

대구소프트웨어마이스터고등학교 특강

Rainbow Is All You Need

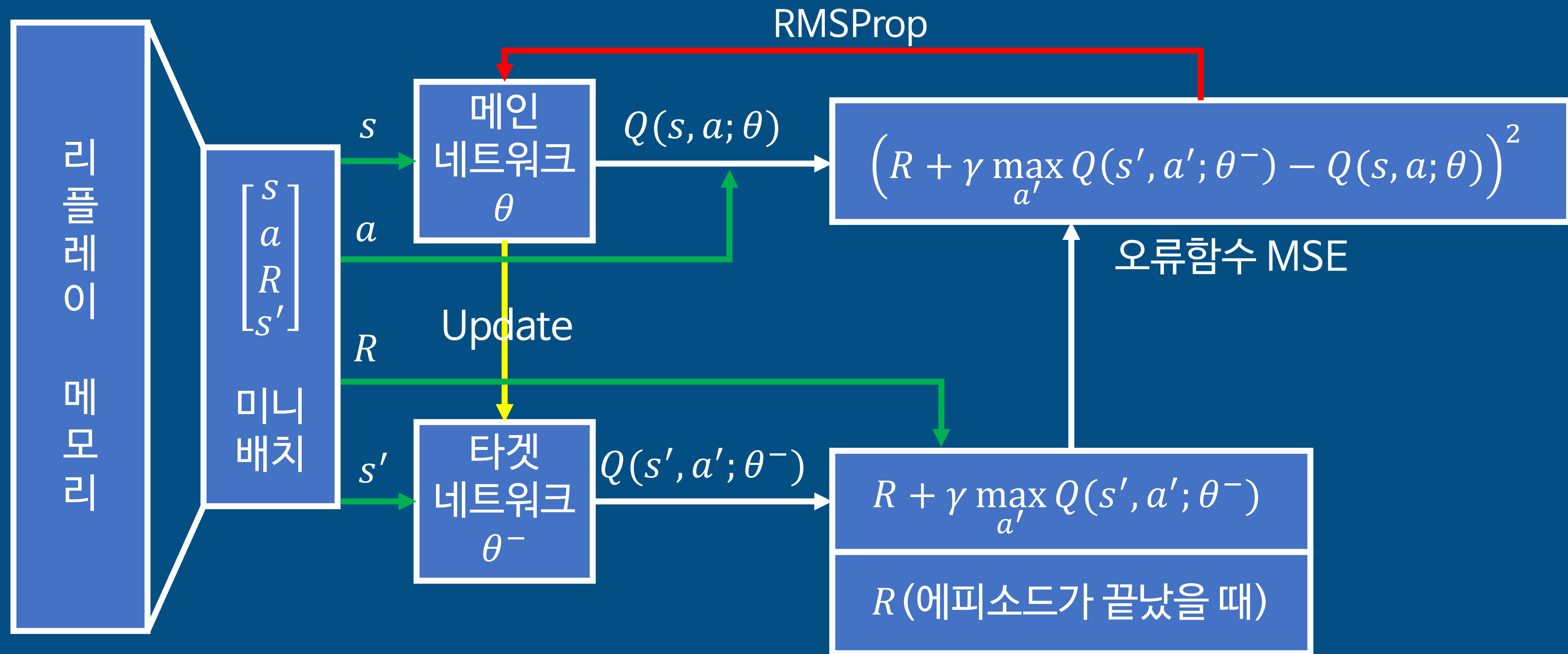
Chris Ohk

utilForever@gmail.com

Basic DQN

2021 DGSW
Rainbow Is All You Need

- DQN 리마인드



- DQN의 특징
 - 오프폴리시 (Off-Policy)
 - 리플레이 메모리 + 미니배치
 - 타겟 신경망
- 온폴리시 : 학습하는 정책과 행동을 고르는 정책이 항상 같아야 한다.
따라서 정책을 업데이트하면 과거의 경험들을 학습에 이용 불가능해 비효율적이다.
- 오프폴리시 : 학습하는 정책과 행동을 고르는 정책이 달라도 된다.
따라서 과거에 경험한 에피소드들도 학습에 계속해서 이용 가능하다.

- 리플레이 메모리 + 미니배치
 - 환경에서 받은 $[s, a, R, s']$ 을 저장한다.
 - 받은 s' 를 새로운 s 로 에이전트에게 전달해 에피소드를 계속 진행시킨다.
 - 리플레이 메모리에 저장되어 있는 $[s, a, R, s']$ 집합에서 일부를 무작위로 샘플링한 뒤 에이전트를 학습시킨다.
 - 리플레이 메모리 크기 이상으로 데이터가 추가되면 오래된 순서대로 지워준다.

- 타겟 신경망

- 기존 오류 함수는 신경망이 스스로 목표를 만들어 내기 때문에 신경망이 업데이트될 때 목표가 되는 정답 부분이 계속 변하고, 그 결과 학습이 굉장히 불안하게 이루어진다.

$$\text{MSE} = (\text{정답} - \text{예측})^2 = \left(R_{t+1} + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta) \right)^2$$

- 따라서 θ^- 를 매개변수로 갖는 타겟 신경망을 추가한다.

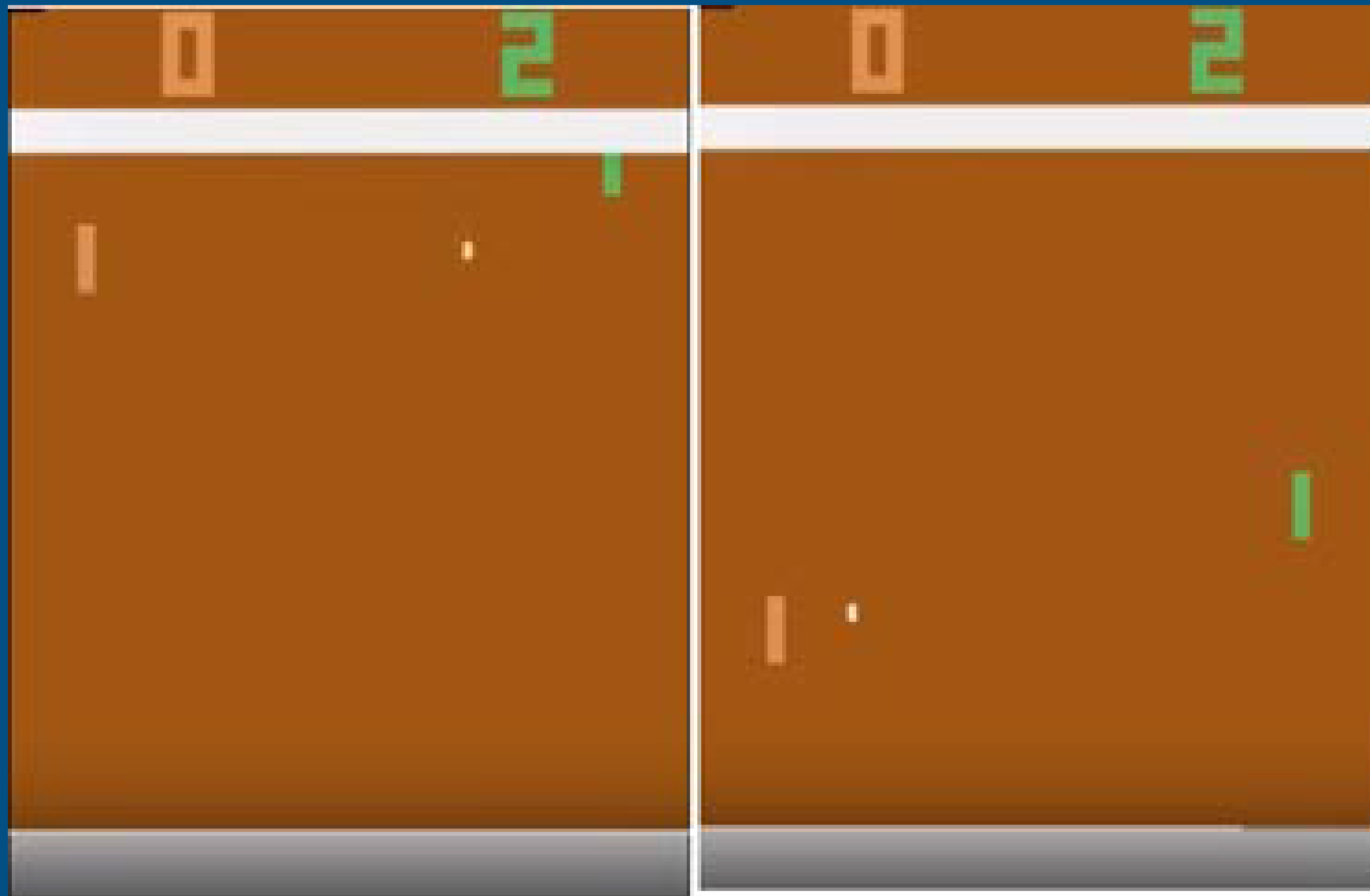
$$\text{MSE} = (\text{정답} - \text{예측})^2 = \left(R_{t+1} + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2$$

- 타겟 신경망은 일정 시간동안 그대로 유지되며 정답을 만들어내다가, 에피소드가 끝날 때마다 업데이트된다.

Basic DQN

2021 DGSW
Rainbow Is All You Need

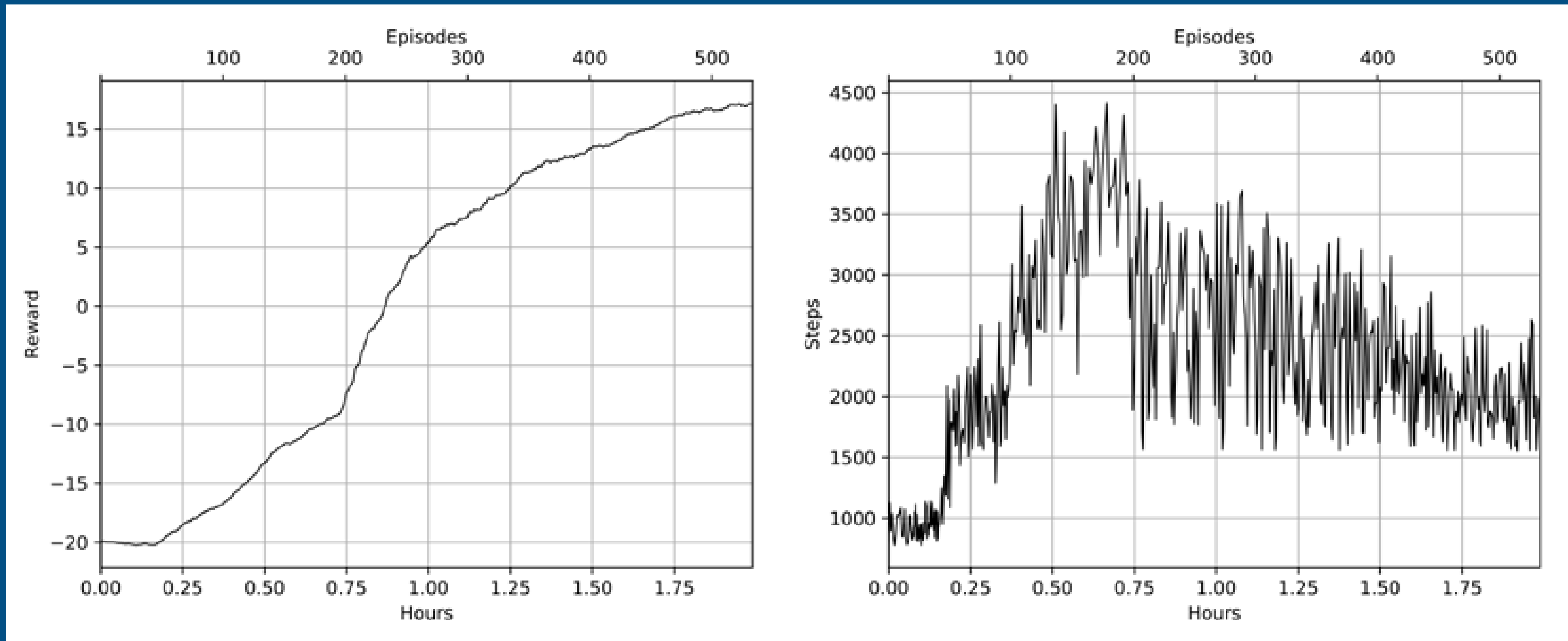
- 테스트 환경 : Pong



Basic DQN

2021 DGSW
Rainbow Is All You Need

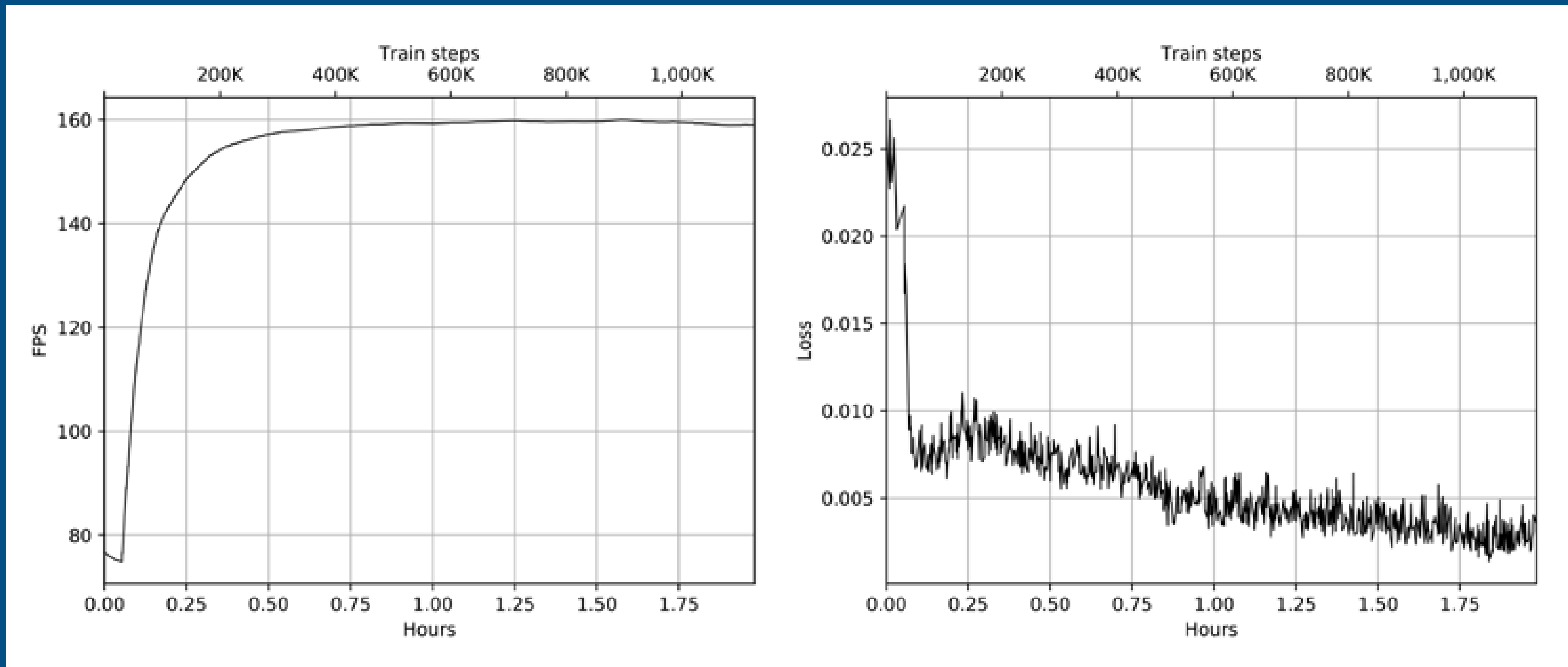
- 실험 결과 (Reward, Steps)



Basic DQN

2021 DGSW
Rainbow Is All You Need

- 실험 결과 (Speed, Loss)



- Learning to Predict by the Methods of Temporal Differences (Sutton, 1988)
- 큐러닝에서 사용했던 벨만 방정식

$$Q(s_t, a_t) = r_t + \gamma \max_a Q(s_{t+1}, a_{t+1})$$

- 위 방정식에서 $Q(s_{t+1}, a_{t+1})$ 을 풀어서 표현할 수 있다.

$$Q(s_t, a_t) = r_t + \gamma \max_a \left[r_{a,t+1} + \gamma \max_{a'} Q(s_{t+2}, a') \right]$$

- 여기서 $r_{a,t+1}$ 은 행동 a 를 수행한 뒤 시간 $t + 1$ 에 받은 보상을 말한다.

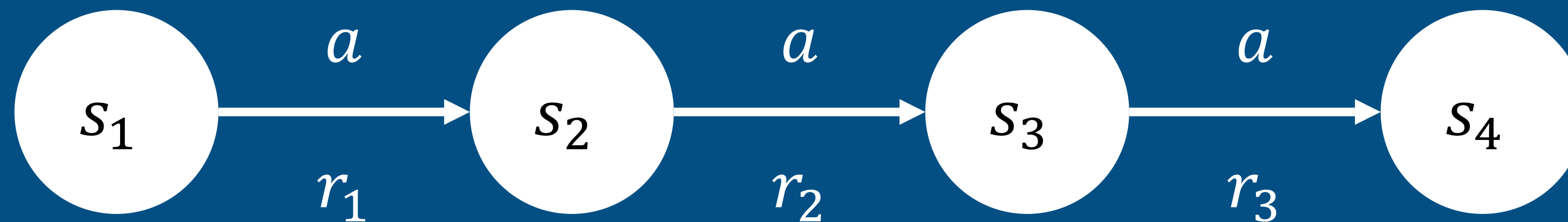
만약 시간 $t + 1$ 에서의 행동 a 가 최적이었다고 가정한다면, \max_a 연산을 생략할 수 있다.

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

N-step DQN

2021 DGSW
Rainbow Is All You Need

- 이를 활용해 DQN의 업데이트 공식을 N-step으로 바꿀 수 있다.
→ N-step까지 관찰된 누적 보상과 N번째 스텝에서 부트스트래핑 값을 합한다.
N을 적절하게 선택한다면 1-step보다 좋은 성능을 낼 수 있다.
- 어떻게 가능할까? 예제를 한 번 살펴보자. 다음과 같이 상태가 4개인 환경이 있다고 하자.



- 1-step인 경우에 무슨 일이 일어나는가? 총 3번의 업데이트가 발생한다.
 - $Q(s_1, a) \leftarrow r_1 + \gamma Q(s_2, a)$
 - $Q(s_2, a) \leftarrow r_2 + \gamma Q(s_3, a)$
 - $Q(s_3, a) \leftarrow r_3$
- 첫번째 반복에서 첫 두 업데이트는 쓸모가 없다.
왜냐하면 현재 $Q(s_2, a)$ 와 $Q(s_3, a)$ 에 임의의 값으로 초기화되어 있기 때문이다.
- 두번째 반복에서는 어떨까?
 $Q(s_2, a)$ 에는 올바른 값이 대입되겠지만 $Q(s_1, a)$ 은 여전히 노이즈가 있다.
세번째 반복이 되어서야 모두 올바른 값을 얻게 된다.
→ 따라서 1-step의 경우 모든 상태에 올바른 값을 전파하려면 3-step이 필요하다.

