute TD error

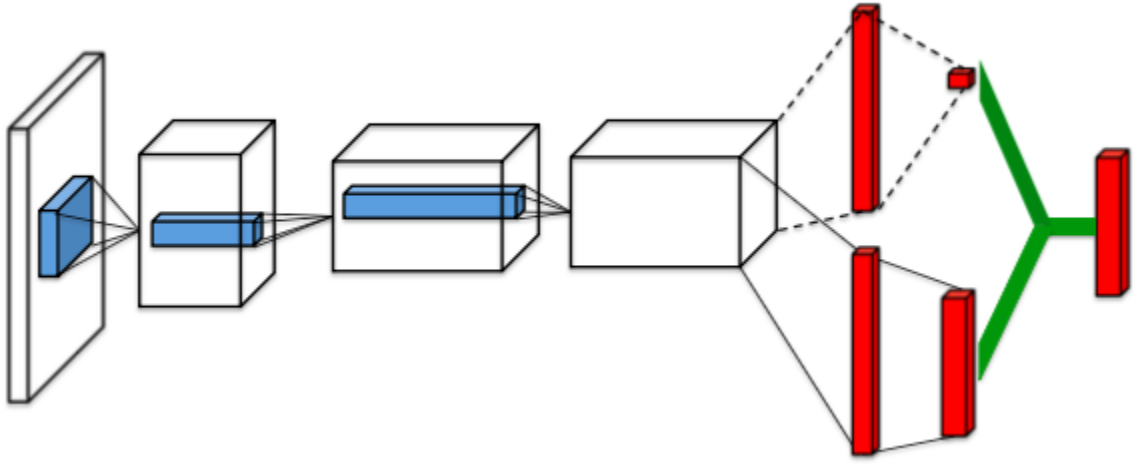
$$p_t \propto \left| R_{t+1} + \gamma_{t+1} max_{a'} q_{\bar{\theta}}(S_{t+1}, a') - q_{theta}(S_t, A_t) \right|^{\omega}$$

4.3 Dueling networks

- Dueling network는 두 computation으로 줄기를 나눈다 : Value , Advanage

$$q_\theta(s,a) = v_\eta(f_\xi(s)) + a_\psi(f_\xi, a) - \frac{\sum_{a'} a_\psi(f_\xi(s)), a^{'})}{a}$$

- ξ, η, ψ : encoder f_ξ, value v_η, advantage 의 share parameter
- $\theta = (\xi, \eta, \psi)$: concatenation



4.4 Multi-step learning

- Q-learning은 하나의 reward를 받고, bootstrap을 위해 다음 step에서 greedy action을 한다.
- 대신, multi-step을 사용 하여 n-step 다음을 bootstrap할 수 있다.

$$R_t^{(n)} \equiv \sum_{k=0}^{n-1} \gamma_t^{(k)} R_{t+k+1}$$

4.4 Multi-step learning

- DQN에서 Multi-step의 Loss는 다음과 같이 정의할 수 있다.

$$(R_t^{(n)} + \gamma_t^{(n)} max_{a'} q_{\bar{\theta}}(S_{t+n}, a^{'}) - q_\theta(S_t, A_t))^2$$

- Multi-step targets with suitably tuned n often lead to faster learning(Sutton and Barto 1988)

4.5 Distributional RL - Idea (Monopoly game)

- 상대방이 소유한 땅에 도착 : $ 2000의 penalty
- 무사히 통과하였을때 : $200 의 reward
- Reinforcement learnig 관점에서 Expect reward를 계산

$$ER(x) = \frac{35}{36} \times 200 - \frac{1}{36} \times 2000 \approx 139$$

4.5 Idea - Monopoly

- Reinforcement learning에서는 대부분 위의 예보다 더 복잡하다.

