Q형님과 학습하기

점심 먹었으니까 다시한번..

정책 이터레이션

가치 이터레이션

벨만 기대 방정식

벨만 최적 방정식

정책 이터레이션

가치 이터레이션

벨만 기대 방정식

벨만 최적 방정식

현재 정책에 대해 가치함수로써 평가

최적 정책 가정

정책 이터레이션

벨만 기대 방정식

현재 정책에 대해 가치함수로써 평가

그 평가로 정책을 업데이트

e-그리디 정책 발전

가치 이터레이션

벨만 최적 방정식

최적 정책 가정

가치함수를 통해 행동 선택

DP 한계

계산 복잡도

차원의 저주

환경을 몰라

DP 한계

계산 복잡도

차원의 저주

~~환경을 몰라~~ → Model-free 하고싶다

모델을 몰라도 학습하고 싶으면?

내 행동에 대한 **결과**를 객관적으로 **평가**해야해

평가 결과에 따라 더 나은 행동을 선택해야해

모델을 몰라도 학습하고 싶으면?

환경과 상호작용을 통해 주어진 **정책에 대한 가치함수 학습**

가치함수를 토대로 정책을 발전시켜 최적의 정책을 학습

모델을 몰라도 학습하고 싶으면?

예측

제어

모델을 몰라도 학습하고 싶으면?

MC

TD

SARSA

on-policy

Q learning

off-policy

근사란...

가위 바위 보

근사란...

가위	바위	보
1/3	1/3	1/3

근사란...

샘플링을 통해 **실제 값**을 **예측**

근사란...

샘플링을 통해 **실제 값**을 **예측**

샘플의 평균 가치함수 추정

근사란...

샘플링을 통해 **실제 값**을 **예측**

샘플의 평균 가치함수 추정

1 에피소드 몬테카를로 근사

MC예측에서 가치함수의 업데이트 식

$$V(s) \leftarrow V(s) + \frac{1}{n}(G(s) - V(s))$$

MC예측에서 가치함수의 업데이트 식

오차

$$V(s) \leftarrow V(s) + \frac{1}{n}(G(s) - V(s))$$

스텝 사이즈

반환값

MC예측에서 가치함수의 업데이트

한 상태의 **가치함수**는
에이전트가 그 상태를 **거칠 때 마다** 업데이트

MC예측에서 가치함수의 업데이트

모든 가치함수는
에이전트가 **한 에피소드 종료 후** 업데이트

MC예측에서 가치함수의 업데이트

모든 가치함수는
에이전트가 **한 에피소드 종료 후** 업데이트

그럼 real-time은요!

TD Temporal-Difference

모든 가치함수는
에이전트가 **한 타임스텝 종료 후** 업데이트

그렇군요!