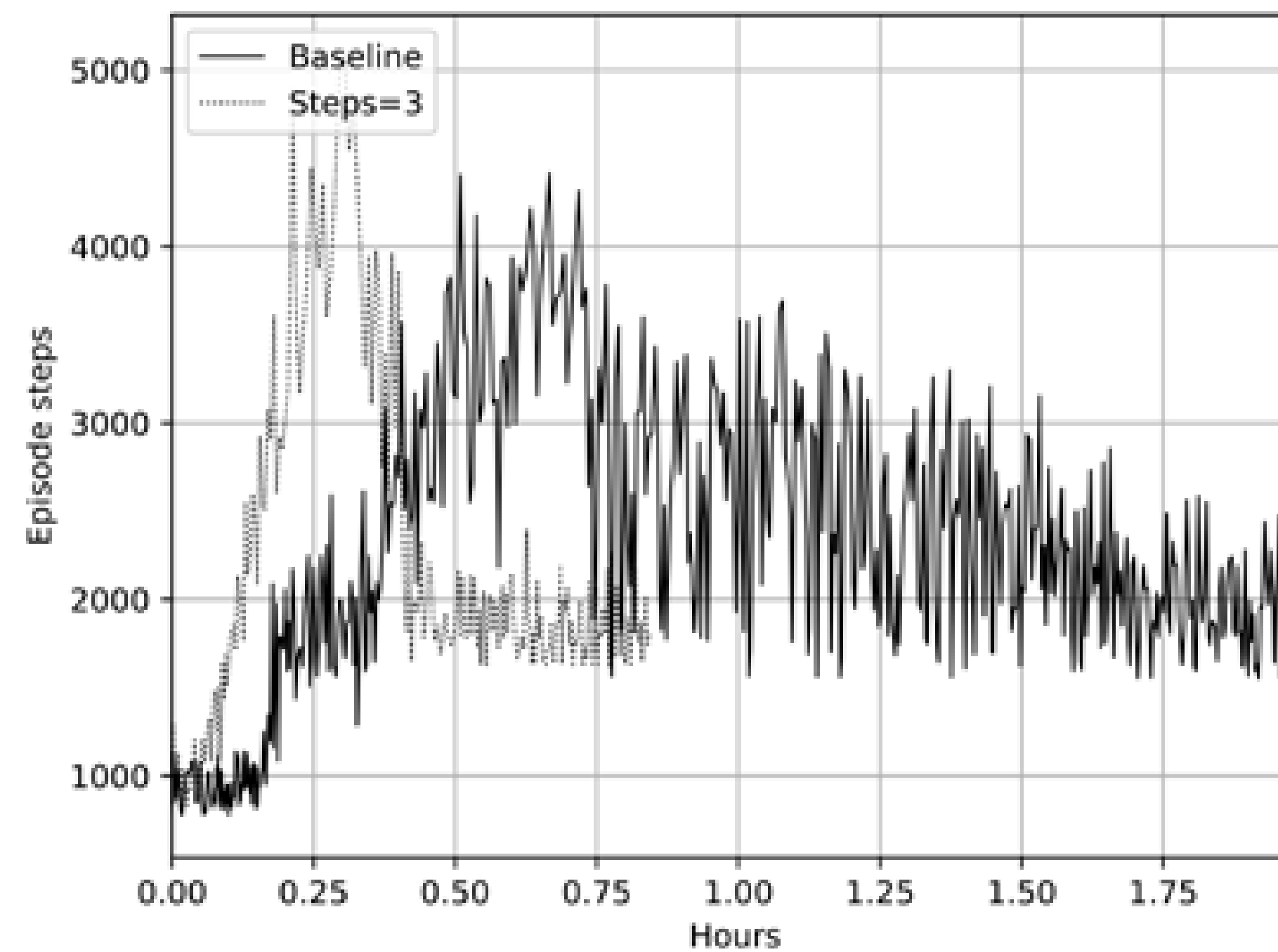
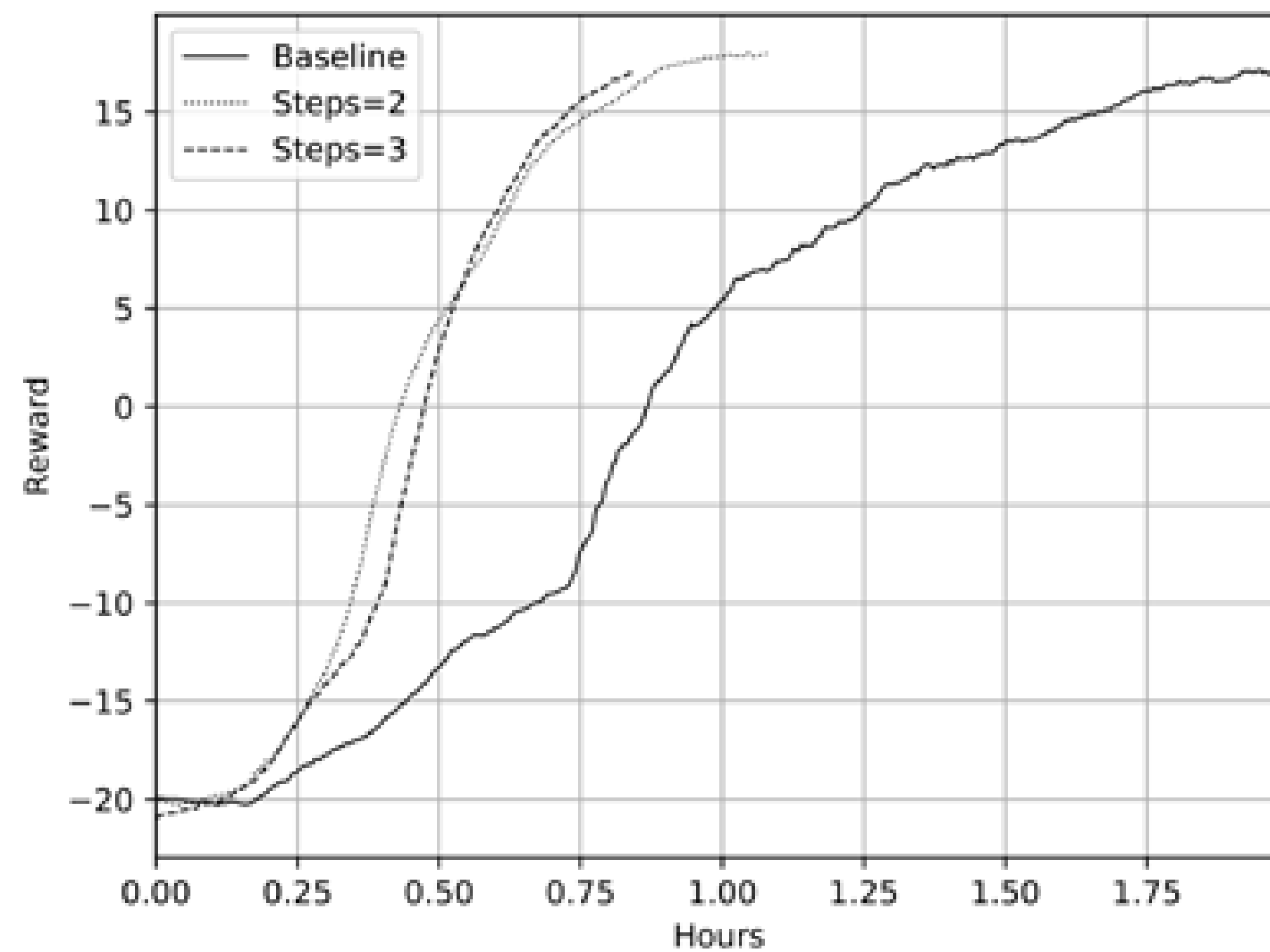
- 이제 2-step으로 수정한 뒤 살펴보자. 총 3번의 업데이트가 발생한다.
 - $Q(s_1, a) \leftarrow r_1 + \gamma r_2 + \gamma^2 Q(s_3, a)$
 - $Q(s_2, a) \leftarrow r_2 + \gamma r_3$
 - $Q(s_3, a) \leftarrow r_3$
- 첫번째 반복에서 $Q(s_2, a)$ 와 $Q(s_3, a)$ 에 올바른 값이 대입될 것이다.
- 두번째 반복에서 $Q(s_1, a)$ 이 올바른 값으로 갱신될 것이다.
 - N-step을 사용하면 값의 전파 속도가 향상되고, 더 빨리 수렴하게 된다.

- 생각해 보자.
“N-step을 사용하면 장점이 많으니, $N = 100$ 으로 해서 벨만 방정식을 풀면 어떨까?”
“이렇게 하면 수렴 속도가 100배 빨라질까?”

N-step DQN

2021 DGSW
Rainbow Is All You Need

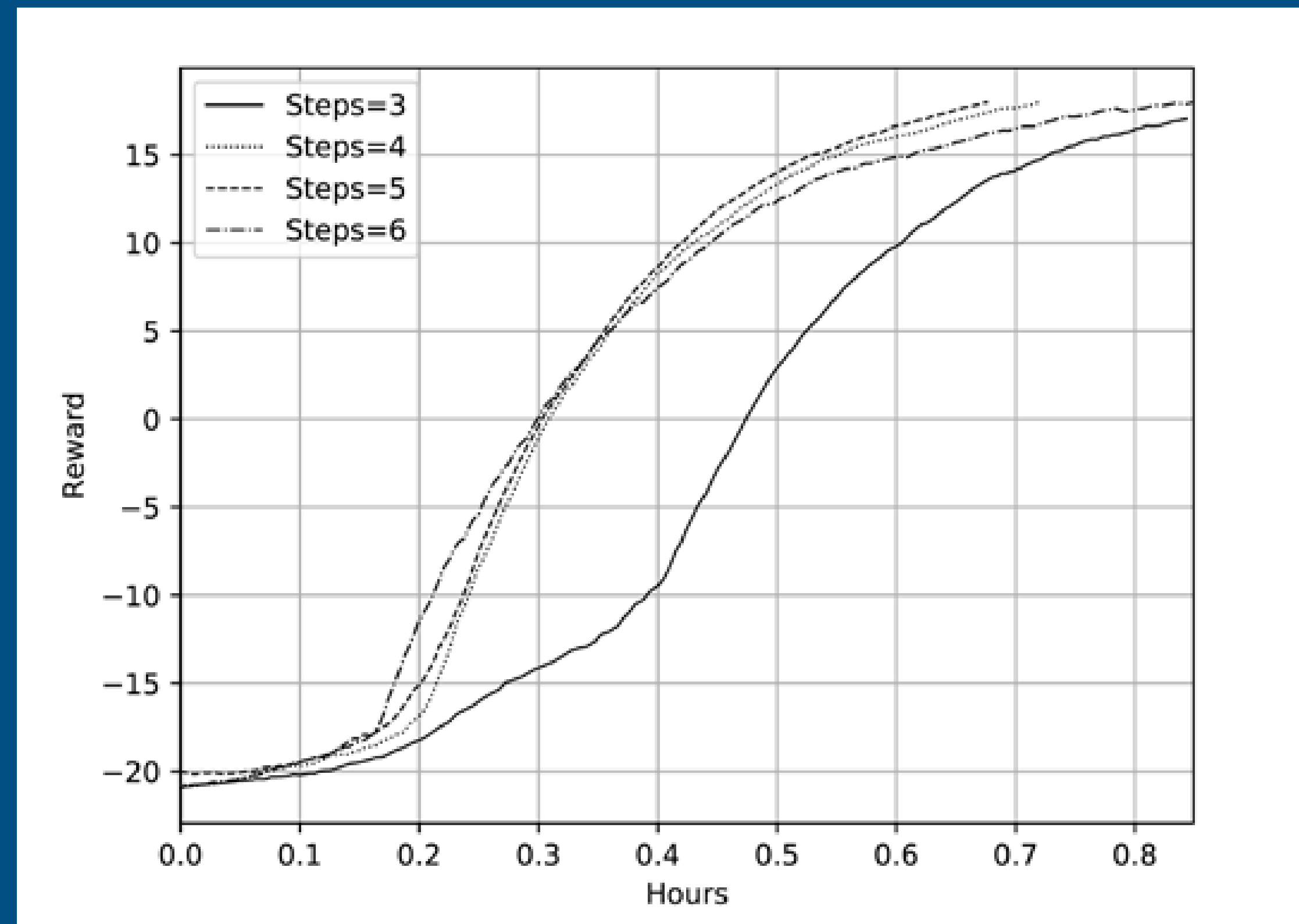
- Basic DQN vs N-step DQN (Reward, Steps)



N-step DQN

2021 DGSW
Rainbow Is All You Need

- 서로 다른 N 값에 따른 비교



- Deep Reinforcement Learning with Double Q-Learning (van Hasselt, Guez, and Silver, 2015)
- DQN에서 타겟 값으로 사용한 식은 다음과 같다. 이대로 괜찮은가?

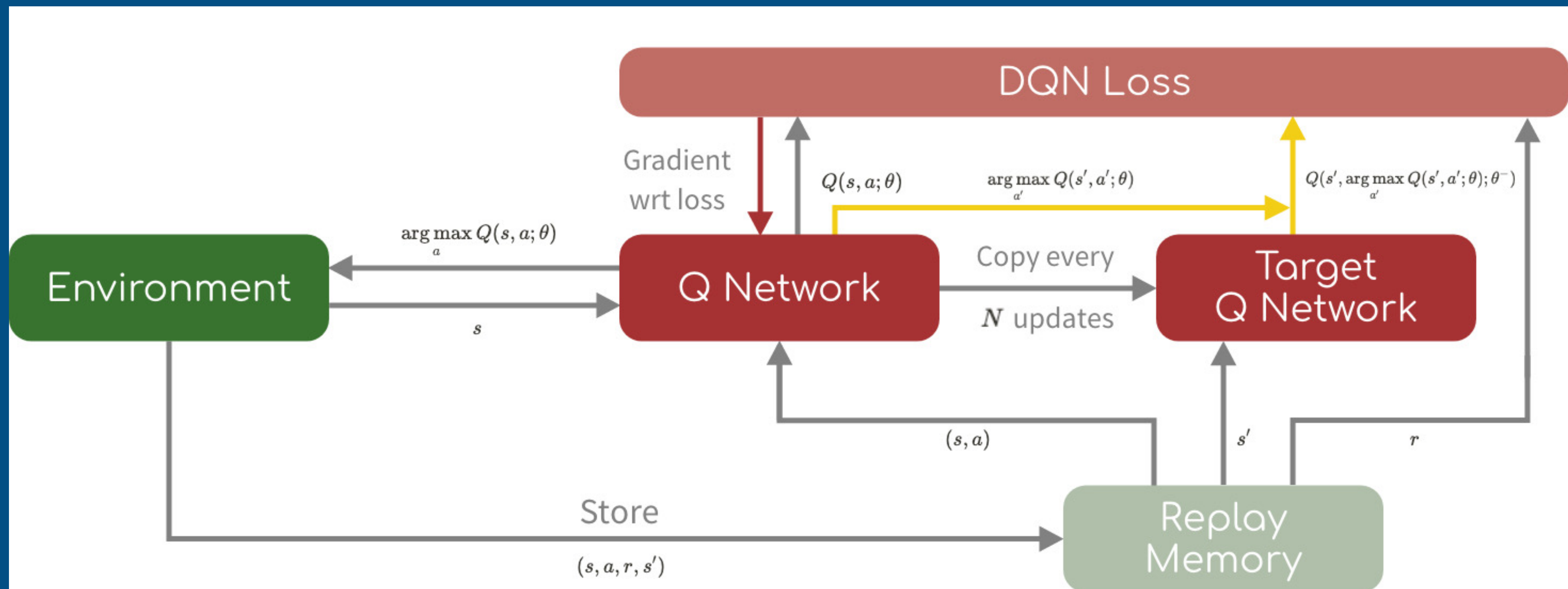
$$Y_t^{DQN} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-)$$

- 위 식에서 \max_a 함수가 쫓아가야 할 타겟 값을 최적 값보다 크게 추정(Overestimate)한다.
→ 논문에서는 타겟 값으로 사용하는 식을 수정한다. (Double DQN)

$$Y_t^{DDQN} = R_{t+1} + \gamma Q\left(S_{t+1}, \operatorname{argmax}_a Q(S_{t+1}, a; \theta_t); \theta_t'\right)$$