$$R(x_0) + \gamma R(x_1) + \gamma^2 R(x_2) + \gamma^3 R(x_3)...$$

- Sum of discounted reward

$$V^{\pi}(x) = E_{p^{\pi}}[\sum_{t=0}^{\infty} \gamma^t R(x_t) \mid x_0 = x]$$

$$= ER(x) + \gamma E_{x' \ P^{\pi}} V^{\pi}(x^{'})$$

4.5 Idea

- next state의 action을 bootstrap + r 과 current state 의 value를 minize 시키는 것이 아닌
- 다음 step 의 distribution과 current state의 distribution을 minimize 하자
- Bellman equation

$$Q(s,a) = ER(s,a,s^{'}) + \gamma EQ(s',a')$$

- Distributional Bellman equation

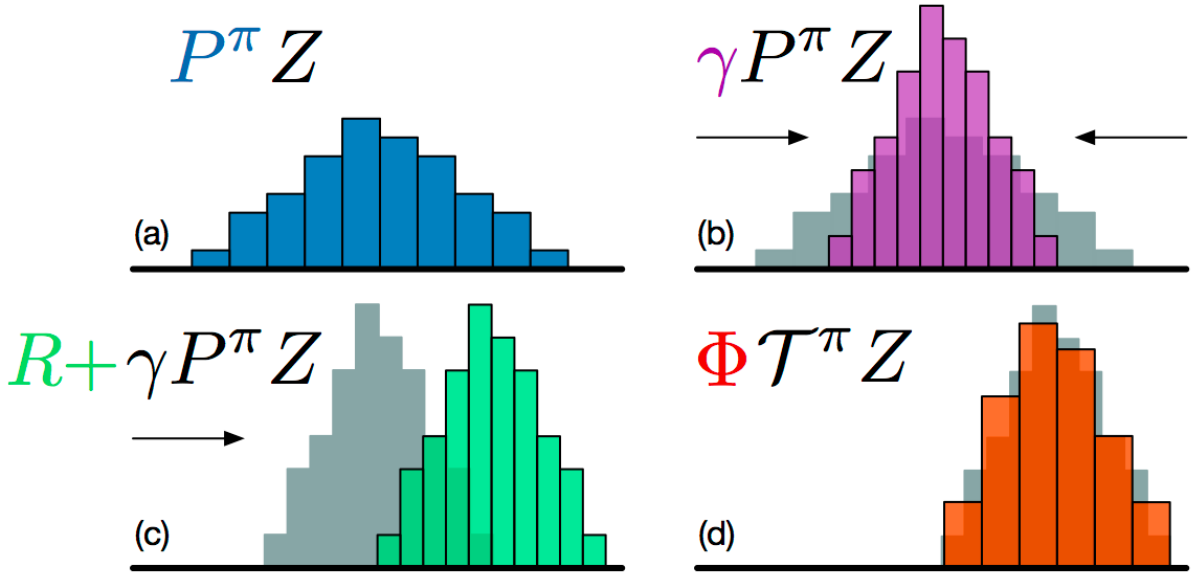
$$Z(s,a) = R(s,a,S') + \gamma Z(S',A')$$

4.5 New Algorithm - C51

- next state의 모든 distribution을 가져온다.
- 그 distribution을 backup 하여 current guess 로 사용한다.

4.5 The C51 Algorithm

- C51은 각 Q-value의 softmax 를 return한다.
- Wasserstein distance 를 minimize한다.
  (참고: https://mtomassoli.github.io/2017/12/08/distributional_rl/)