TD Temporal-Difference

기댓값을 계산 X

샘플링을 통해 현재의 가치함수 업데이트

TD Temporal-Difference

기댓값을 계산 X

샘플링을 통해 현재의 가치함수 업데이트

$$V(S_t) \leftarrow V(S_t) + A\,(R(S_t) + rV(S_{t+1}) - V(S_t))$$

TD Temporal-Difference

기댓값을 계산 X

샘플링을 통해 현재의 가치함수 업데이트

$$V(S_t) \leftarrow V(S_t) + A\,(R(S_t) + rV(S_{t+1}) - V(S_t))$$

업데이트 목표

업데이트 크기

TD Temporal-Difference

기댓값을 계산 X

샘플링을 통해 현재의 가치함수 업데이트

한번에 **하나의 가치함수**를 업데이트

TD Temporal-Difference

기댓값을 계산 X

샘플링을 통해 현재의 **Q함수** 업데이트

가치함수가 아닌 **Q**를 보고 판단한다면

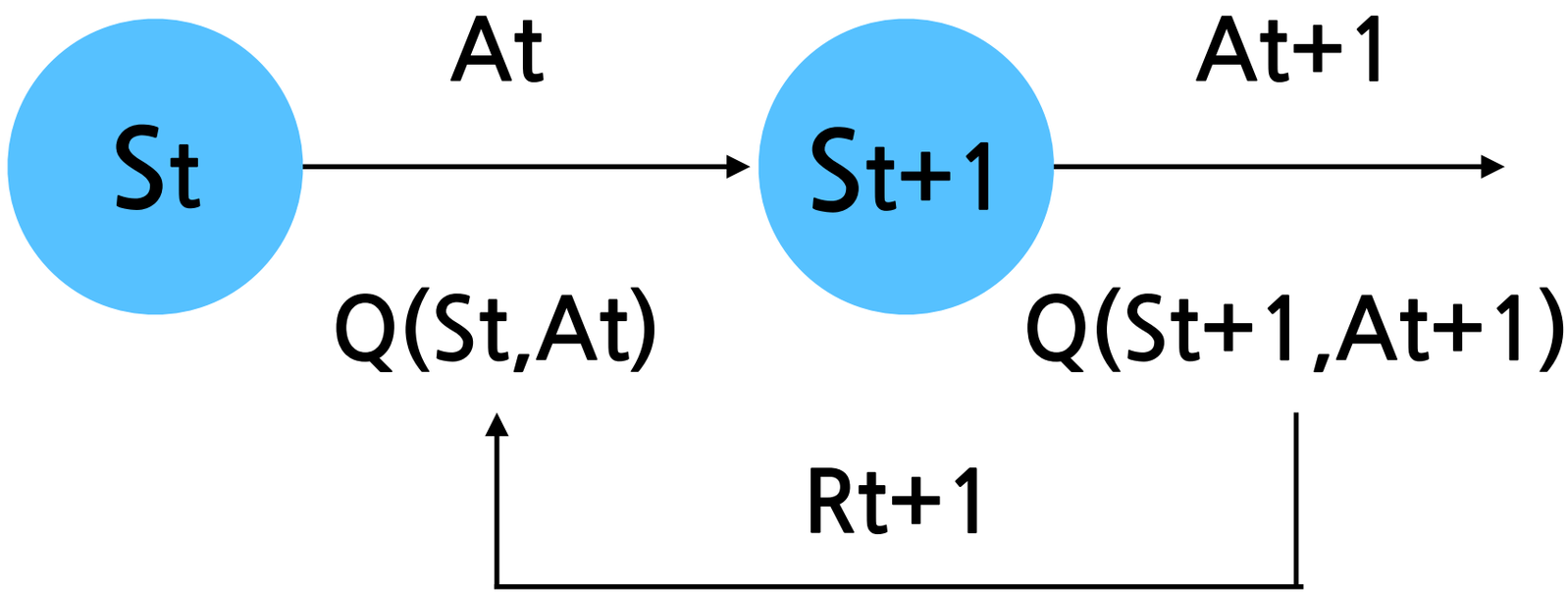
Model-free

TD Temporal-Difference

기댓값을 계산 X

샘플링을 통해 현재의 **Q함수** 업데이트

$$Q(St,At) \leftarrow Q(St,At) + a(R + rQ(St+1,At+1) - Q(St,At))$$


St
At
St+1
At+1
Q(St,At)
Q(St+1,At+1)
Rt+1

한편, 탐욕 정책은 -

e-greedy

따라서,

1. **e-greedy**통해 샘플획득

2. 획득한 샘플로 다음 식을 통해 **Q(St,At) 업데이트**

$$Q(St,At) \leftarrow Q(St,At) + a(R + rQ(St+1,At+1) - Q(St,At))$$

따라서,

1. **e-greedy**통해 샘플획득

2. 획득한 샘플로 다음 식을 통해 **Q(St,At) 업데이트**

$$Q(St,At) \leftarrow Q(St,At) + a(R + rQ(St+1,At+1) - Q(St,At))$$

따라서,

1. **e-greedy**통해 샘플획득

2. 획득한 샘플로 다음 식을 통해 **Q(St,At) 업데이트**

$$Q(St,At) \leftarrow Q(St,At) + a(R + rQ(St+1,At+1) - Q(St,At))$$

SARSA

SARSA의 한계

자신이 **행동한대로 학습하는** TD

그럼 탐험 다 못하나요!

Off policy

행동하는 정책과
학습하는 정책 분리

그렇군요!

Off policy

행동하는 정책과
학습하는 정책 분리

SARSA와 같다
Q함수

Off policy

행동하는 정책과
학습하는 정책 분리

SARSA와 같다
Q함수

$$q(s,a) \leftarrow q(s,a) + \alpha\left(r + \gamma \max_{a'} q(s',a') - q(s,a)\right)$$

Off policy

행동하는 정책과
학습하는 정책 분리

SARSA와 같다
Q함수

$$q(s,a) \leftarrow q(s,a) + \alpha\left(r + \gamma \max_{a'} q(s',a') - q(s,a)\right)$$

$$q(s,a) \leftarrow q(s,a) + \mathbb{E}\left(r + \gamma \max_{a'} q(s',a')\right)$$

왜 이런 차이?