Double DQN

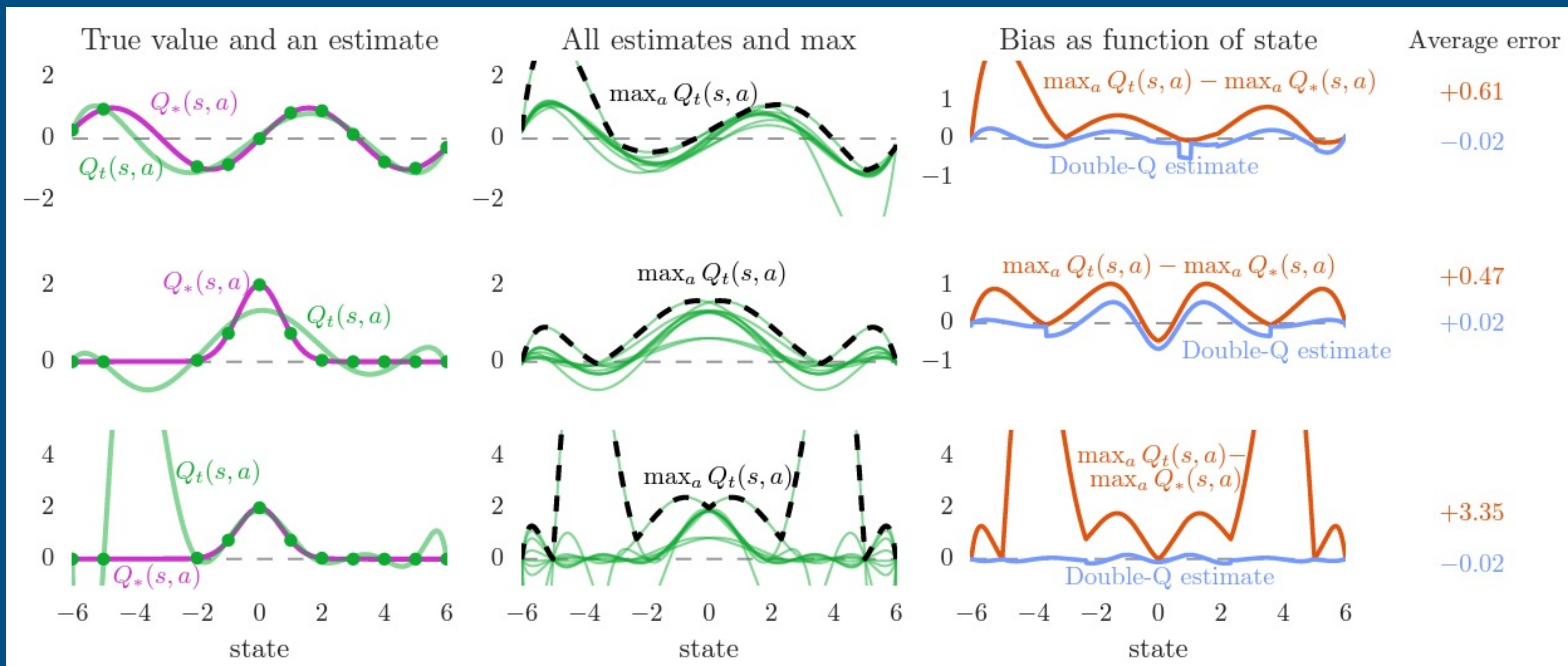
2021 DGSW
Rainbow Is All You Need



Double DQN

2021 DGSW
Rainbow Is All You Need

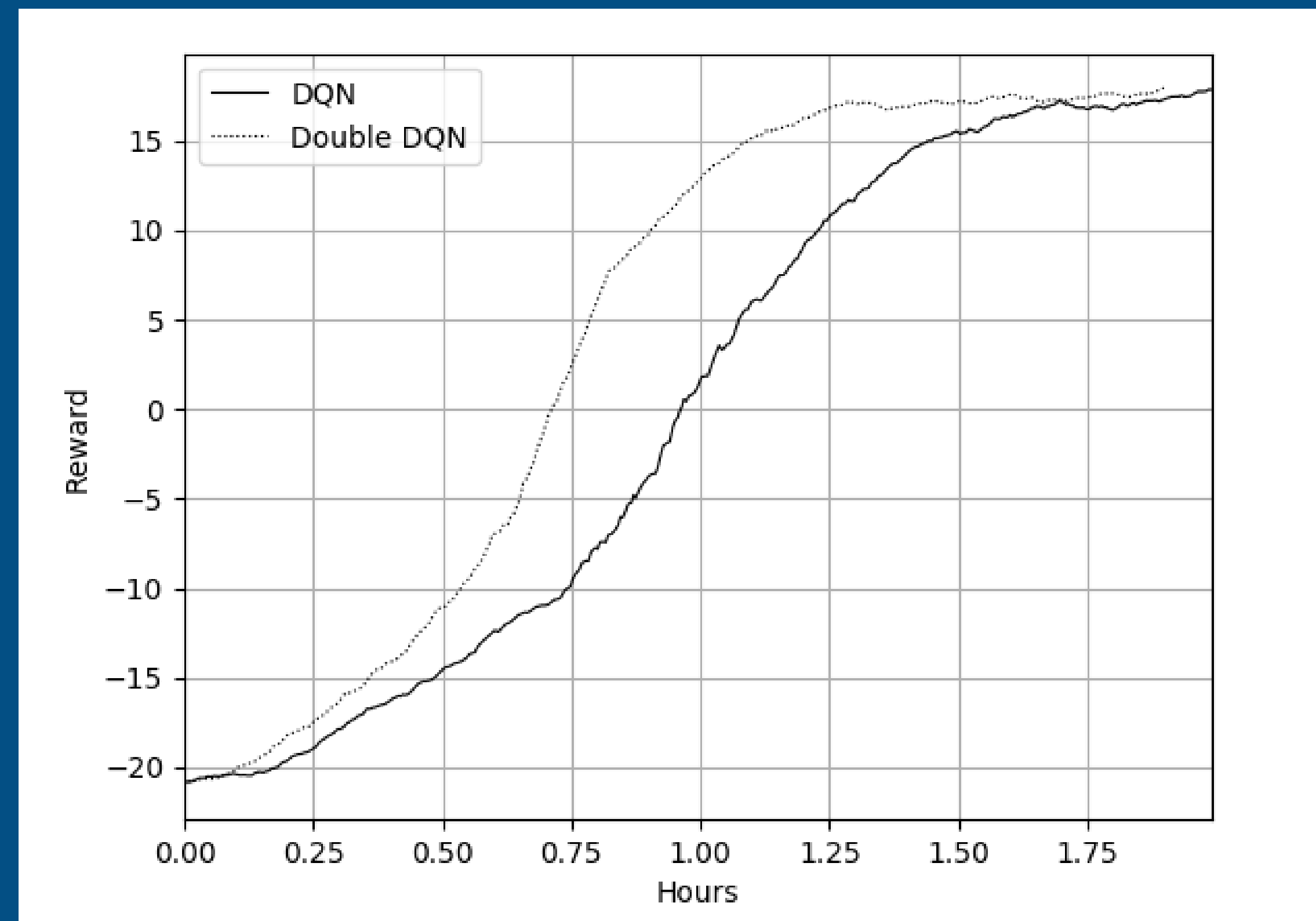
- Q_* : 최적 값, Q_t : DDQN을 이용한 함수 추정 값



Double DQN

2021 DGSW
Rainbow Is All You Need

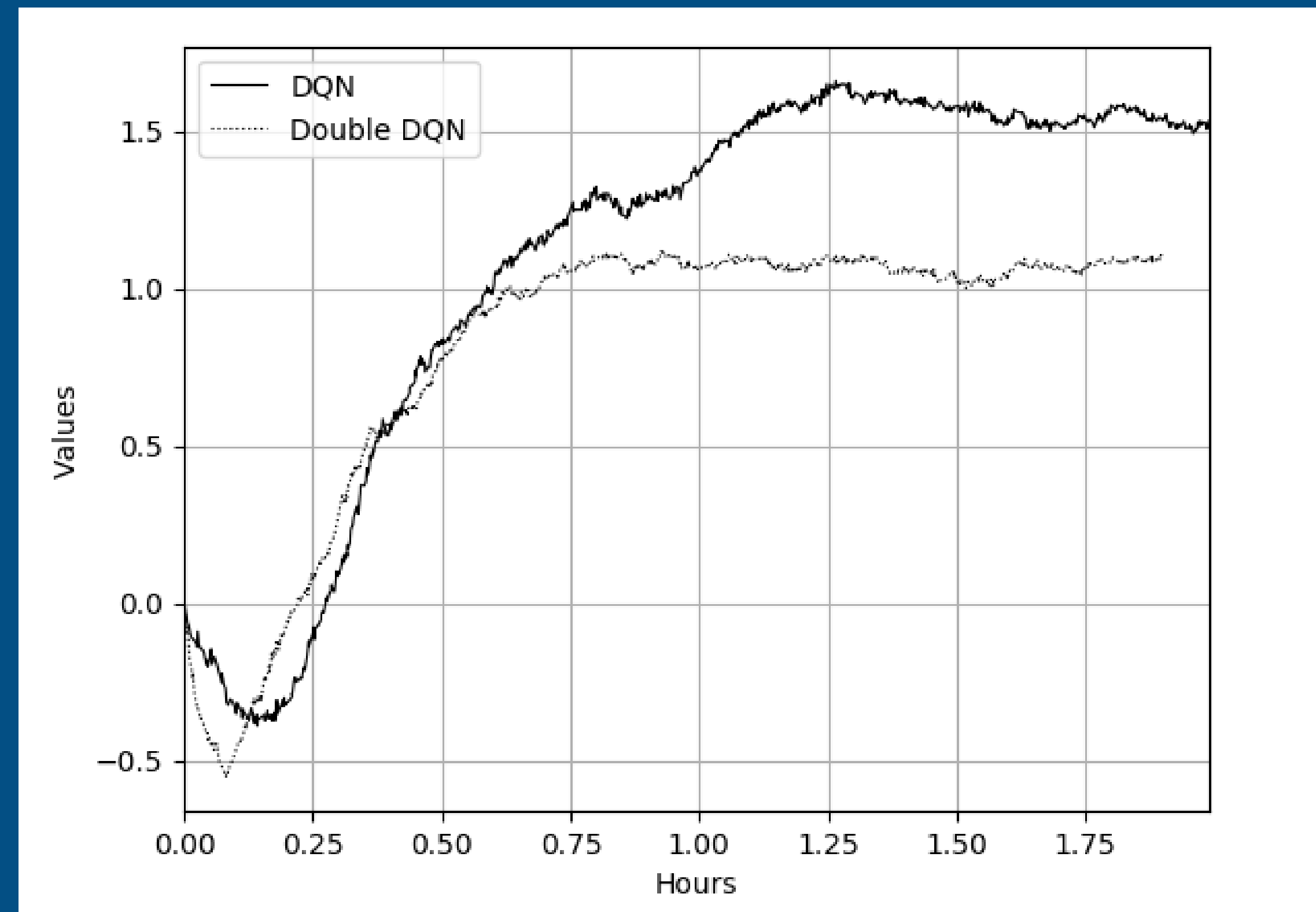
- Basic DQN vs Double DQN (Reward)



Double DQN

2021 DGSW
Rainbow Is All You Need

- Basic DQN vs Double DQN (Values)



Noisy Networks

2021 DGSW
Rainbow Is All You Need

- Noisy Networks for Exploration (Fortunato and others, 2017)
- Exploitation vs Exploration

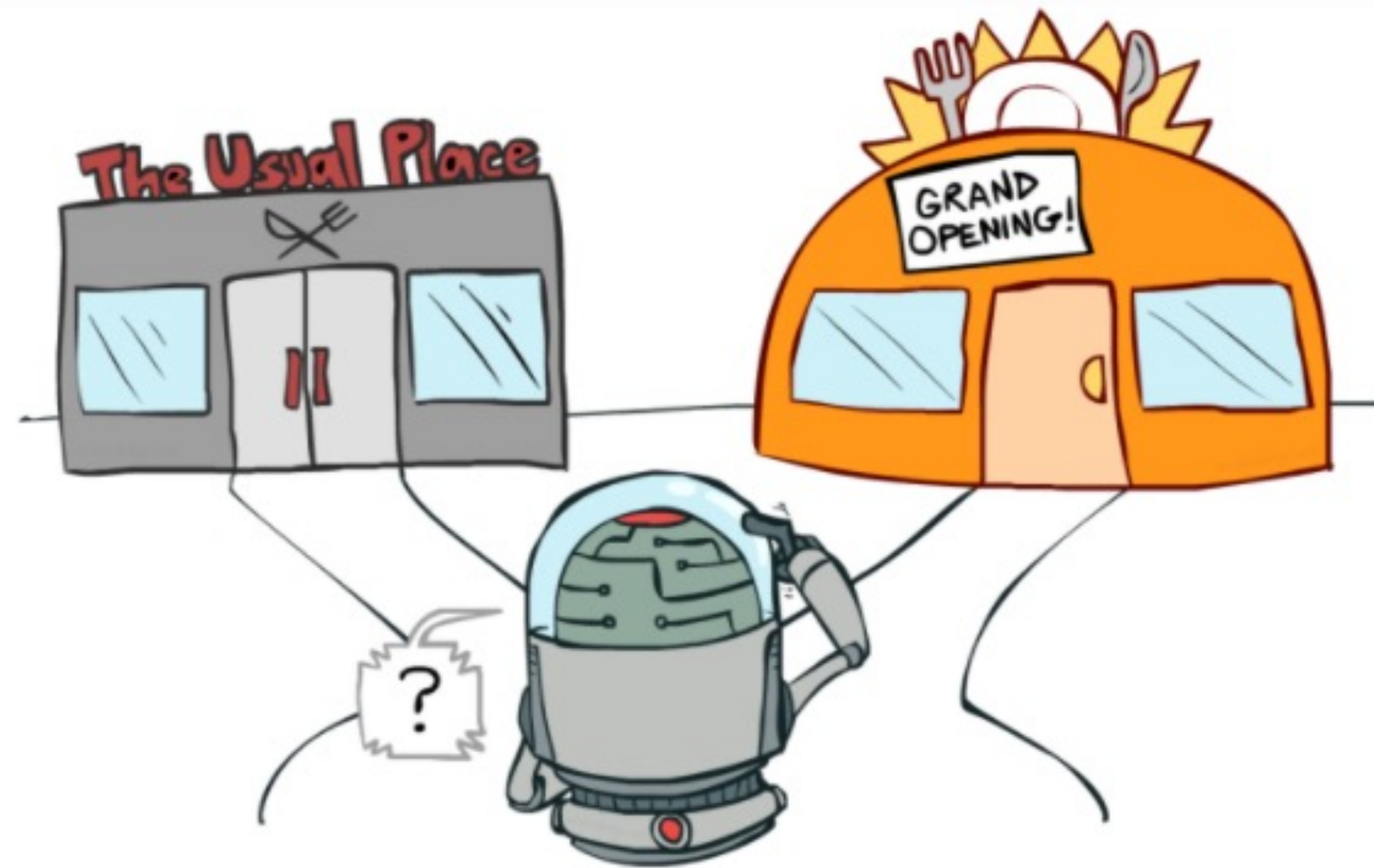


Fig. 1. A real-life example of the exploration vs exploitation dilemma: where to eat? (Image source: UC Berkeley AI course [slide](#), [lecture 11](#).)

- DQN에서는 Epsilon-Greedy 알고리즘을 사용한다.

$$\pi(s) = \begin{cases} a^* = \operatorname{argmax}_{a \in A} Q(s, a), & 1 - \epsilon \\ a \neq a^* & , \epsilon \end{cases}$$

- $1 - \epsilon$ 의 확률로 현재 상태에서 가장 큰 큐함수의 값을 갖는 행동을 선택 = 탐욕 정책
- ϵ 의 확률로 엉뚱한 행동을 선택 = 탐험
- 하지만 위와 같은 탐험 알고리즘은 단점이 존재한다.
 - 휴리스틱한 방법이다.
 - 현재 상태에 관계 없이 노이즈를 적용한다.
- 그렇다면 휴리스틱하지 않고 현재 상태에 따라 노이즈를 어떻게 적용할 수 있을까?
→ 신경망의 가중치에 가우시안 노이즈를 적용해서 탐험을 하자! (Noisy Network)

Noisy Networks

2021 DGSW
Rainbow Is All You Need

- Parameters and Variables

Noisy networks

- Fully connected layer
- Noisy fully connected layer

w shape=(q, p), b shape=(q)

$$y = wx + b,$$
$$y \stackrel{\text{def}}{=} (\mu^w + \sigma^w \odot \varepsilon^w)x + \mu^b + \sigma^b \odot \varepsilon^b,$$

(q, p) (q, p) (q, p) (q) (q) (q)

Noisy Networks

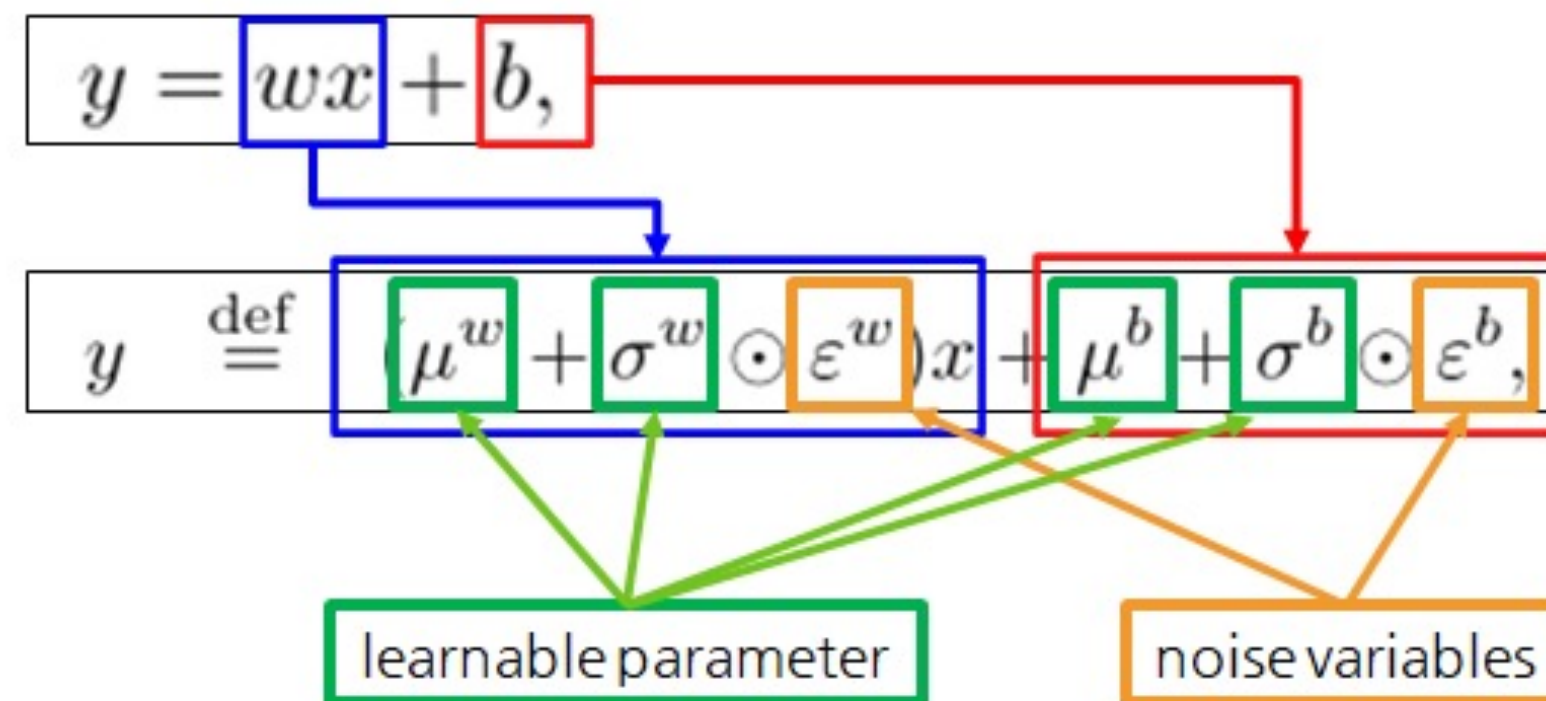
2021 DGSW
Rainbow Is All You Need

- Learnable Parameters vs Noise Variables

Noisy networks

- Fully connected layer
- Noisy fully connected layer
- Element-wise multiplication

w shape=(q, p), b shape=(q)

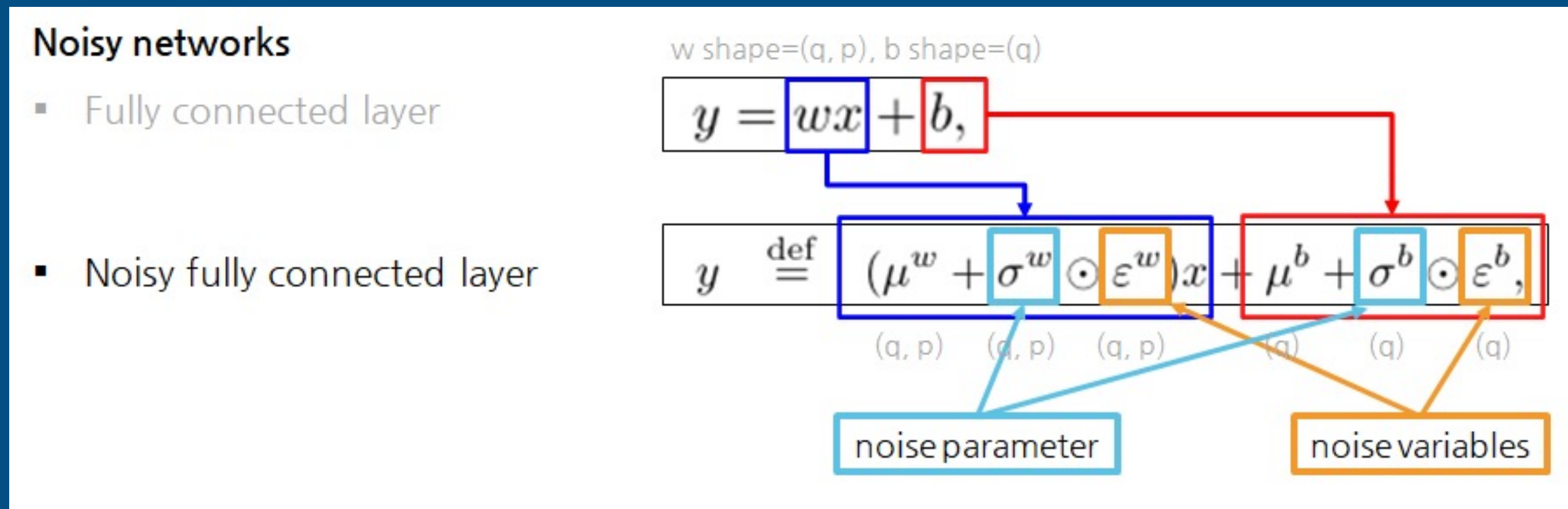


$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11} b_{11} & a_{12} b_{12} & a_{13} b_{13} \\ a_{21} b_{21} & a_{22} b_{22} & a_{23} b_{23} \\ a_{31} b_{31} & a_{32} b_{32} & a_{33} b_{33} \end{bmatrix}$$

Noisy Networks

2021 DGSW
Rainbow Is All You Need

- Noise Parameters vs Noise Variables



Noisy Networks

2021 DGSW
Rainbow Is All You Need

- Independent Gaussian Noise

Noisy networks

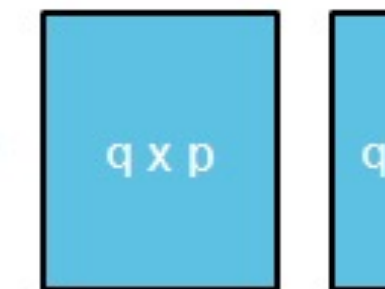
- Fully connected layer
- Noisy fully connected layer

w shape=(q, p), b shape=(q)

$$y = wx + b,$$
$$y \stackrel{\text{def}}{=} (\mu^w + \sigma^w \odot \varepsilon^w)x + \mu^b + \sigma^b \odot \varepsilon^b,$$

Diagram illustrating the noisy fully connected layer equation. The equation is shown with color-coded boxes and arrows indicating the mapping of terms between the standard and noisy versions. A blue box highlights the weight term w in the first equation and its noisy counterpart $(\mu^w + \sigma^w \odot \varepsilon^w)$ in the second. A red box highlights the bias term b in the first equation and its noisy counterpart $(\mu^b + \sigma^b \odot \varepsilon^b)$ in the second. A light blue box labeled "noise parameter" points to σ^w and σ^b . An orange box labeled "noise variables" points to ε^w and ε^b . Dimensions are noted below the terms: (q, p) for μ^w , σ^w , and ε^w ; (q) for μ^b , σ^b , and ε^b .

- (a) Independent Gaussian noise
 - Gaussian distribution에서 $q \times p$ 개, q 개의 noise를 추출 : ε^w , ε^b



Noisy Networks

2021 DGSW
Rainbow Is All You Need

- Factorized Gaussian Noise

Noisy networks

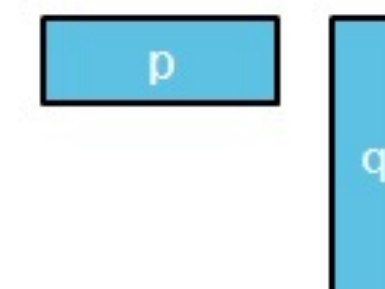
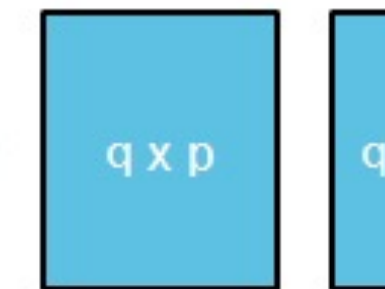
- Fully connected layer
- Noisy fully connected layer

w shape=(q, p), b shape=(q)

$$y = wx + b,$$
$$y \stackrel{\text{def}}{=} (\mu^w + \sigma^w \odot \varepsilon^w)x + \mu^b + \sigma^b \odot \varepsilon^b,$$

Diagram illustrating the mapping from the standard fully connected layer equation to the noisy fully connected layer equation. The standard equation $y = wx + b$ is shown with w (shape (q, p)) and b (shape (q)). The noisy equation $y \stackrel{\text{def}}{=} (\mu^w + \sigma^w \odot \varepsilon^w)x + \mu^b + \sigma^b \odot \varepsilon^b$ is shown with corresponding shapes: μ^w (q, p), σ^w (q, p), ε^w (q, p), μ^b (q), σ^b (q), and ε^b (q). Arrows indicate the mapping from the standard equation to the noisy equation. A blue box labeled "noise parameter" points to σ^w and σ^b . An orange box labeled "noise variables" points to ε^w and ε^b .