4.6 Noisy Nets

- $\epsilon - greedy$의 exploration 방법은 특정 환경에서 한계를 보인다.
- Noisy Nets을 사용하자
- 참고 : https://wonseokjung.github.io//reinforcementlearning/update/RL-Totherb-2/

$$y = (b + Wx) + (b_{noisy} \odot \epsilon^b + (W_{noisy} \odot \epsilon^w)x)$$

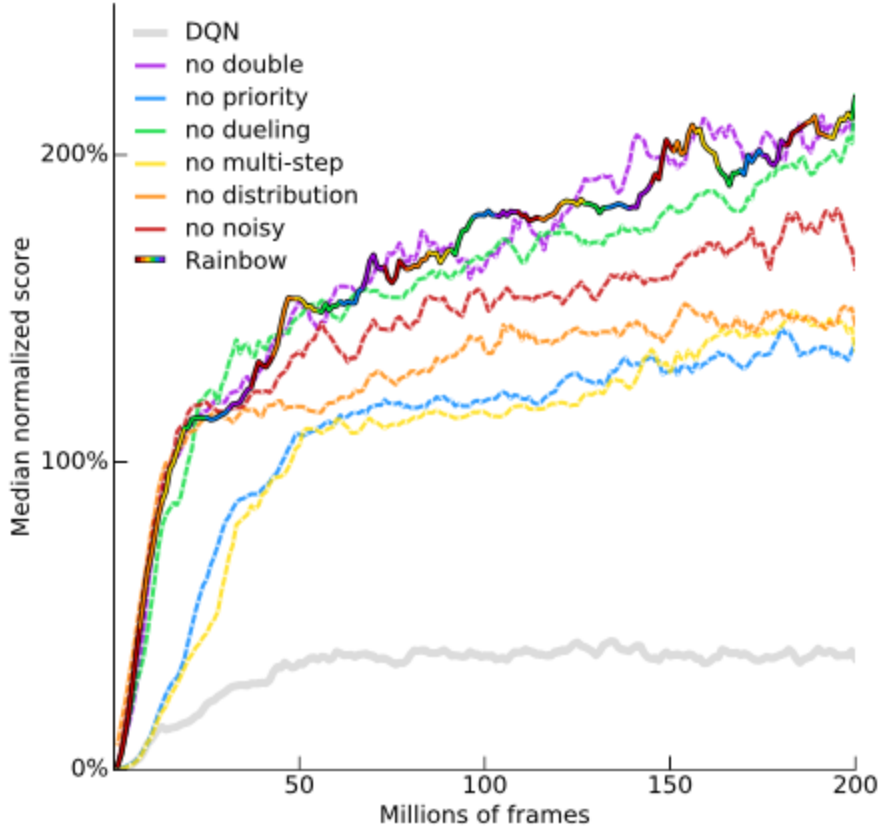
4.6 Noisy Nets

$$y = (b + Wx) + (b_{noisy} \odot \epsilon^b + (W_{noisy} \odot \epsilon^w)x)$$

- ϵ^b, ϵ^w : random variable
- $\odot$: element-wise product

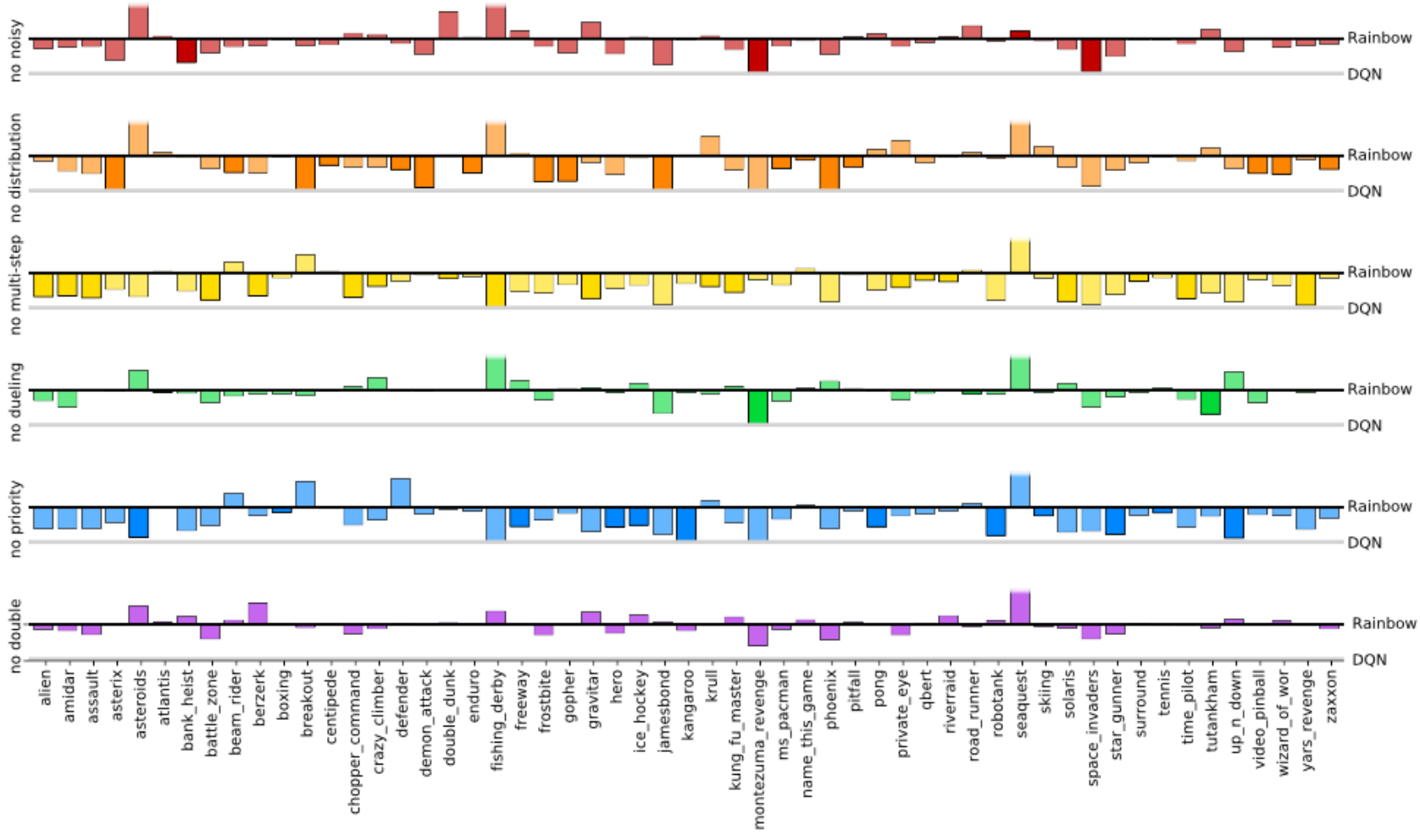
5. Rainbow

- Rainbow DQN은 위의 언급된 six extenstion DQN이 모두 적용된 버전이다.
- 기존 DQN에 비해 월등한 성능을 보였으며, muti-step 또는 priority를 제외하였을때 레인보우의 성능이 떨어졌다.



5. Rainbow -Atari성능 비교

-아타리 57개의 환경에서 비교한 결과는 다음과 같다.



5. Rainbow Hyper-parameters

- Six extenstion DQN이 모두 적용되었으며,
- Rainbow DQN 논문에서 적용된 hyper parameter는 다음과 같다.

Parameter	Value
Min history to start learning	80K frames
Adam learning rate	0.0000625
Exploration ϵ	0.0
Noisy Nets σ_0	0.5
Target Network Period	32K frames
Adam ϵ	1.5×10^{-4}
Prioritization type	proportional
Prioritization exponent ω	0.5
Prioritization importance sampling β	$0.4 \rightarrow 1.0$
Multi-step returns n	3
Distributional atoms	51
Distributional min/max values	$[-10, 10]$

Table 1: Rainbow hyper-parameters

References

Dueling Network Architectures for Deep Reinforcement Learning
https://arxiv.org/pdf/1511.06581.pdf

Rainbow: Combining Improvements in Deep Reinforcement Learning

https://arxiv.org/pdf/1710.02298.pdf

https://wonseokjung.github.io