Q러닝에서 학습했던 **다음 상태의 행동**

실제로 다음 상태에서의 행동 다르다

벨만 기대 방정식 → 정책 이터레이션 → 살사

벨만 최적 방정식 → 가치 이터레이션 → 큐러닝

DP 한계

계산 복잡도

차원의 저주

환경을 몰라

DP 한계

계산 복잡도

차원의 저주

~~환경을 몰라~~

DP 한계

~~계산 복잡도~~

~~차원의 저주~~

Q함수를 매개변수로 **근사**

~~환경을 몰라~~

DP 한계

~~계산 복잡도~~

~~차원의 저주~~

→ Q함수를 근사함수로 근사

~~환경을 몰라~~

DP 한계

~~계산 복잡도~~

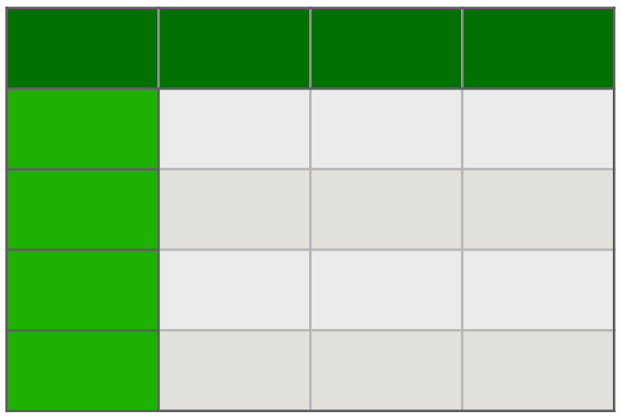