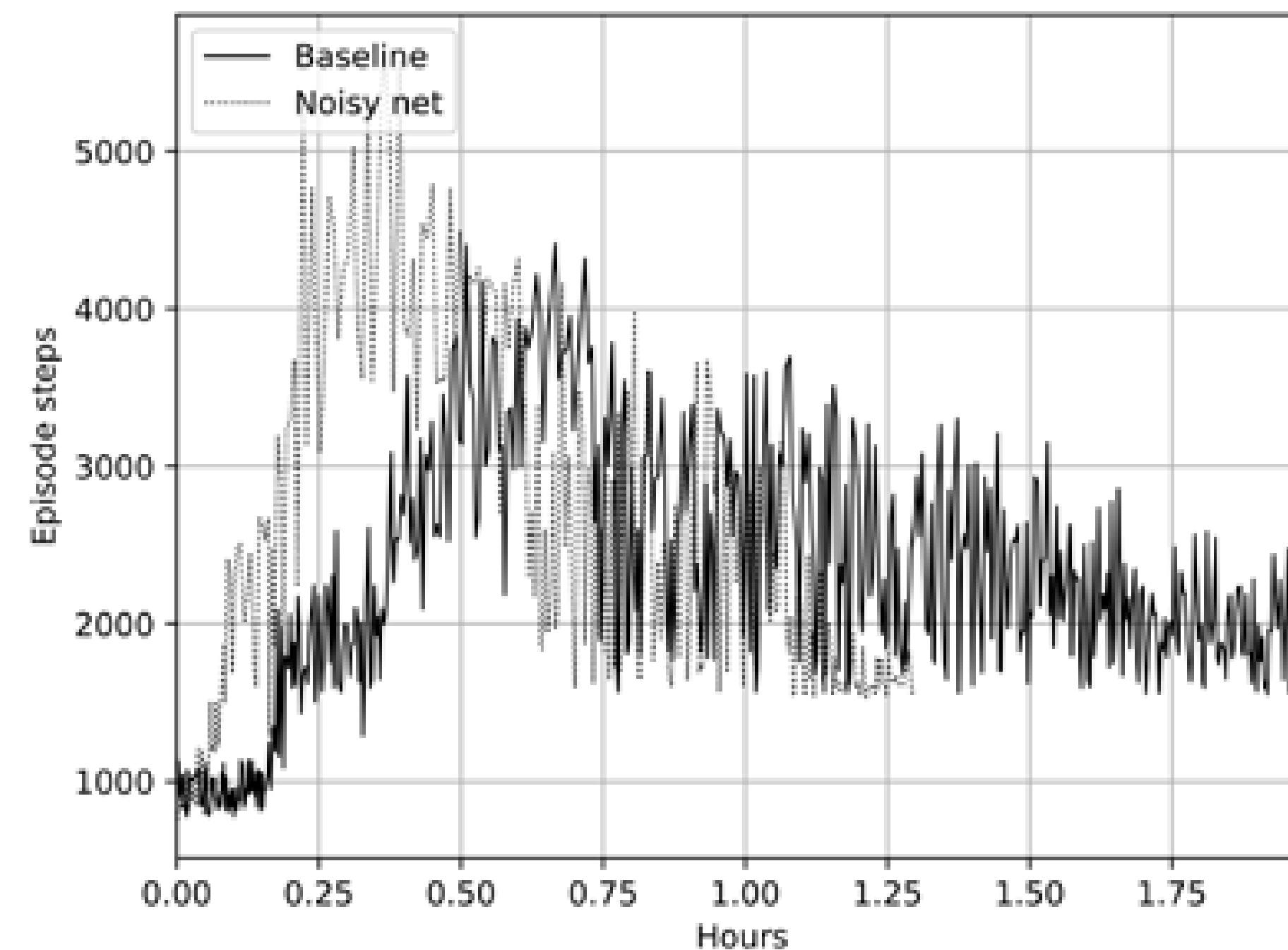
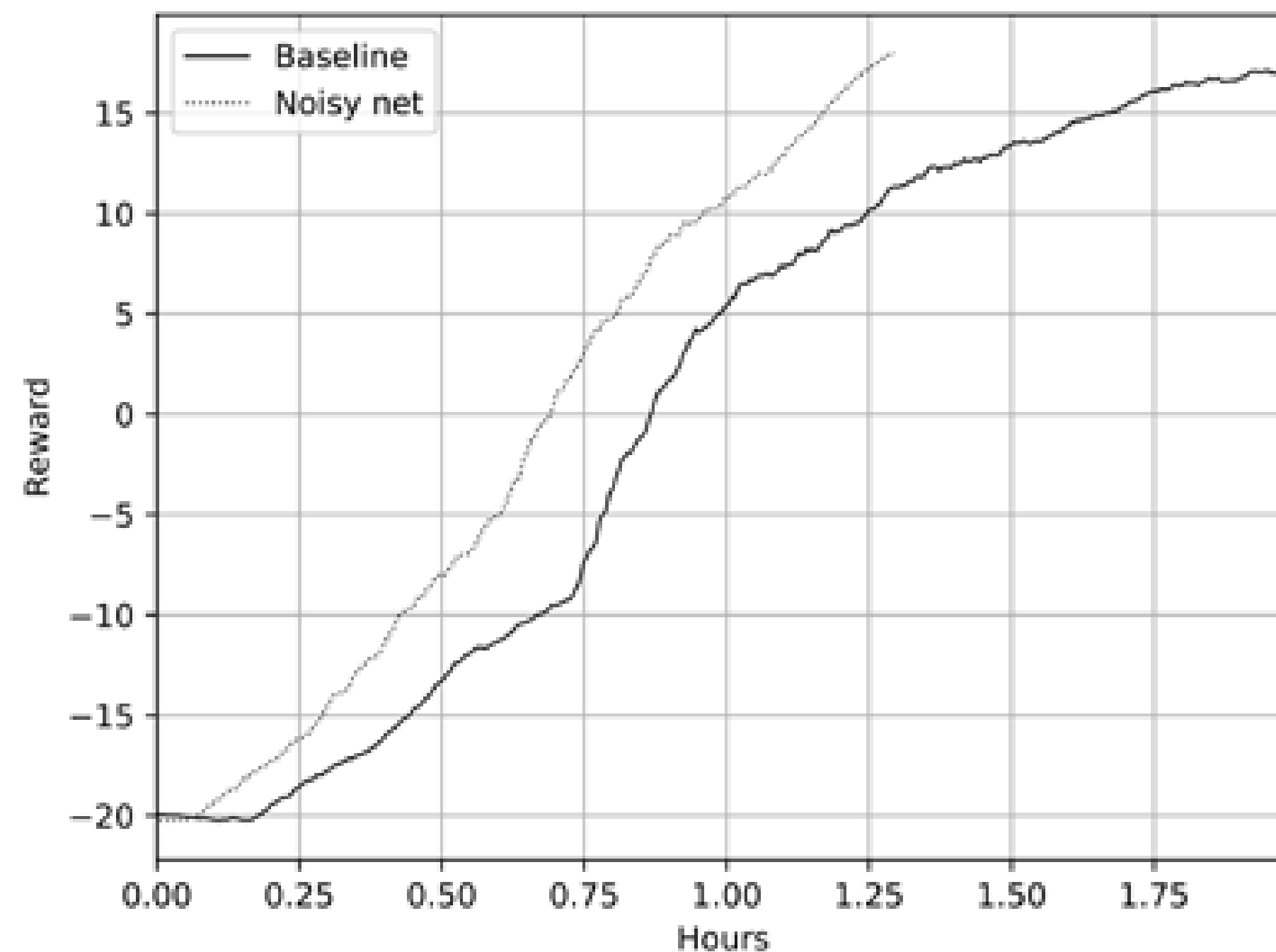
- (a) Independent Gaussian noise
 - Gaussian distribution에서 $q \times p$ 개, q 개의 noise를 추출 : $\varepsilon^w, \varepsilon^b$
- (b) Factorised Gaussian noise
 - Gaussian distribution에서 p 개, q 개의 noise를 추출 : $\varepsilon_p, \varepsilon_q$
 - 우리가 필요한 건 $q \times p$ 개인 ε^w , q 개인, ε^b
 - $\varepsilon^w = f(\varepsilon_p) \otimes f(\varepsilon_q)$, $\varepsilon^b = f(\varepsilon_q)$, $f(x) = \text{sgn}(x) \sqrt{|x|}$



Noisy Networks

2021 DGSW
Rainbow Is All You Need

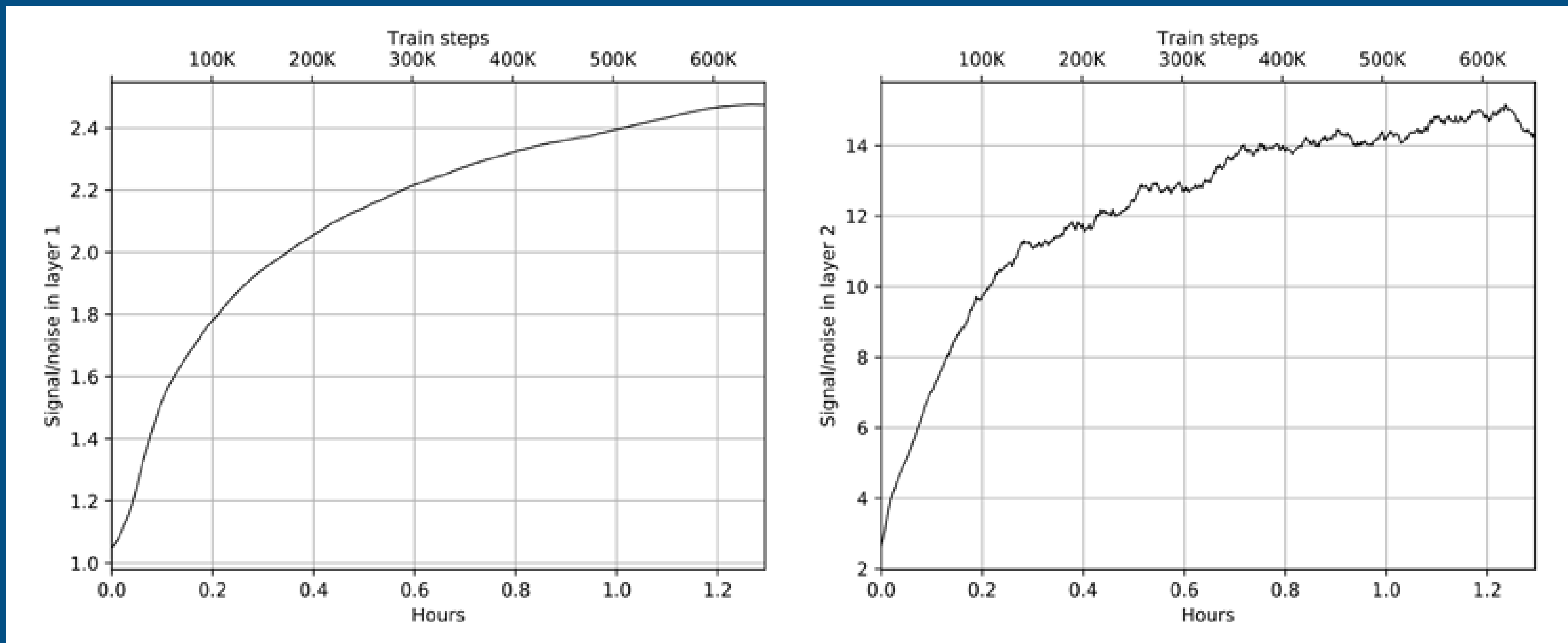
- Basic DQN vs Noisy Network (Reward, Steps)



Noisy Networks

2021 DGSW
Rainbow Is All You Need

- Signal-to-Noise Ratio (SNR) 비교



Prioritized Replay Buffer

2021 DGSW
Rainbow Is All You Need

- Prioritized Experience Replay (Schaul and others, 2015)
- 게임을 할 때 승패에 지대한 영향을 주는 사건이나 경험은 드문드문 발생한다.
대부분의 경험은 그에 비해 상대적인 중요성이 떨어진다.
- 우리가 학습시킬 에이전트 역시 균등한 랜덤 샘플링으로 모든 경험을 균등하게
취급할 것이 아니라, 더 중요하고 희소한 경험들에 더 가중치를 주어야 하지 않을까?
→ 각 경험의 우선 순위를 매기고 이를 기반으로 샘플링을 수행! (PER)

- TD Error as a Reasonable Proxy

- 에이전트가 환경 안에서 쌓는 수많은 경험 데이터 $[s, a, R, s']$ 중에 어떤 경험이 더 중요한 경험일까? 중요한 경험을 무엇을 기준으로 정량화할 수 있을까? 어떤 경험으로부터 얻는 교훈의 크기가 클수록 그 경험이 중요하다고 할 수 있겠지만, 이는 강화학습에서 직접적으로 얻을 수 없다.

- 논문에서는 기준으로 TD Error δ 의 크기를 사용한다.
TD Error은 큐러닝 시 Target Q와 Expected Q 값의 차이를 의미한다.

$$\delta_t = R_t + \gamma_t \max_a Q(S_t, a) - Q(S_{t-1}, A_{t-1})$$

- PER에서는 각 경험의 가중치를 우선 순위(Priority)로 명명하고 다음과 같이 정의한다.

$$p_i = |\delta_i| + \epsilon$$

- TD Error δ 의 크기를 구해야 하므로 절대값을 씌운다. 그리고 아주 작은 상수 ϵ 을 더해주는데, 만약 Target Q와 Expected Q 값이 완전 같아 δ 가 0이 되어버리면 해당 경험은 아예 뽑히지 않을 가능성이 생기기 때문이다.
모든 경험이 약간씩은 추출될 가능성을 담보하는 역할을 ϵ 이 수행한다.

Prioritized Replay Buffer

2021 DGSW
Rainbow Is All You Need

- Hyperparameter α
 - 우선 순위를 샘플링 확률로 그대로 사용하면 우선 순위가 높은 소수의 샘플만 계속 학습에 사용될 가능성이 있다. 이를 어느정도 조절하기 위해 하이퍼파라미터 α 를 사용해 샘플링 확률 $P(i)$ 를 계산한다.

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

- α 는 0과 1사이의 값으로, 0이 되면 $P(i) = \frac{1}{k}$ 가 되어 균등한 랜덤 샘플링과 같아진다. 그리고 1이 되면 $P(i) = p_i$ 이 되어 우선 순위가 그대로 샘플링 확률에 반영된다.

- Hyperparameter β

- 몬테 카를로 방식은 수많은 에피소드 샘플을 생성한 다음, 각 에피소드를 순회하면서 상태-행동에 해당하는 Q 테이블의 값을 업데이트하는 방식을 취했다.

$$Q \leftarrow Q + \frac{1}{N}(G - Q)$$

- G 는 해당 에피소드의 상태-행동 값, Q 는 누적 이동 평균의 값, N 은 에피소드의 횟수를 의미한다. 에피소드가 계속 쌓일수록 N 이 커져 업데이트의 크기가 줄어드는 문제를 해결하기 위해 $1/N$ 을 상수 α 로 바꾸는 방식을 사용하게 되었다.
- α 를 사용하더라도 각 샘플마다 다른 값이 곱해지는 것은 아니기 때문에 모든 샘플이 똑같이 사용되지만, PER에서는 우선 순위를 사용하기 때문에 가중치가 높은 샘플들이 더 많이 사용될 가능성이 매우 높다. PER 이전까지는 균등 분포를 따르는 샘플링을 썼기 때문에 이러한 균등 가정을 깨뜨릴 이유가 없었다.

- Hyperparameter β
 - 균등하지 않은 랜덤 샘플링에 따른 바이어스를 보정하기 위해 PER에서는 매개 변수를 업데이트할 때 중요도 샘플링 (Importance Sampling; IS) 가중치를 곱해주고, 보정을 얼마나 할 지 결정하는 하이퍼파라미터 β 를 사용한다.

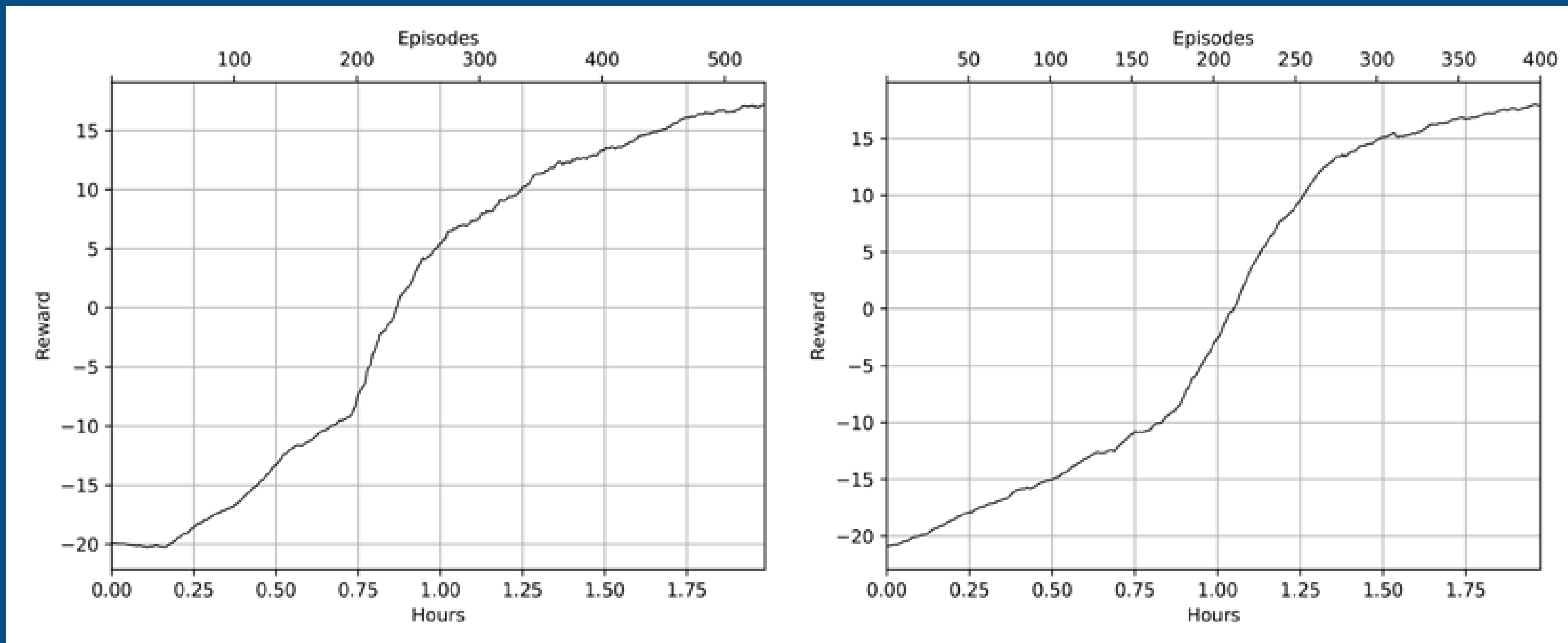
$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)} \right)^\beta$$

- β 는 0과 1사이의 값으로 1이 되면 균등하지 않은 확률 $P(i)$ 를 완전 보상하고 0이 되면 w_i 가 1이 되어 사라진다.
- 보통 강화학습에서 바이어스를 수정하는 건 수렴하게 되는 학습 후반부에 중요해진다.
따라서 논문에서는 시작 값 β_0 를 0.4나 0.5로 설정한 다음, 학습 종료 시 1이 되도록 선형적으로 증가시켰다.

Prioritized Replay Buffer

2021 DGSW
Rainbow Is All You Need

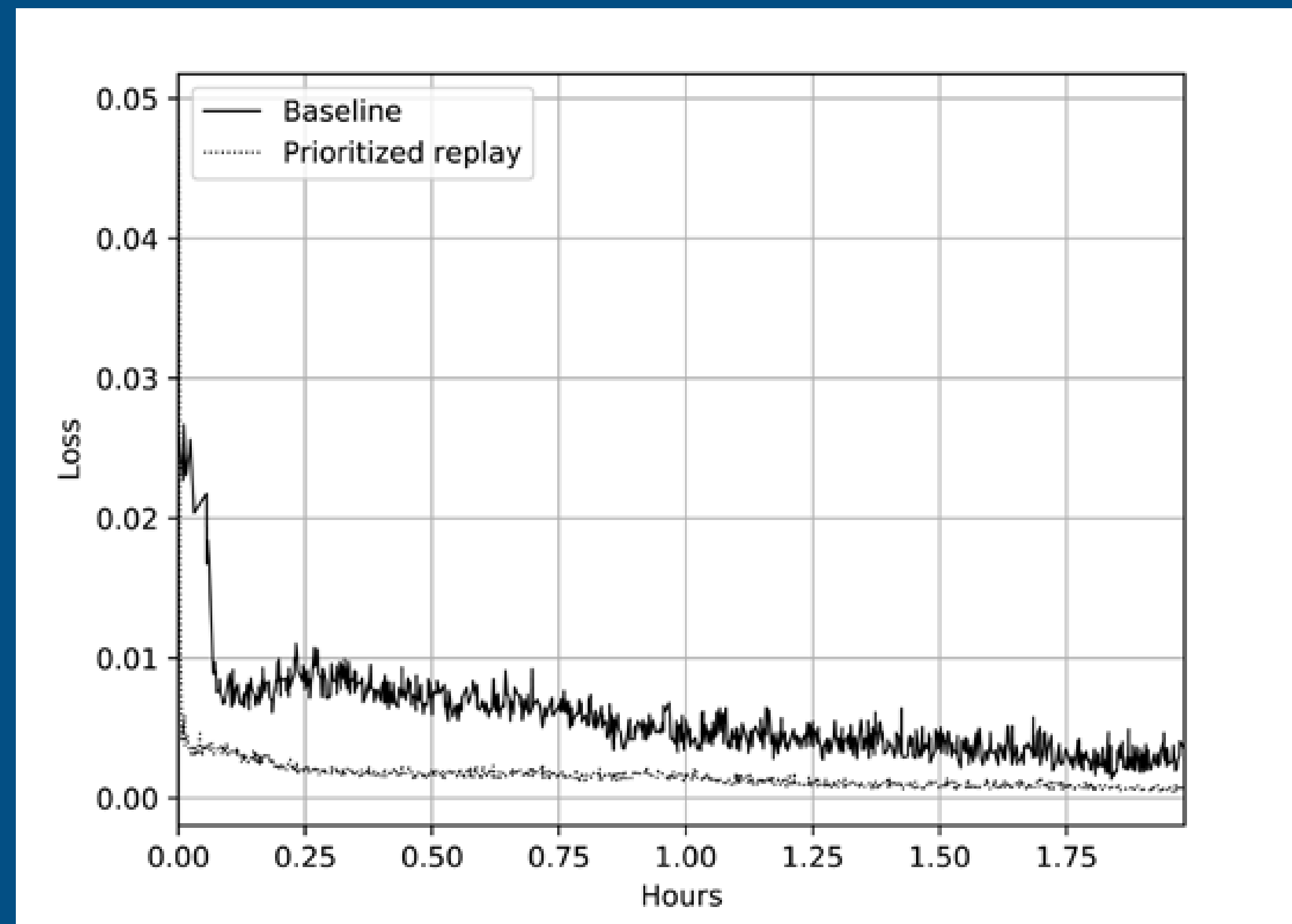
- Basic DQN vs Prioritized Replay Buffer (Reward)



Prioritized Replay Buffer

2021 DGSW
Rainbow Is All You Need

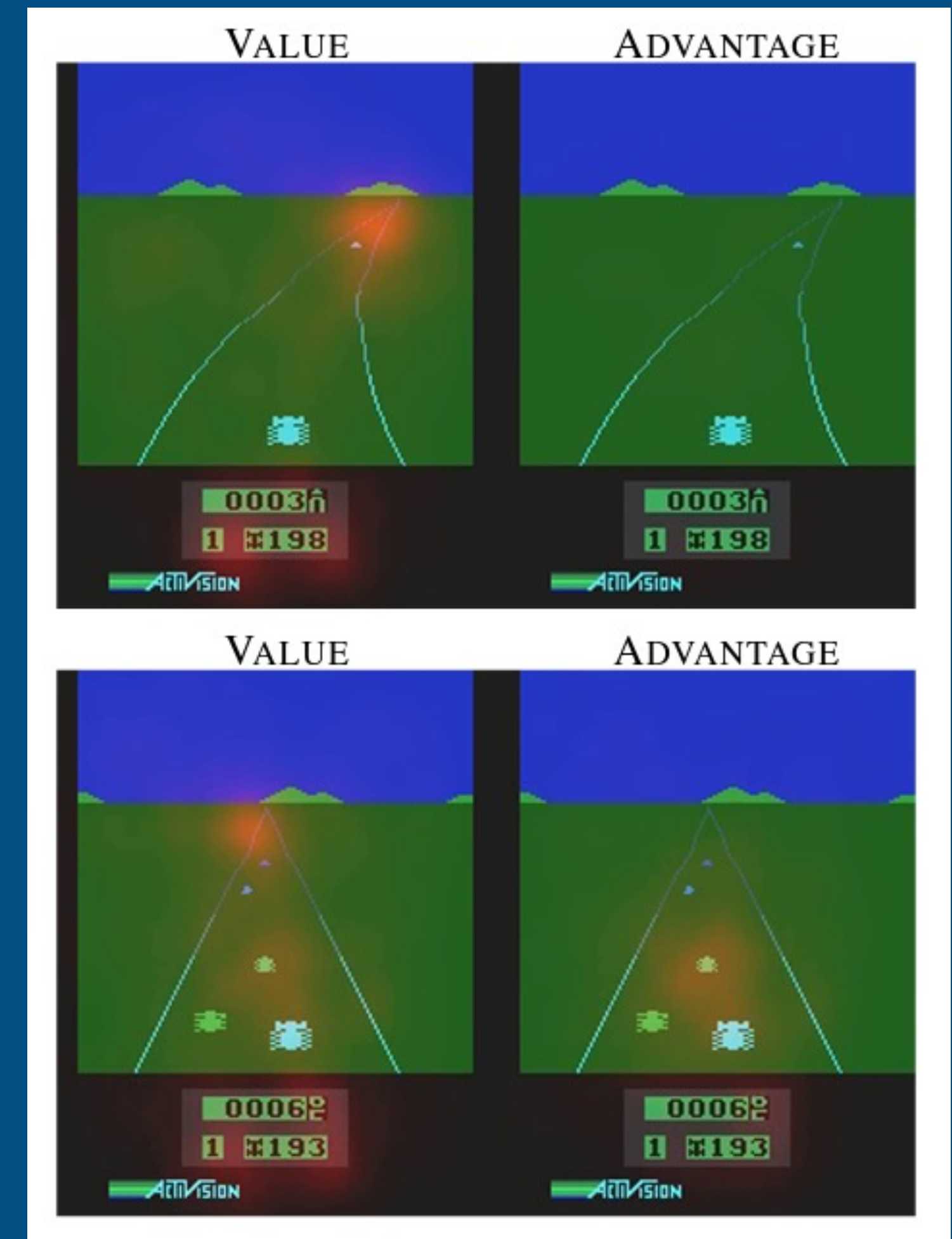
- Basic DQN vs Prioritized Replay Buffer (Loss)



Dueling DQN

2021 DGSW
Rainbow Is All You Need

- Dueling Network Architectures for Deep Reinforcement Learning (Wang, 2015)
- 위 그림 (플레이어 차와 다른 차들이 먼 상황)
 - Value Stream : 앞으로의 보상을 최대화하기 위해
1) 점수, 2) 멀리 있는 차, 3) 가야할 길에 집중한다.
 - Advantage Stream : 크게 신경쓰지 않는다.
(당장에 행동을 하지 않아도 어차피 차는 나아가고 점수는 쌓이고 있기 때문)
- 아래 그림 (플레이어 차와 다른 차들이 매우 가까운 상황)
 - Value Stream : 1) 점수, 2) 앞에 있는 차, 3) 가야할 길에 집중한다.
 - Advantage Stream : 앞에 있는 차에 집중한다.
(당장에 행동을 취하지 않으면 보상을 얻는 데 영향이 있기 때문)



Dueling DQN

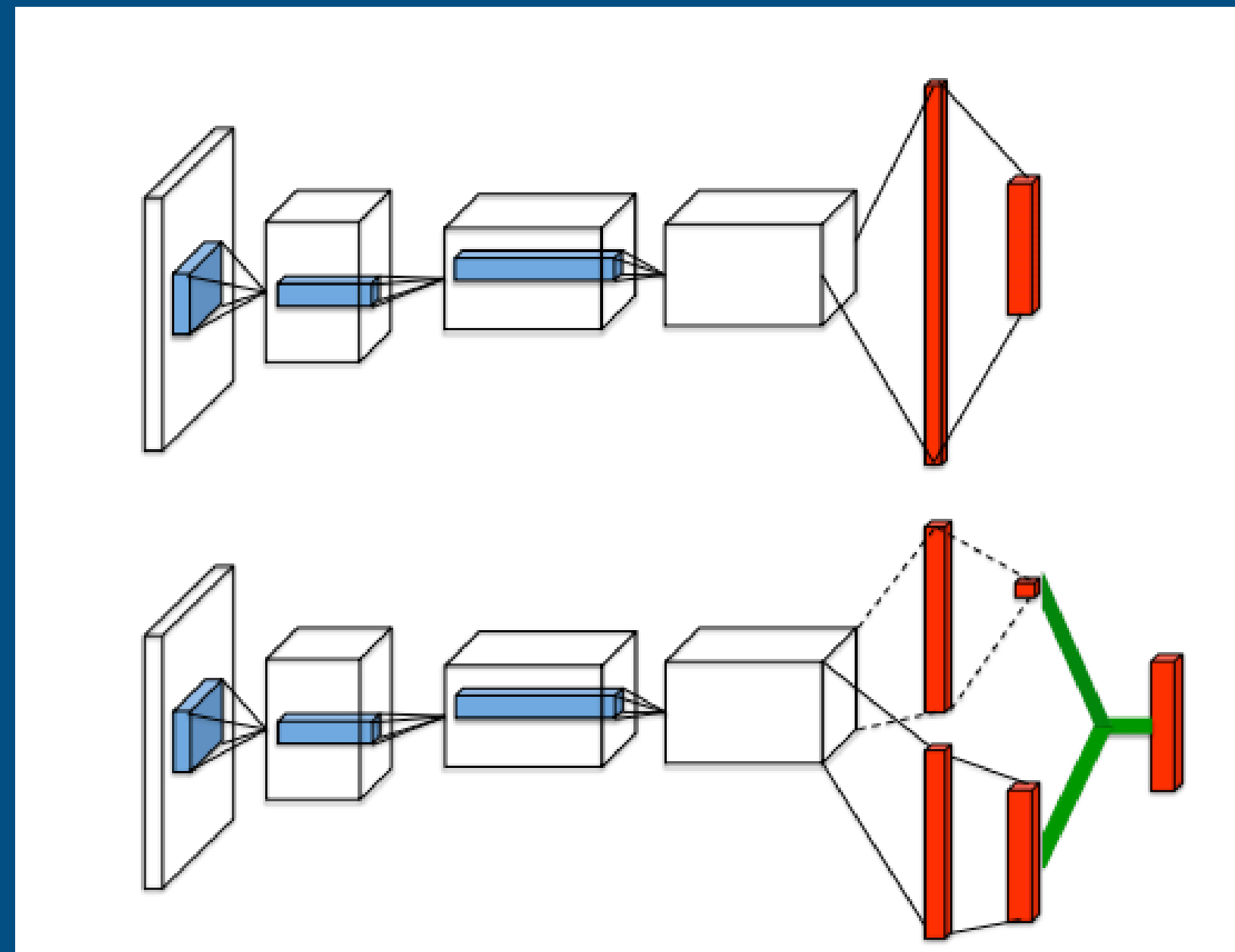
2021 DGSW
Rainbow Is All You Need

- Dueling DQN은 Q 값을 두 값으로 나눈다.

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha)$$

- $V(s)$: 상태 s 로부터 얼마나 많은 보상을 받을지 알려주는 함수 (Value)
- $A(s, a)$: 해당 행동이 다른 행동에 비해 얼마나 더 좋은지를 나타내는 함수 (Advantage)

→ 행동에 대해서 정확한 값을 알 필요 없이, 상태 가치 함수를 배우는 것만으로도 충분하다.



- 하지만, Q 값에 V 와 A 가 얼마나 영향을 줬는지 알 방법이 없다는 문제가 있다.
예를 들어 $Q = 20$ 이면 $V + A = 20$ 인데, 이때 경우의 수는 거의 무한에 가깝다.
- 이 문제를 해결하기 위해, 가장 높은 Q 값($Q(s, a^*)$)을 $V(s)$ 와 같게 한다.
즉, Advantage 함수의 최댓값을 0으로 만들고 다른 모든 값을 음수로 만든다.
이렇게 하면 V 값을 정확하게 알 수 있으므로 문제를 해결할 수 있다.

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \max_{a' \in |\mathcal{A}|} A(s, a'; \theta, \alpha) \right)$$

- 논문에서는 max 연산자를 평균으로 대체해서 사용한다.

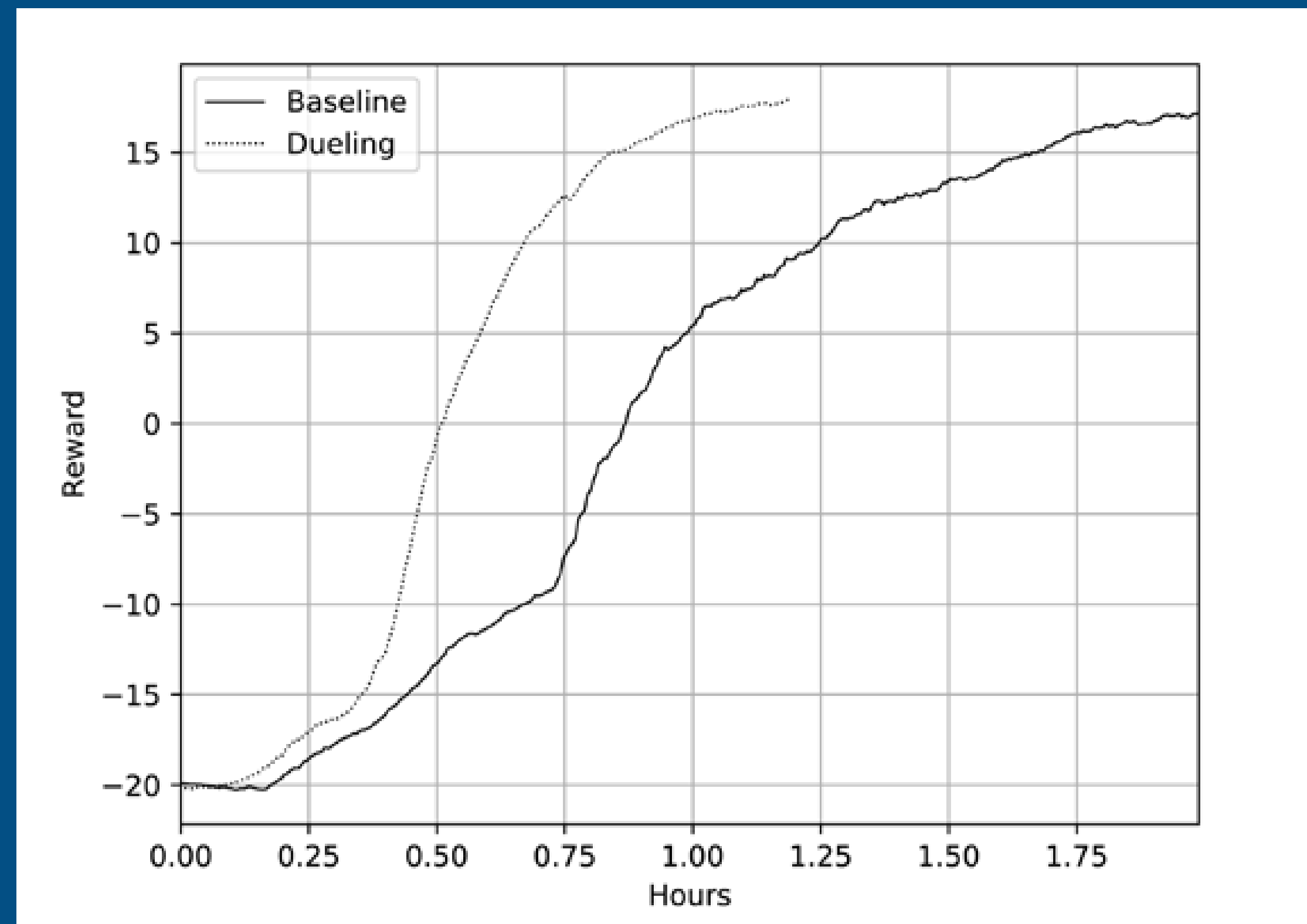
$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \frac{1}{|\mathcal{A}|} A(s, a'; \theta, \alpha) \right)$$

- 원래 수식을 사용하면 V 와 A 의 의미를 갖게 되지만 평균으로 대체하면 의미를 잃게 된다.
하지만 상수로 고정해 놓은 목표가 아니기 때문에, 최적화의 안정성이 증가하게 된다.
- 실제로 두 수식을 같이 실험하면 유사한 결과가 나온다.