~~차원의 저주~~

Q함수를 **인공신경망**으로 근사

~~환경을 몰라~~

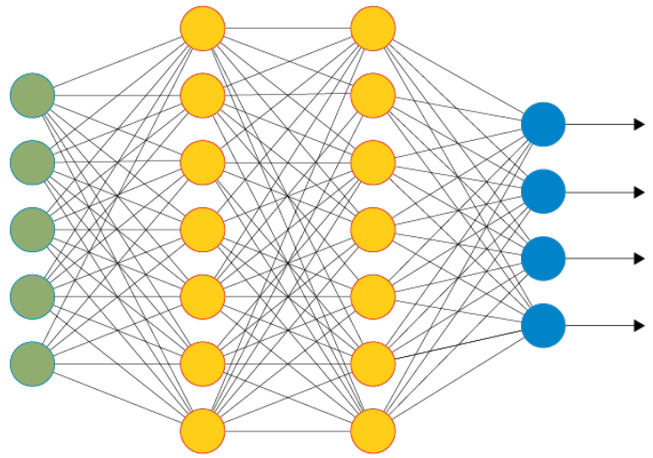
SARSA

Table



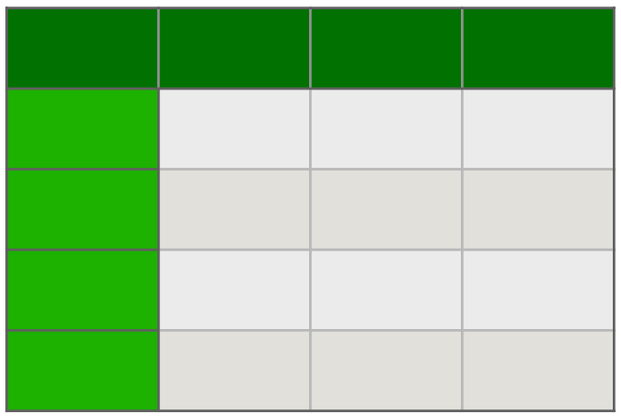
Deep SARSA

인공신경망



SARSA

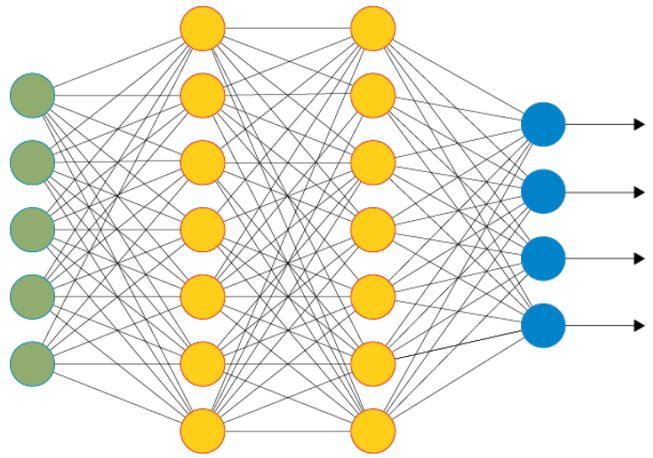
Table



Q 함수

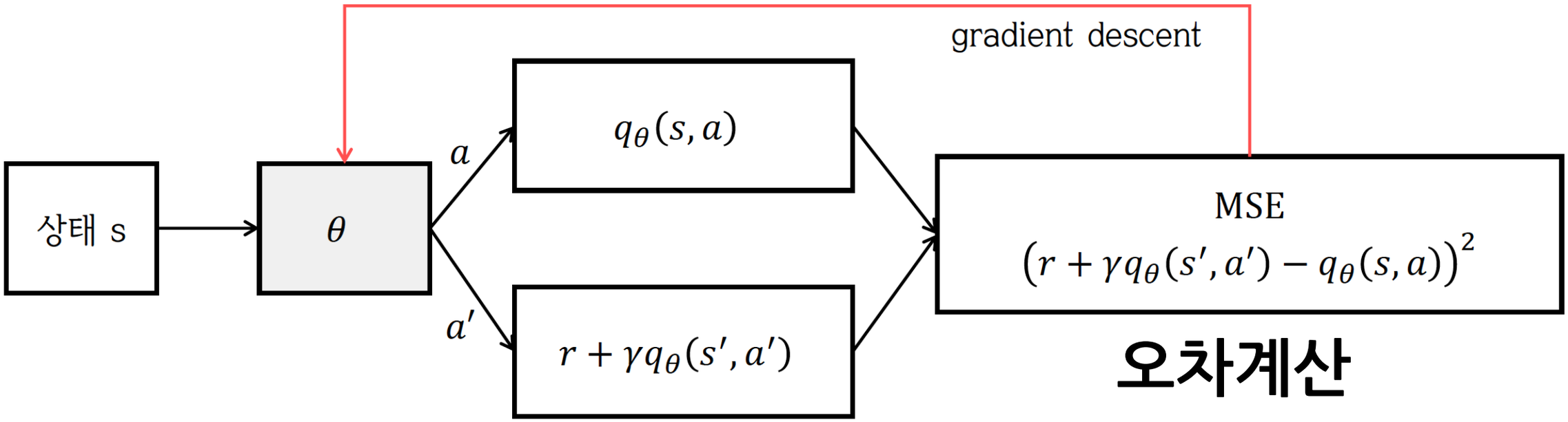
Deep SARSA

인공신경망



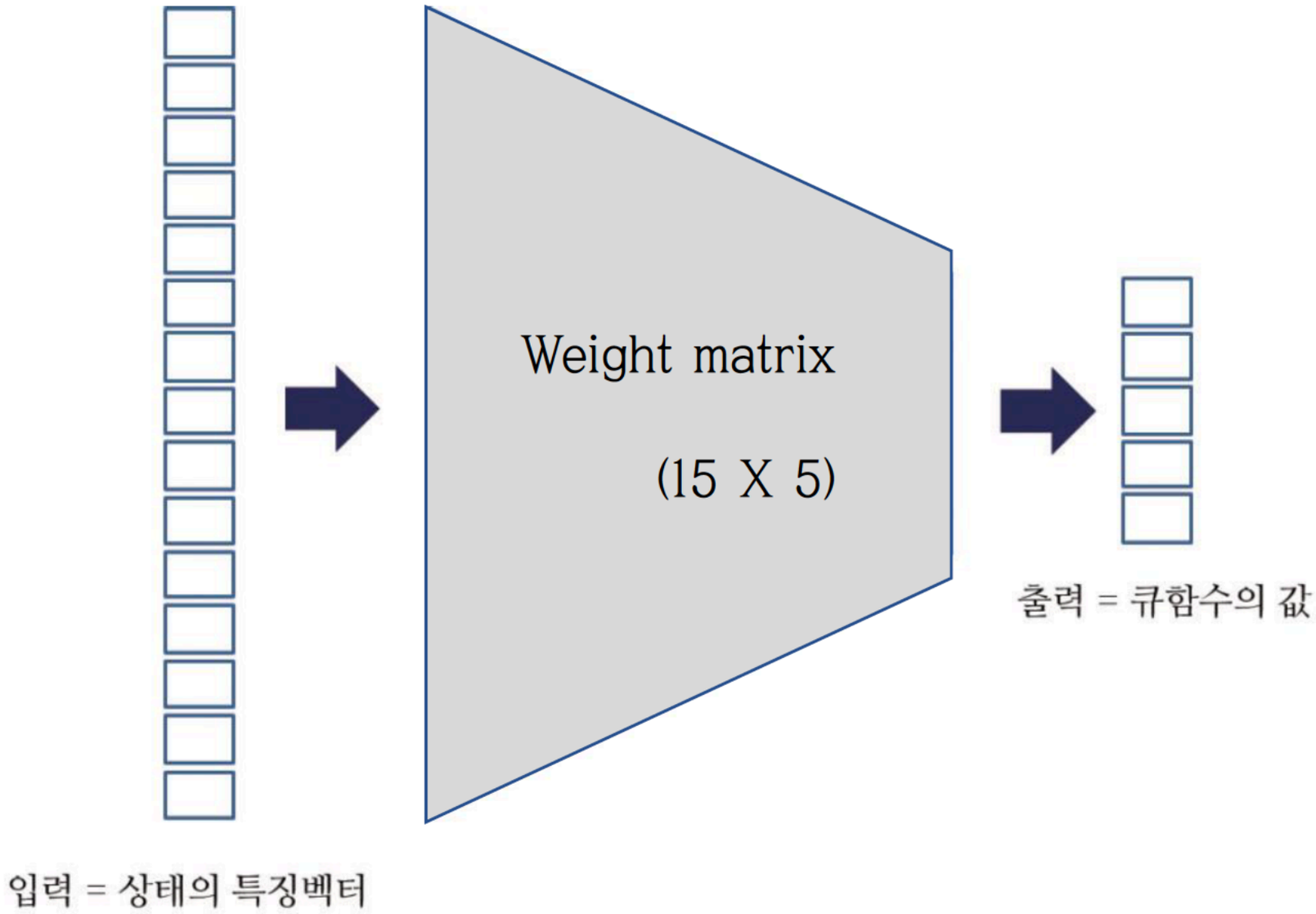
MSE

예측역할