Dueling DQN

2021 DGSW
Rainbow Is All You Need

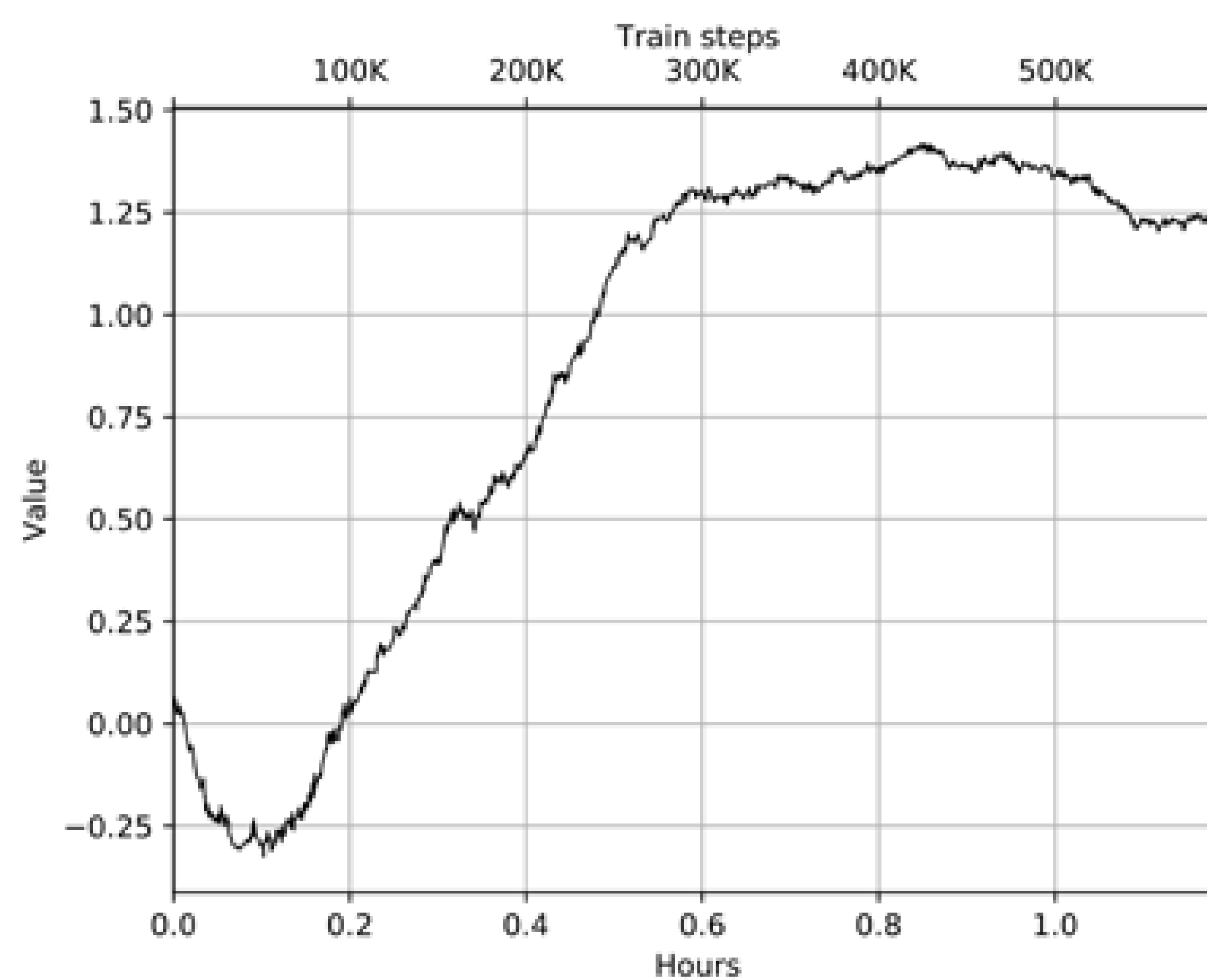
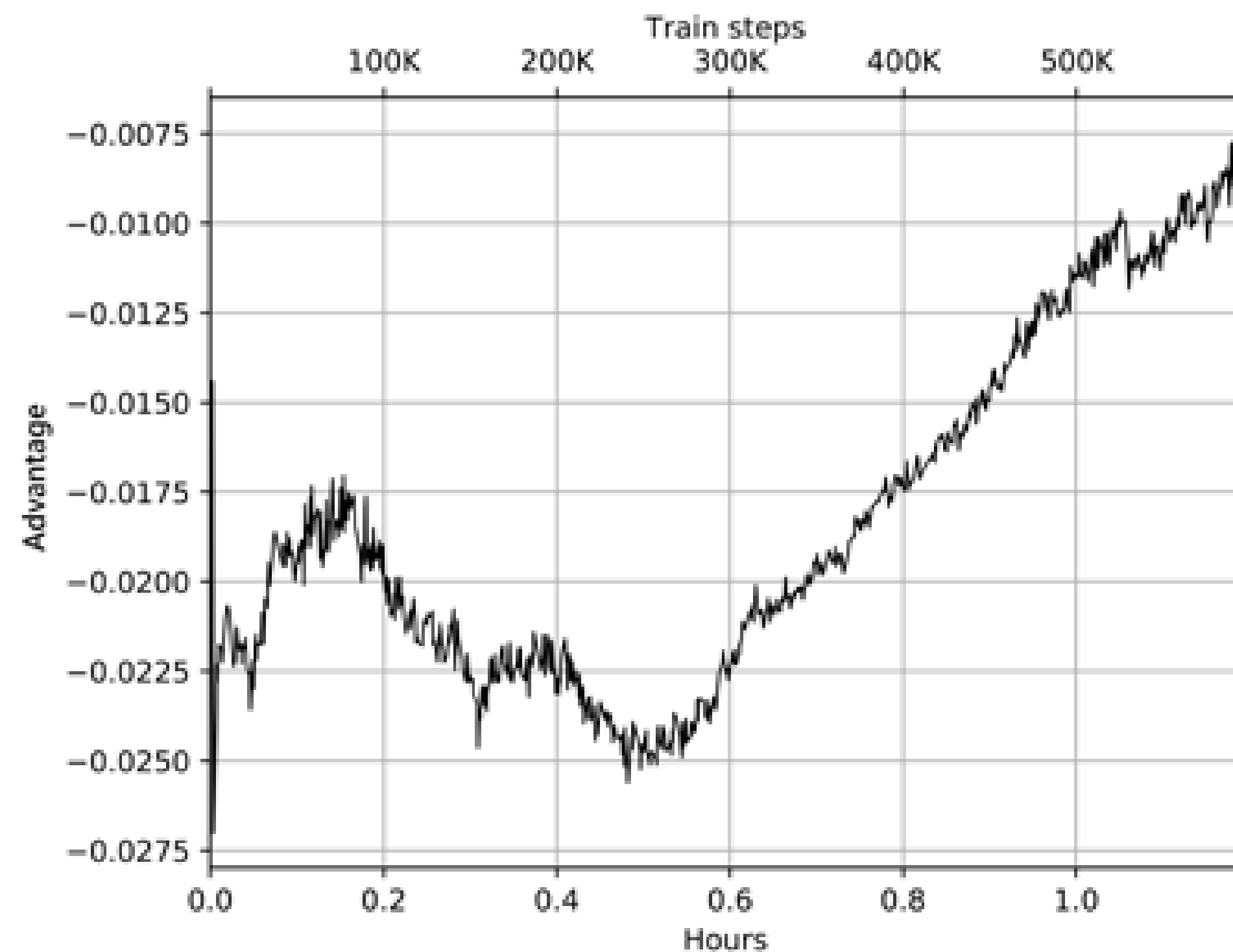
- Basic DQN vs Dueling DQN (Reward)



Dueling DQN

2021 DGSW
Rainbow Is All You Need

- Dueling DQN (Advantage, Value)



Categorical DQN

2021 DGSW
Rainbow Is All You Need

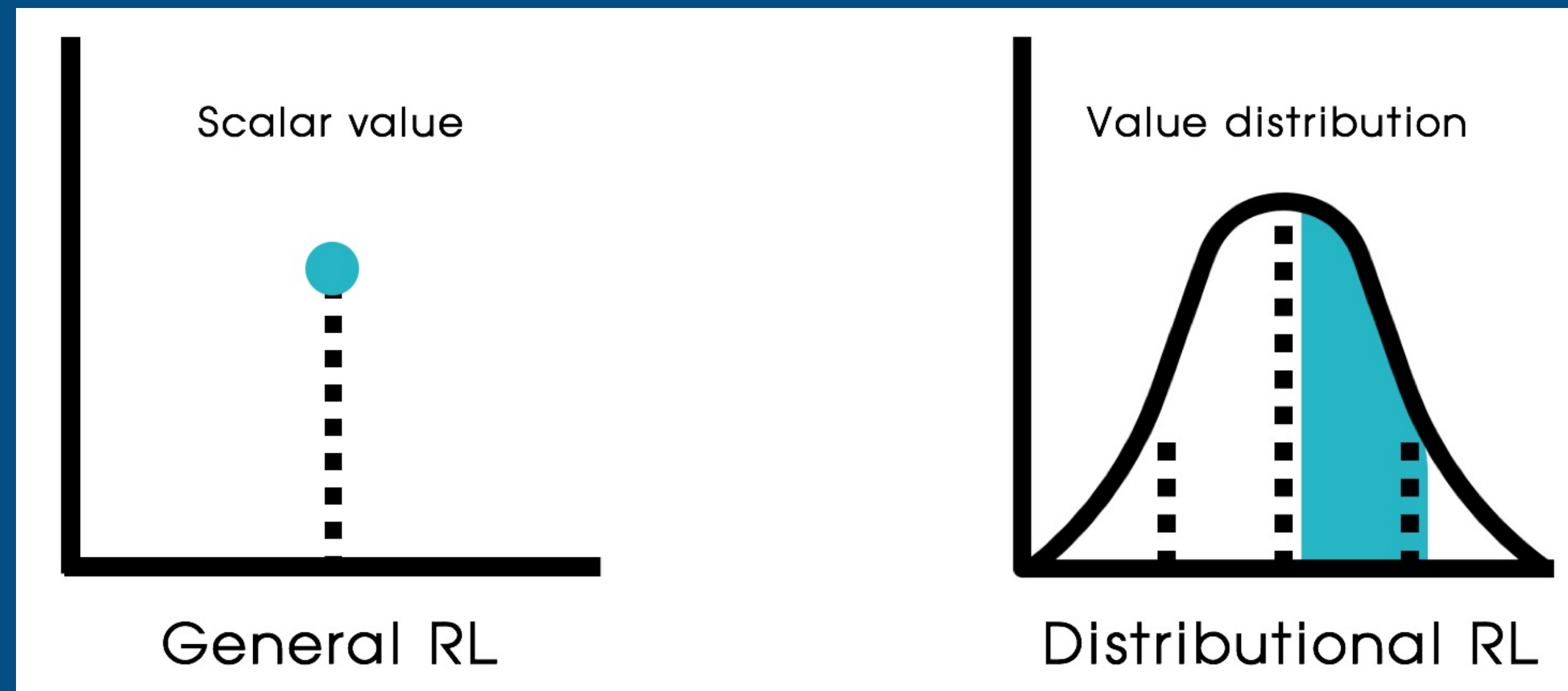
- A Distributional Perspective on Reinforcement Learning (Bellemare, 2017)

- General RL : 가치를 하나의 스칼라 값으로 예측

$$Q(x, a) = \mathbb{E}R(x, a) + \gamma \mathbb{E}Q(X', A')$$

- Distributional RL : 가치를 분포로 예측

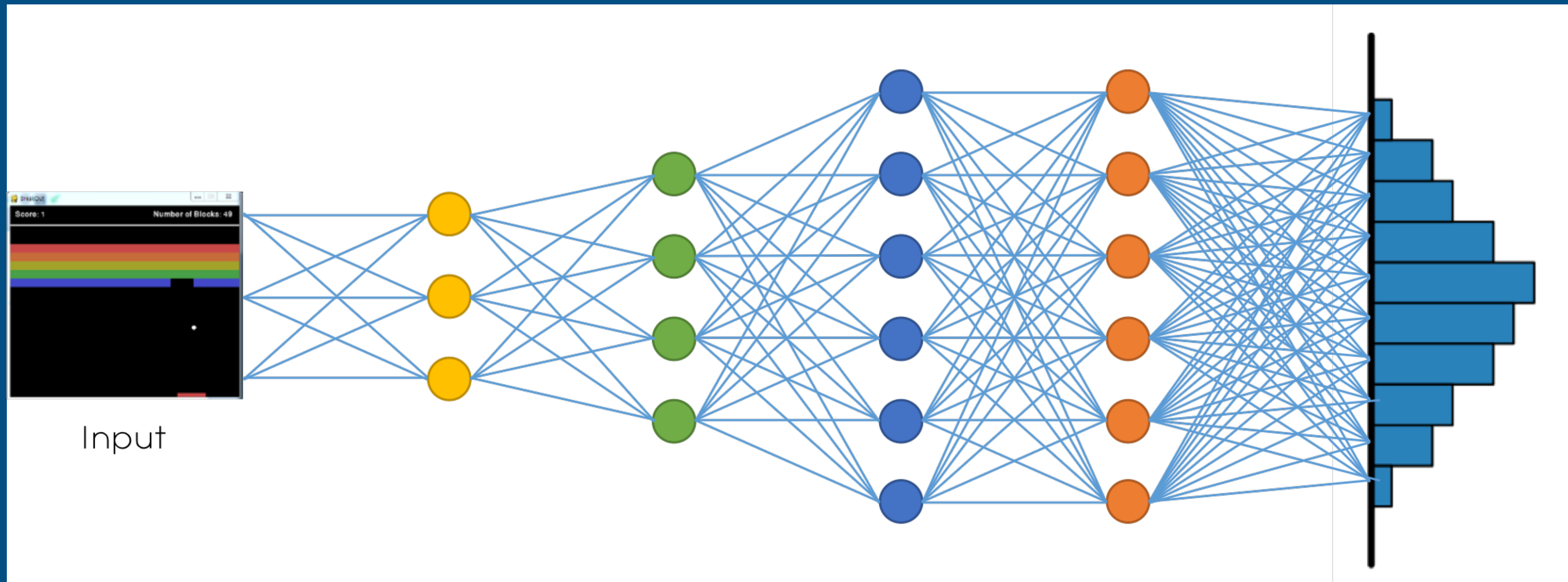
$$Z(x, a) = R(x, a) + \gamma Z(X', A')$$



Categorical DQN

2021 DGSW
Rainbow Is All You Need

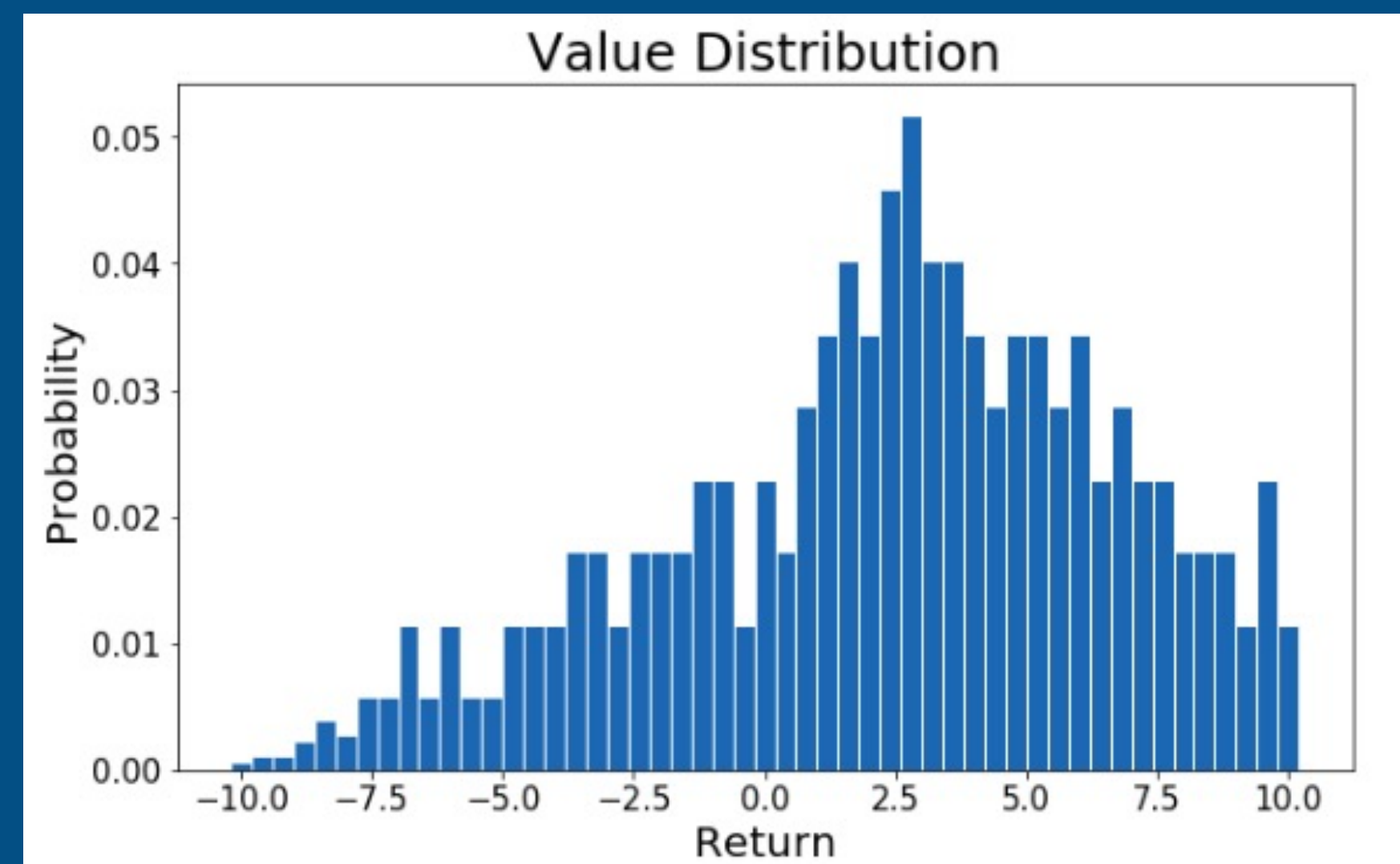
- DQN에서 네트워크의 출력 : 각 행동에 대한 Q 값
- Distributional RL에서 네트워크의 출력 : 각 행동에 대한 가치 분포



Categorical DQN

2021 DGSW
Rainbow Is All You Need

- 각 행동에 대한 가치 분포 = 이산 확률 분포 (Discrete Probability Distribution)
 - 가로축 : Support 또는 Atom (가치 값), 세로축 : 확률 \rightarrow 각각의 가치와 그 가치를 받을 확률을 나타내는 분포
- 이 분포를 결정하기 위해서 몇 가지 매개 변수가 필요하다.
 - Support의 개수, Support의 최댓값, Support의 최솟값
 - Support 값은 최솟값부터 최댓값까지 Support의 개수에 맞게 일정한 간격으로 나누게 된다.
즉, 미리 결정된 매개 변수들에 의해 그 값이 정해지게 된다. 신경망은 바로 이 Support들에 대한 확률을 구한다.
각 행동에 대해서 하나의 분포가 필요하기 때문에 신경망의 출력 크기는 [Support의 개수 * 행동의 개수]가 된다.



- Categorical DQN의 알고리즘은 DQN과 유사하다. 차이점은 다음과 같다.
- Q 값 계산 : 이산 확률 분포의 기댓값

$$Q(x_{t+1}, a) = \sum_i z_i p_i(x_{t+1}, a)$$

- Loss 계산 : 타겟 분포와 추정 분포 간의 차이를 줄이는 방향으로 학습
→ 크로스 엔트로피 (Cross Entropy)

$$Loss = - \sum_i m_i \log p_i(x_t, a_t)$$

- 타겟 분포 : 우선 Support 값들을 통해 타겟 값을 구한다.

$$\hat{J}z_j \leftarrow [r_t + \gamma_t z_j]_{V_{\text{MIN}}}^{V_{\text{MAX}}}$$

- 각 Support에 감가율을 곱하고 보상을 더해준다. (에피소드가 끝난 경우에는 모든 Support 값들을 보상 값으로 사용) 그리고 최댓값보다 큰 경우 최댓값과 같도록, 최솟값보다 작은 경우 최솟값과 같도록 설정한다.
- 그런데 이 경우 문제가 생길 수 있다.
 - 예를 들어, Support들이 [1, 2, 3, 4, 5]이고 $r_t = 0.1$, $\gamma_t = 0.9$ 라고 가정하자. 위 식에 따라 연산을 하면 Support들이 [1, 1.9, 2.8, 3.7, 4.6]이 된다.
 - Loss인 크로스 엔트로피 연산을 하기 위해서는 두 분포의 Support들이 일치해야 하는데, 현재는 서로 다르다.
→ 타겟 분포의 Support들을 원래의 Support들과 같이 분배해주는 Projection 과정이 추가로 필요하다.

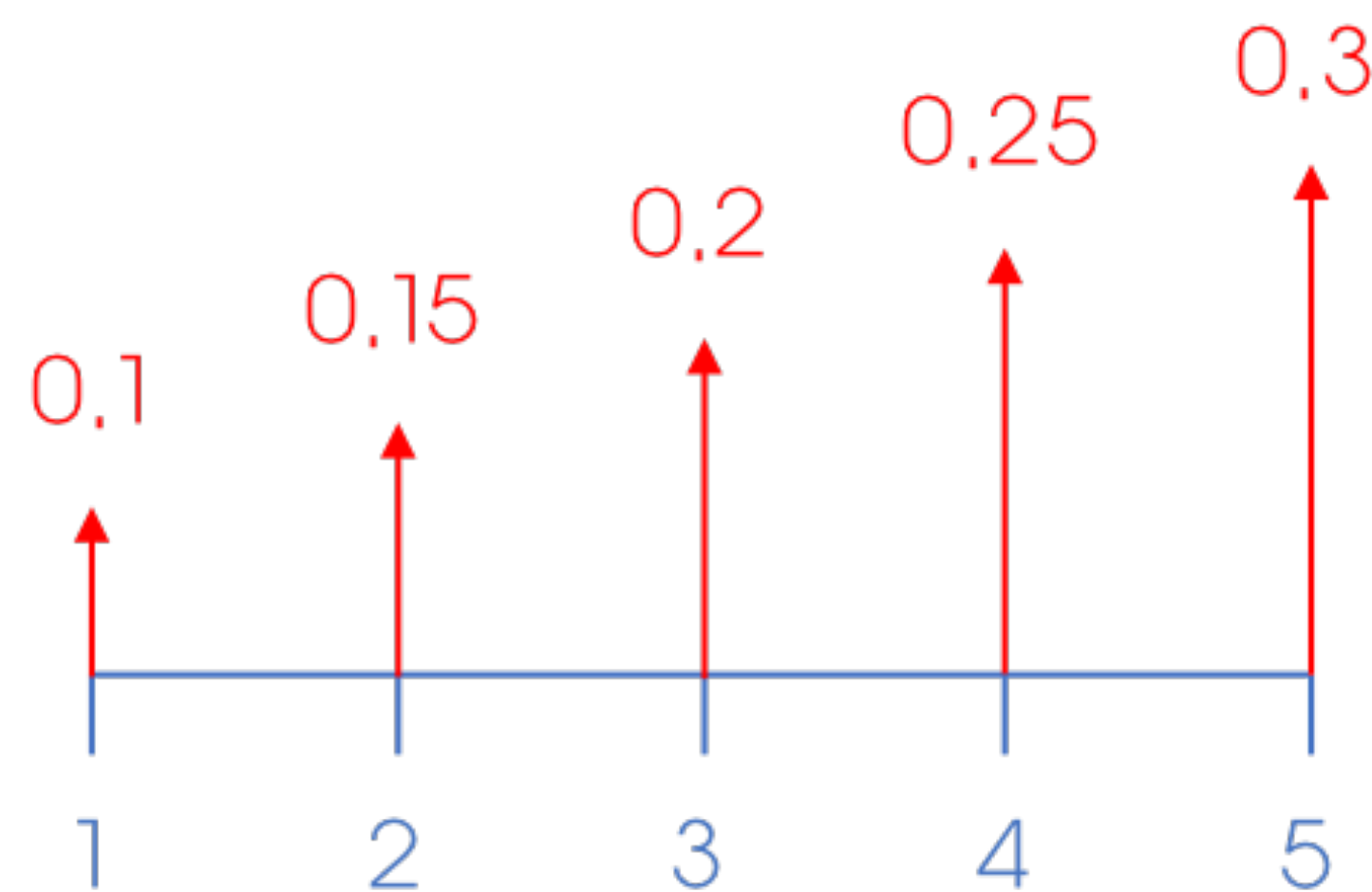
$$m_l \leftarrow m_l + p_j(x_{t+1}, a^*)(u - b_j)$$

$$m_u \leftarrow m_u + p_j(x_{t+1}, a^*)(b_j - l)$$

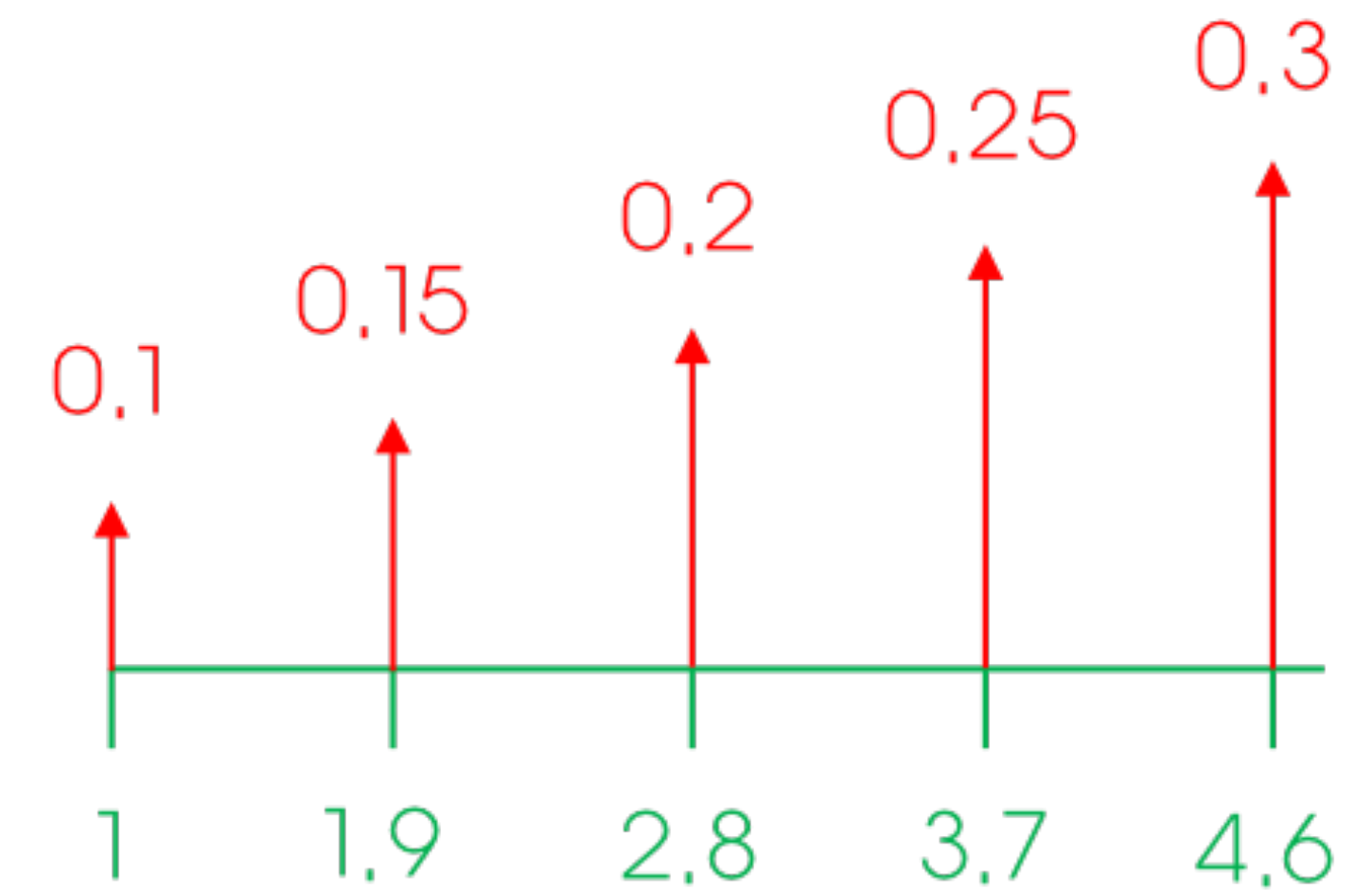
Categorical DQN

2021 DGSW
Rainbow Is All You Need

- Support: [1, 2, 3, 4, 5]
- Probability: [0.1, 0.15, 0.2, 0.25, 0.3]
- Reward: 0.1
- Discount factor: 0.9

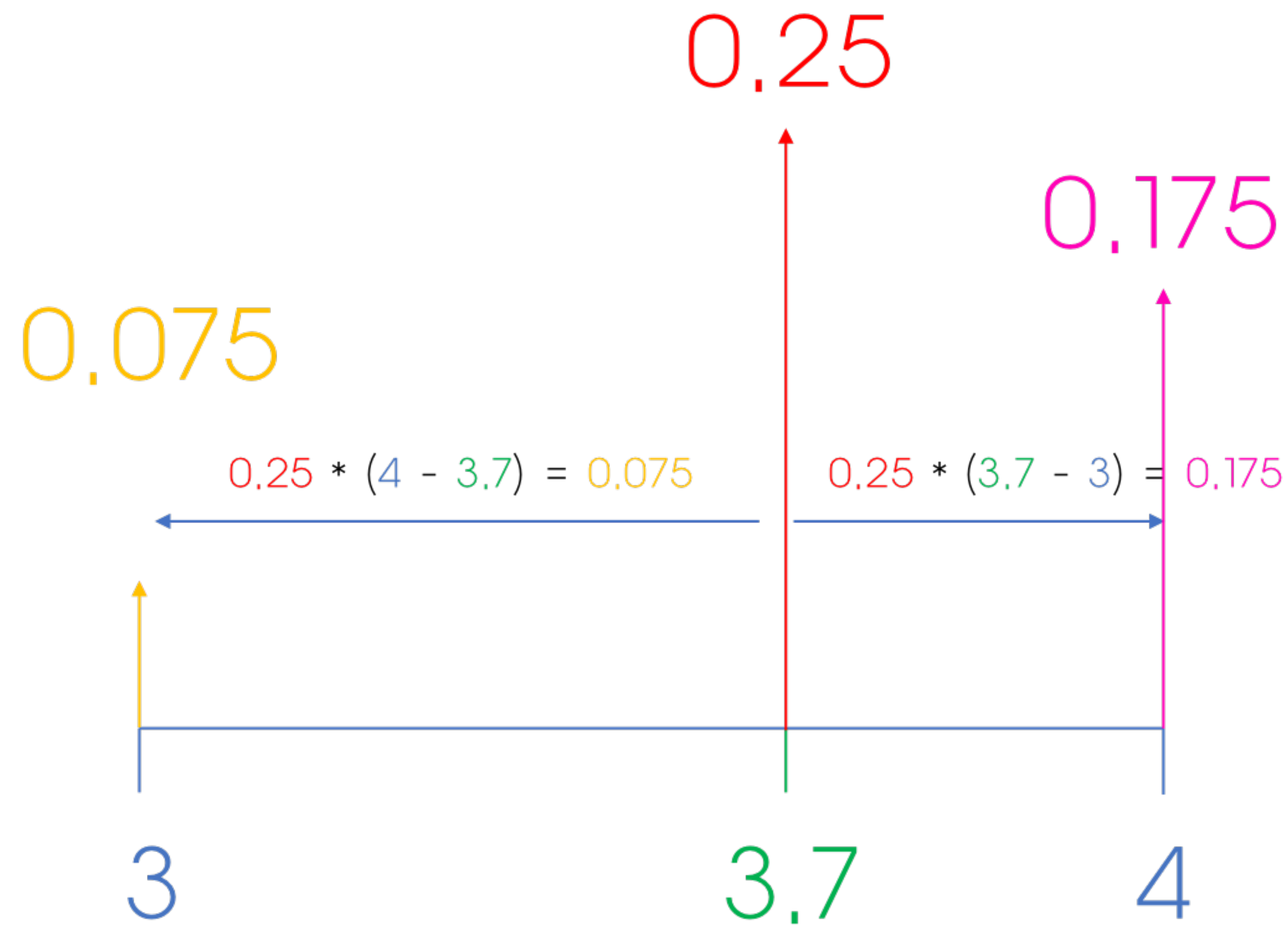


$$\hat{\mathcal{T}}z_i \leftarrow [r_t + \gamma z_i]_{V_{min}}^{V_{max}}$$



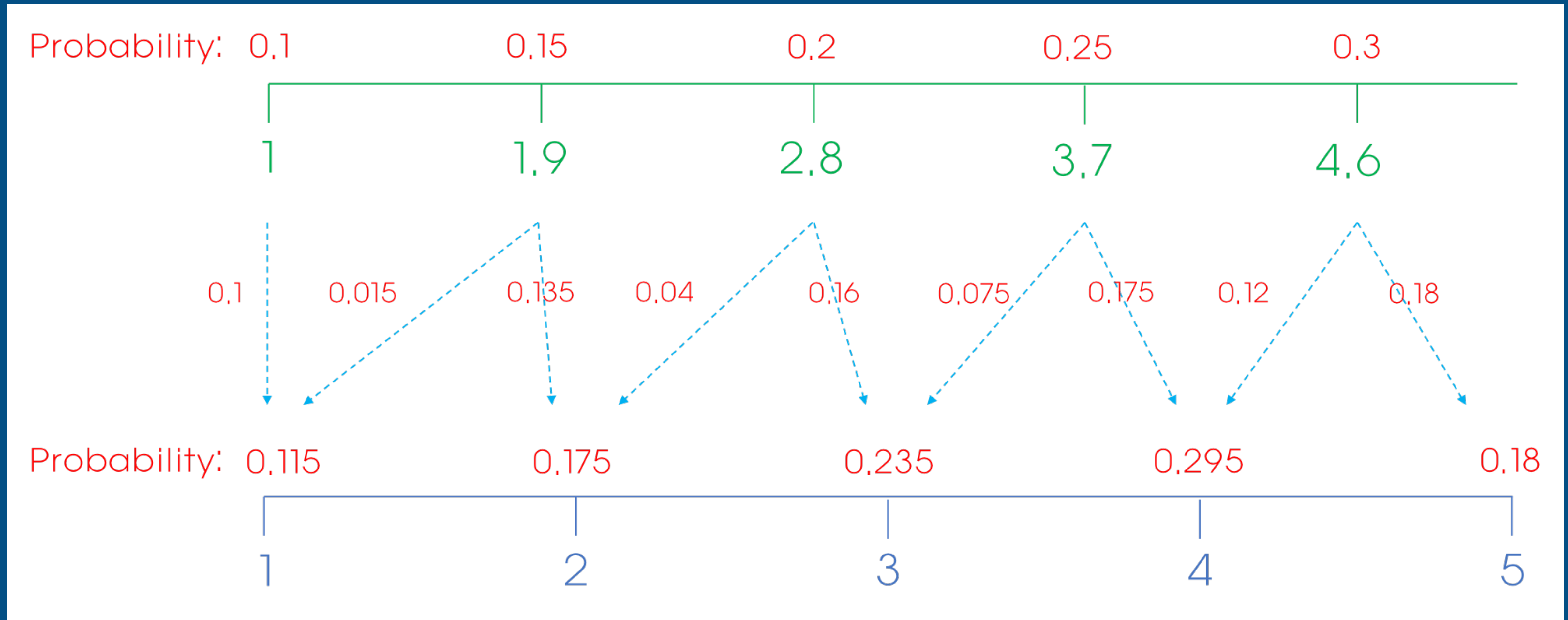
Categorical DQN

2021 DGSW
Rainbow Is All You Need



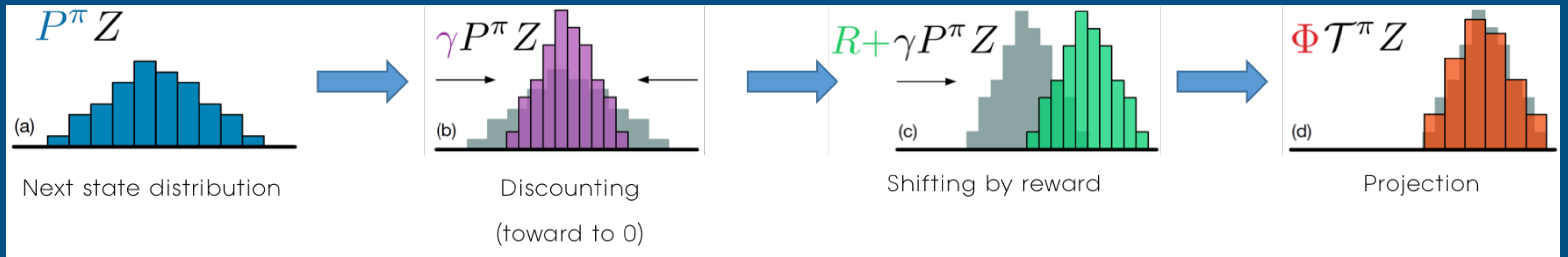
Categorical DQN

2021 DGSW
Rainbow Is All You Need



Categorical DQN

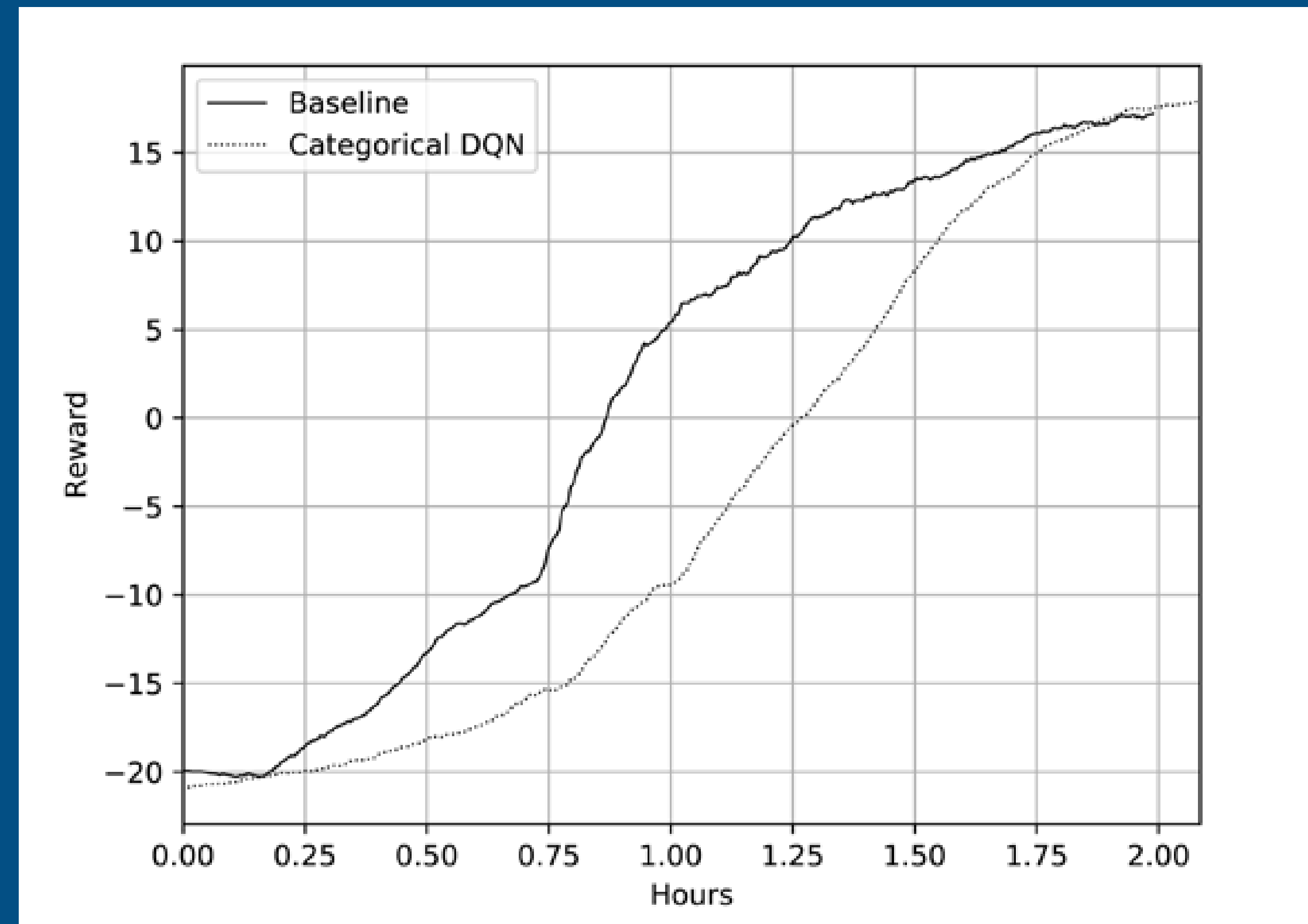
2021 DGSW
Rainbow Is All You Need



Categorical DQN

2021 DGSW
Rainbow Is All You Need

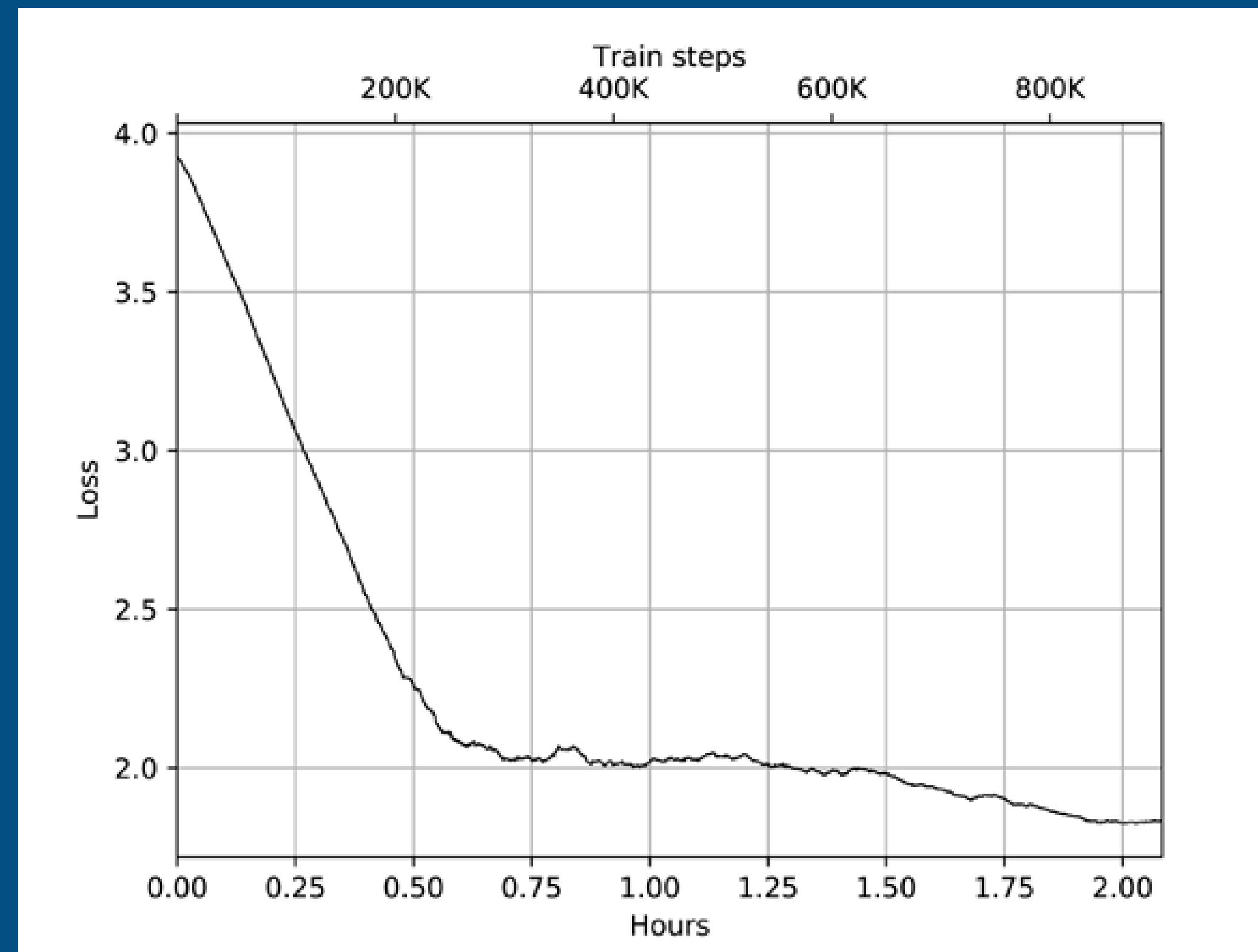
- Basic DQN vs Categorical DQN (Reward)



Categorical DQN

2021 DGSW
Rainbow Is All You Need

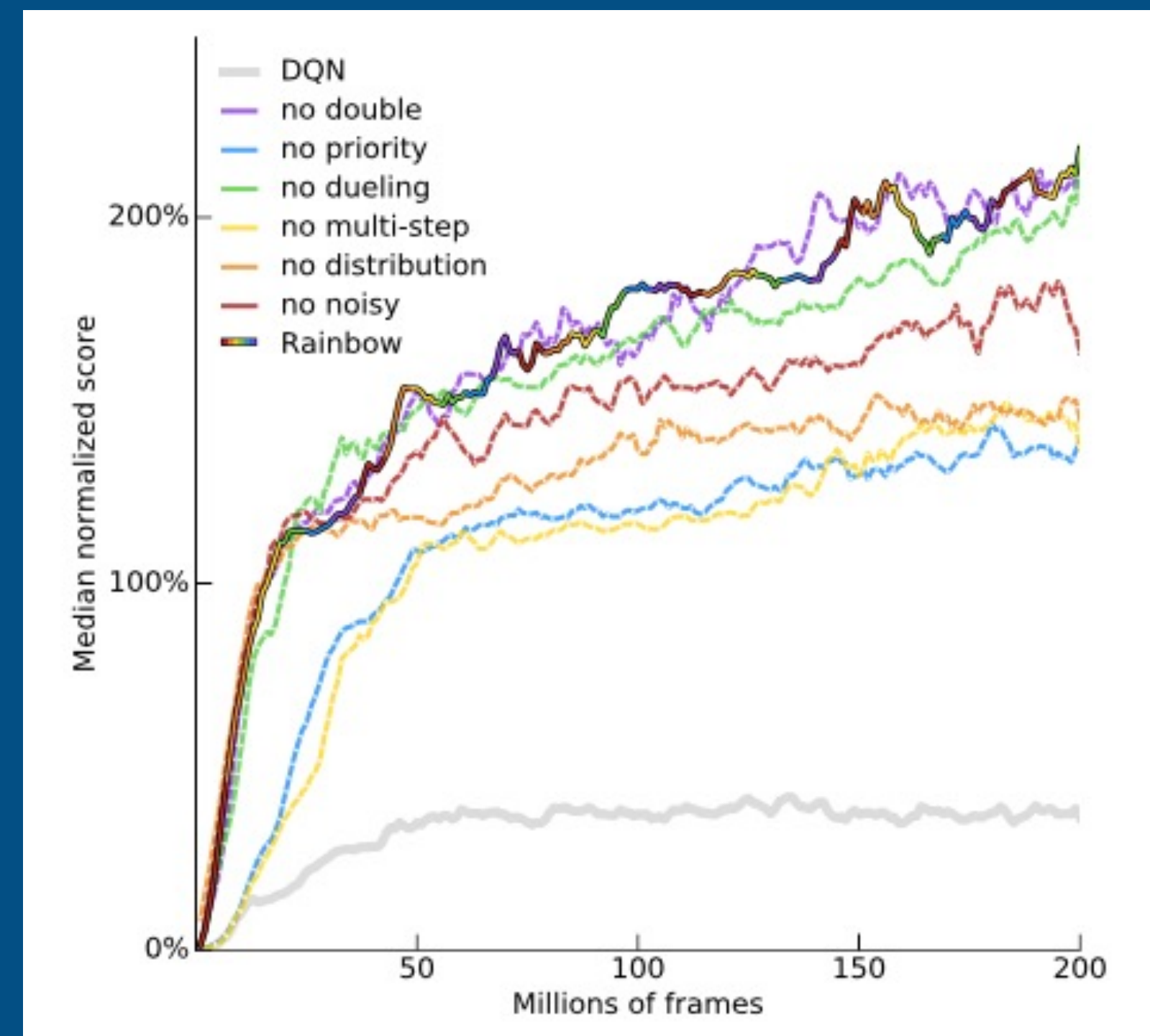
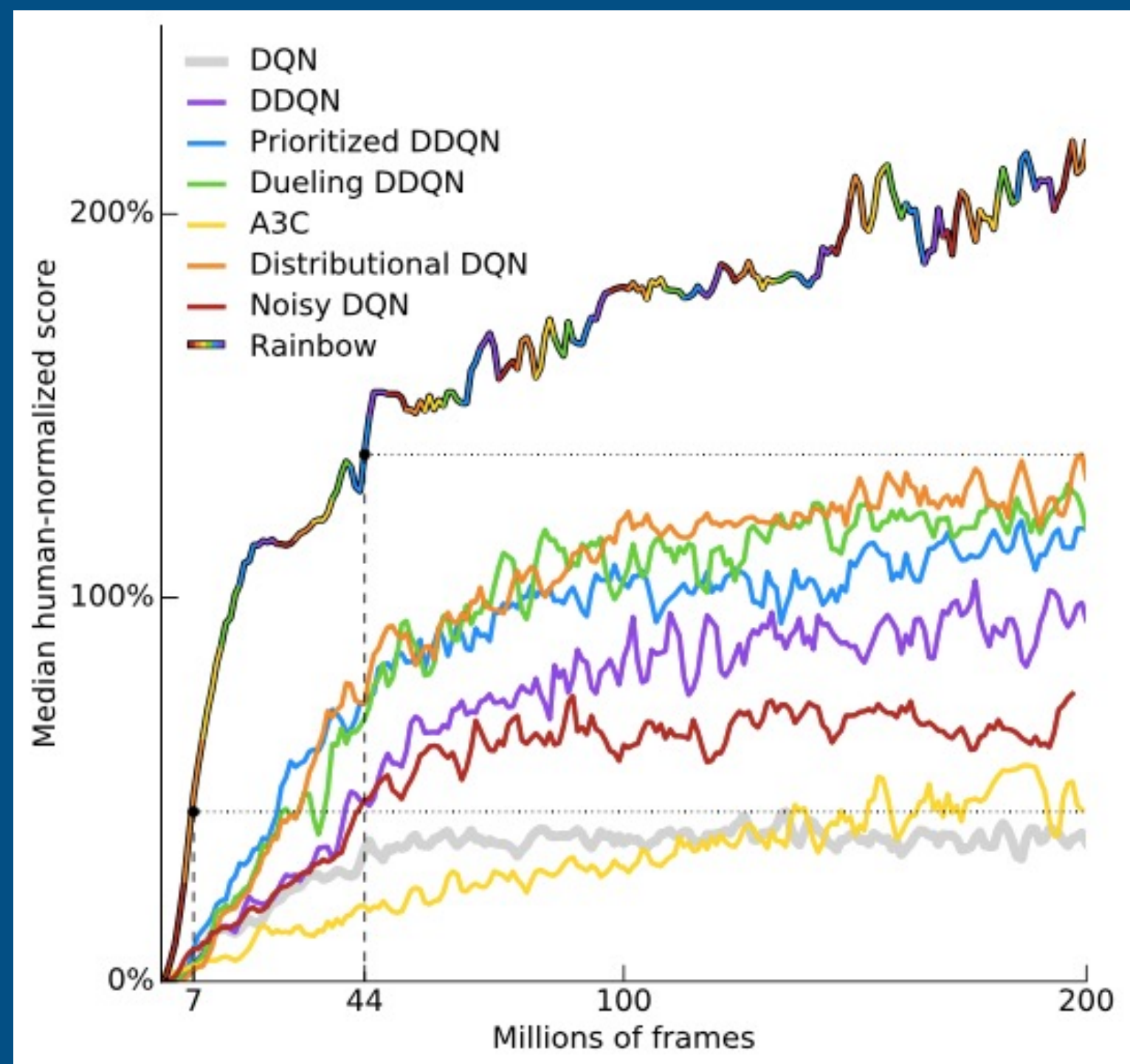
- Categorical DQN (Loss)



Rainbow DQN

2021 DGSW
Rainbow Is All You Need

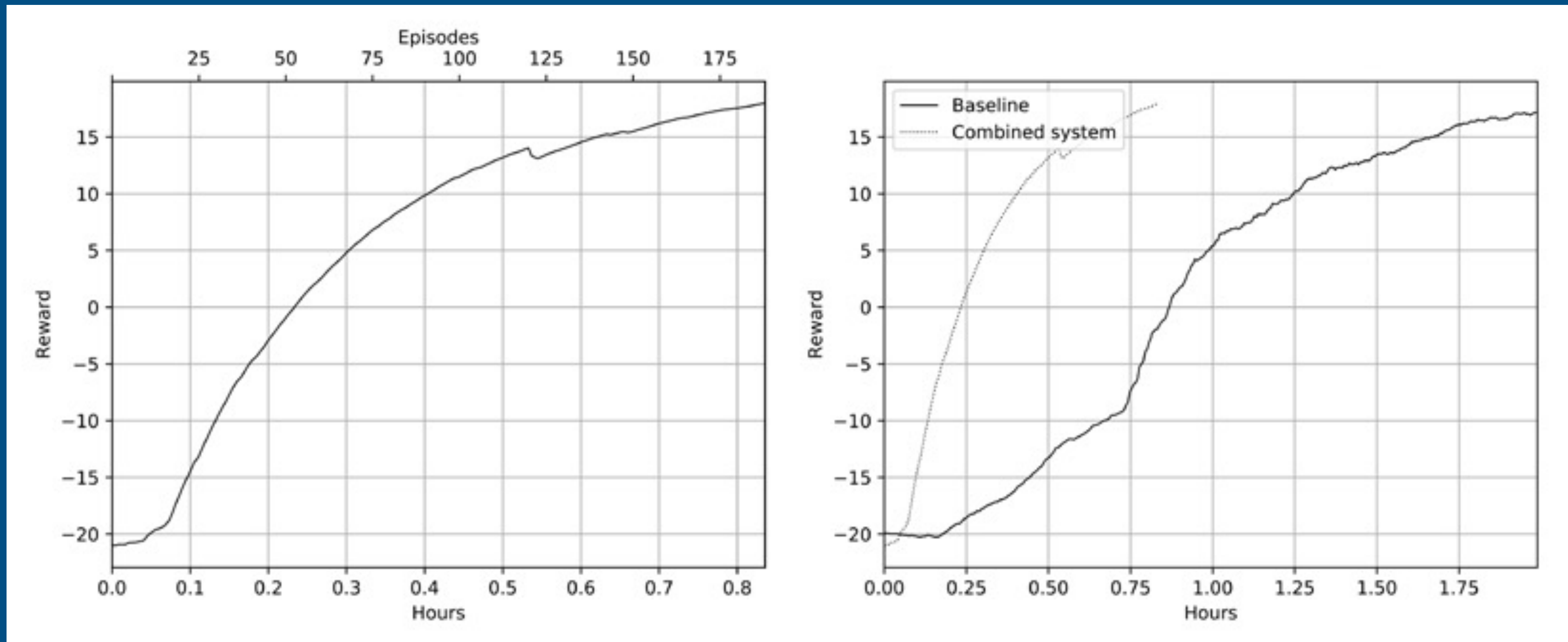
- Rainbow: Combining Improvements in Deep Reinforcement Learning
- Rainbow = DQN + Multi-step DQN + Double DQN (DDQN) + Noisy Network + Prioritized Experience Replay (PER) + Dueling DQN + Categorical DQN



Rainbow DQN

2021 DGSW
Rainbow Is All You Need

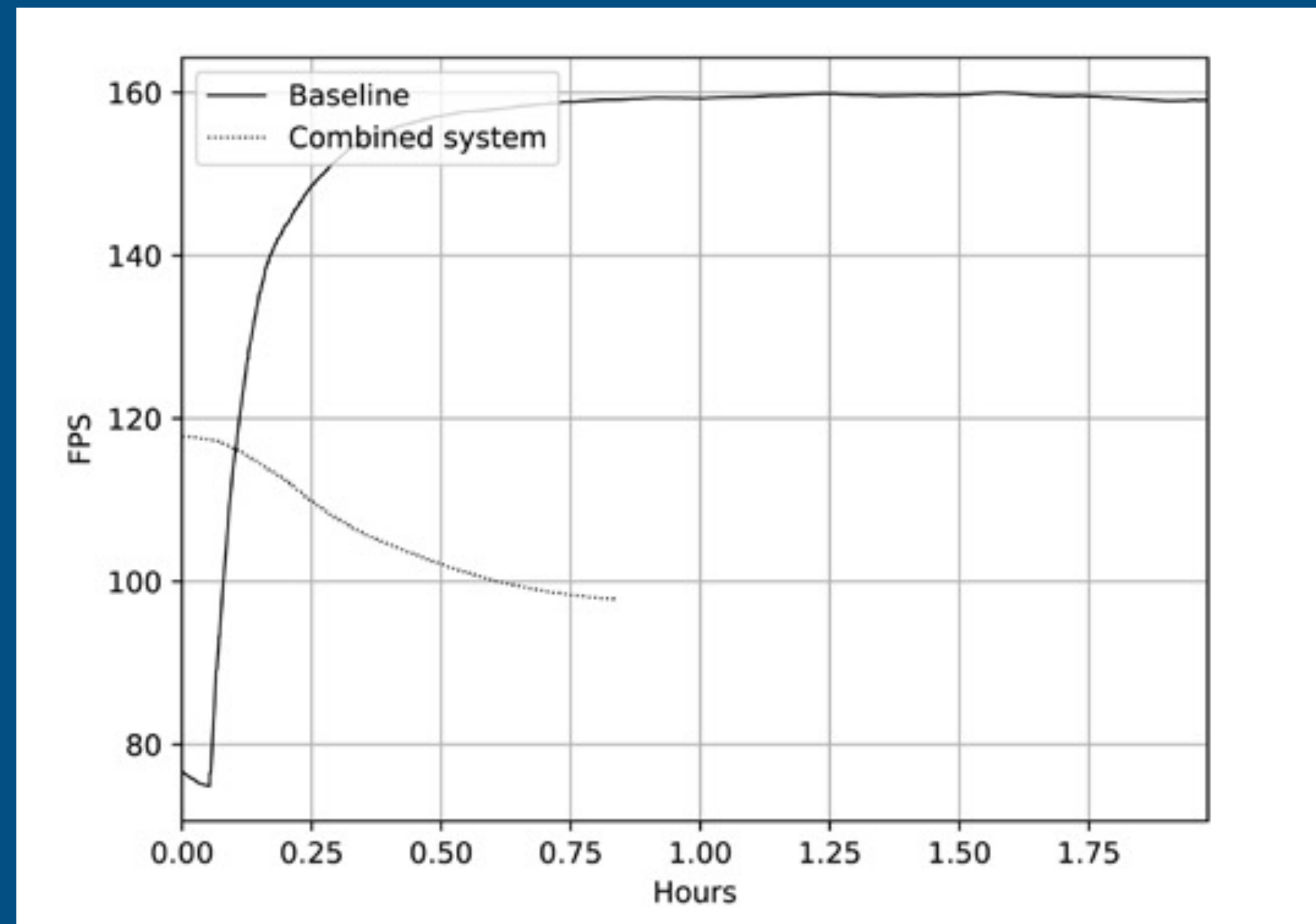
- Basic DQN vs Rainbow DQN (Reward)



Rainbow DQN

2021 DGSW
Rainbow Is All You Need

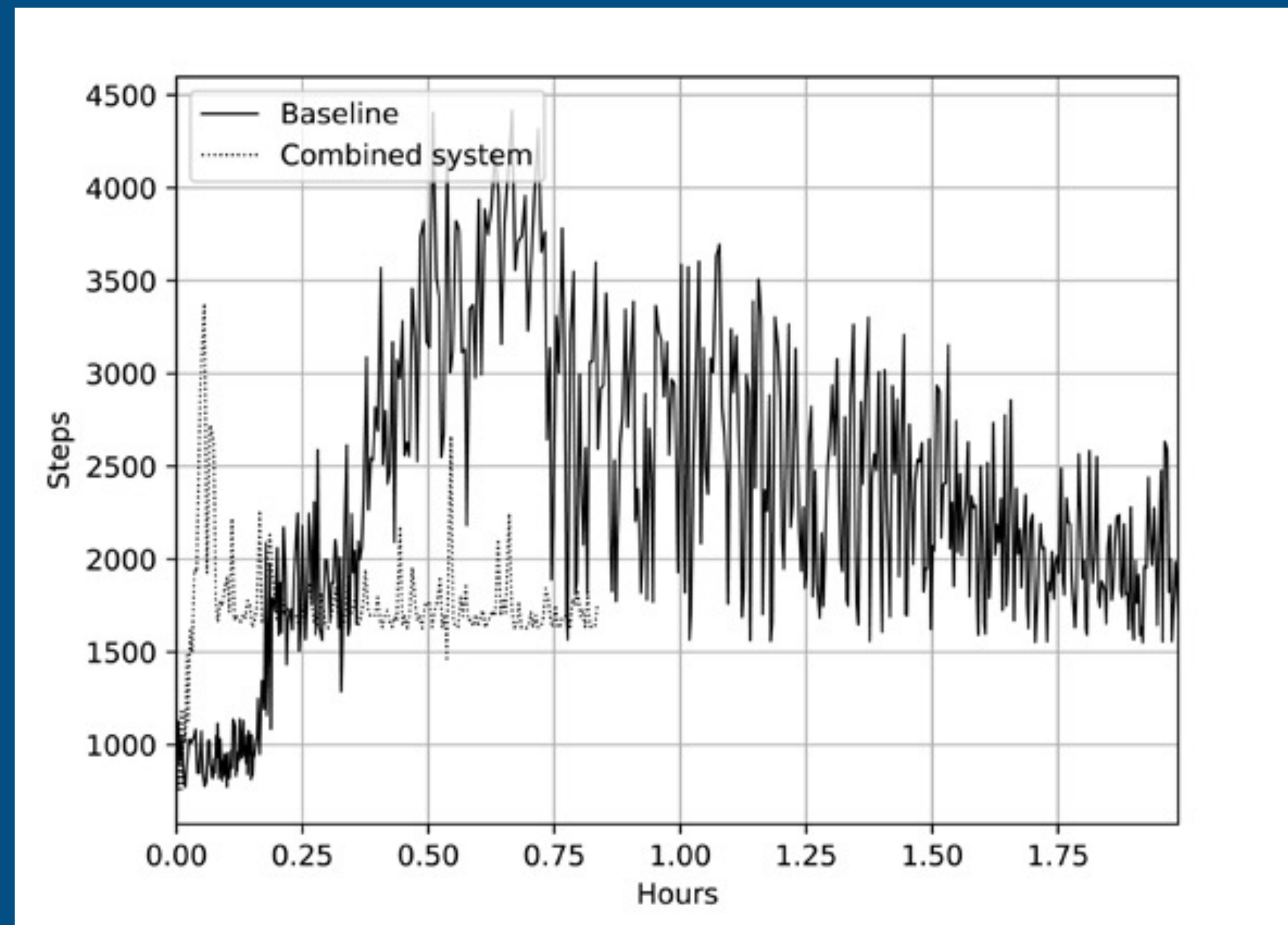
- Basic DQN vs Rainbow DQN (FPS)



Rainbow DQN

2021 DGSW
Rainbow Is All You Need

- Basic DQN vs Rainbow DQN (Steps)



감사합니다!

스터디 듣느라 고생 많았습니다.