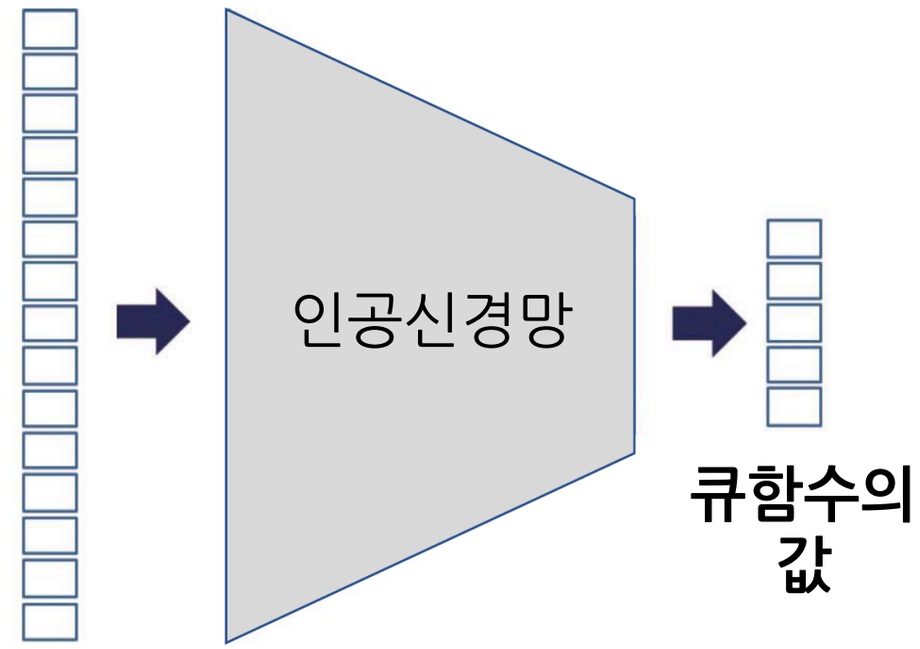


정답역할

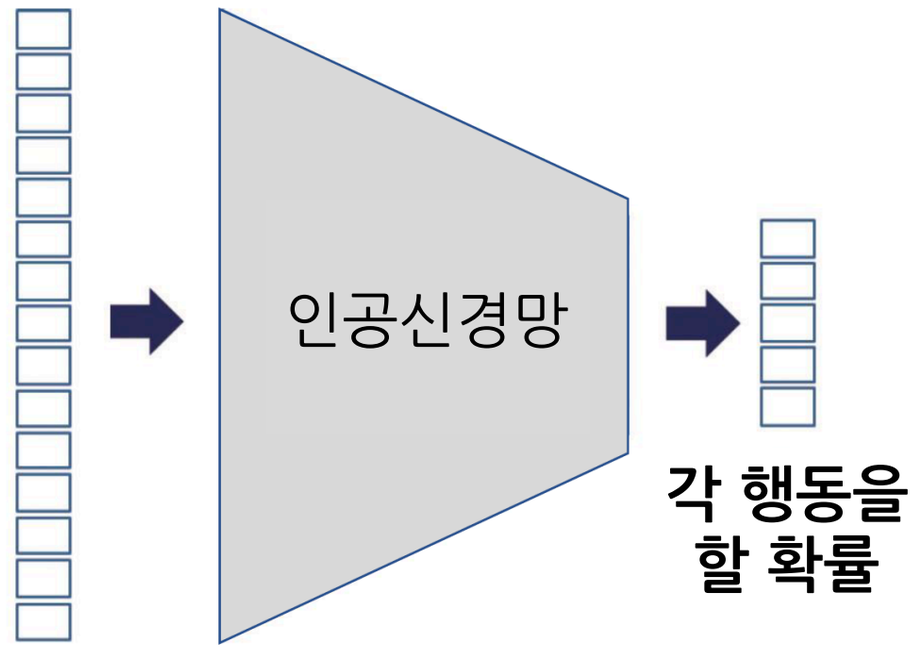
오차계산


Weight matrix
(15 X 5)
입력 = 상태의 특징벡터
출력 = 큐함수의 값

가치(Q)를 근사



정책을 근사

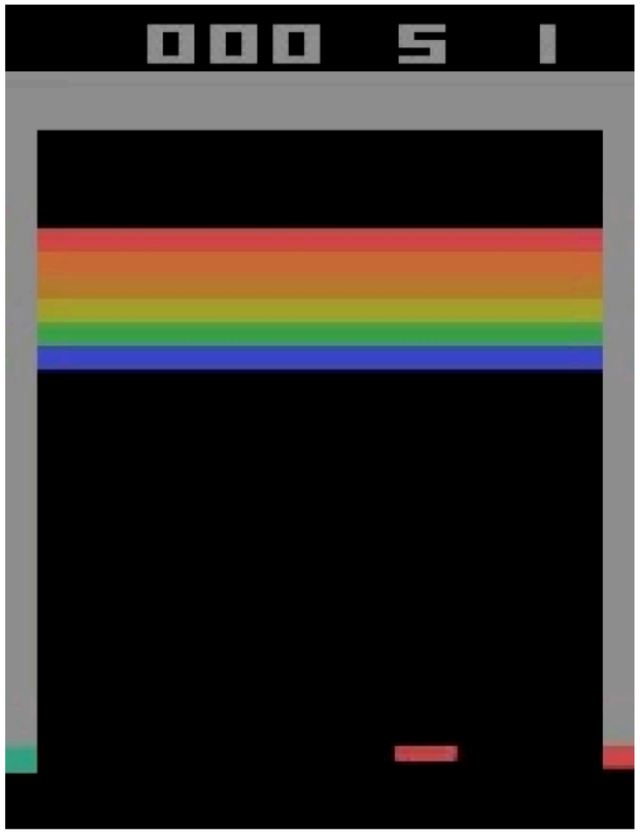


DQN

1. Deep RL = Deep learning + RL

2. **DQN (2013)**

3. 화면은 high D -> CNN 사용



DQN 특징

1. CNN

2. Experience replay

3. Online learning with Stochastic gradient descent

4. Target Q-network

DQN 특징

1. CNN　화면으로부터 바로 학습가능

2. Experience replay　　샘플들의 상관관계를 깸

3. Online learning with Stochastic gradient descent

4. Target Q-network

DQN 특징

3. Online learning with Stochastic gradient descent

매 스텝마다 replay 메모리에서 추출한 미니배치로 Q업데이트

$$q(s,a) = q(s,a) + \alpha\left(r + \gamma \max_{a'} q(s',a') - q(s,a)\right)$$

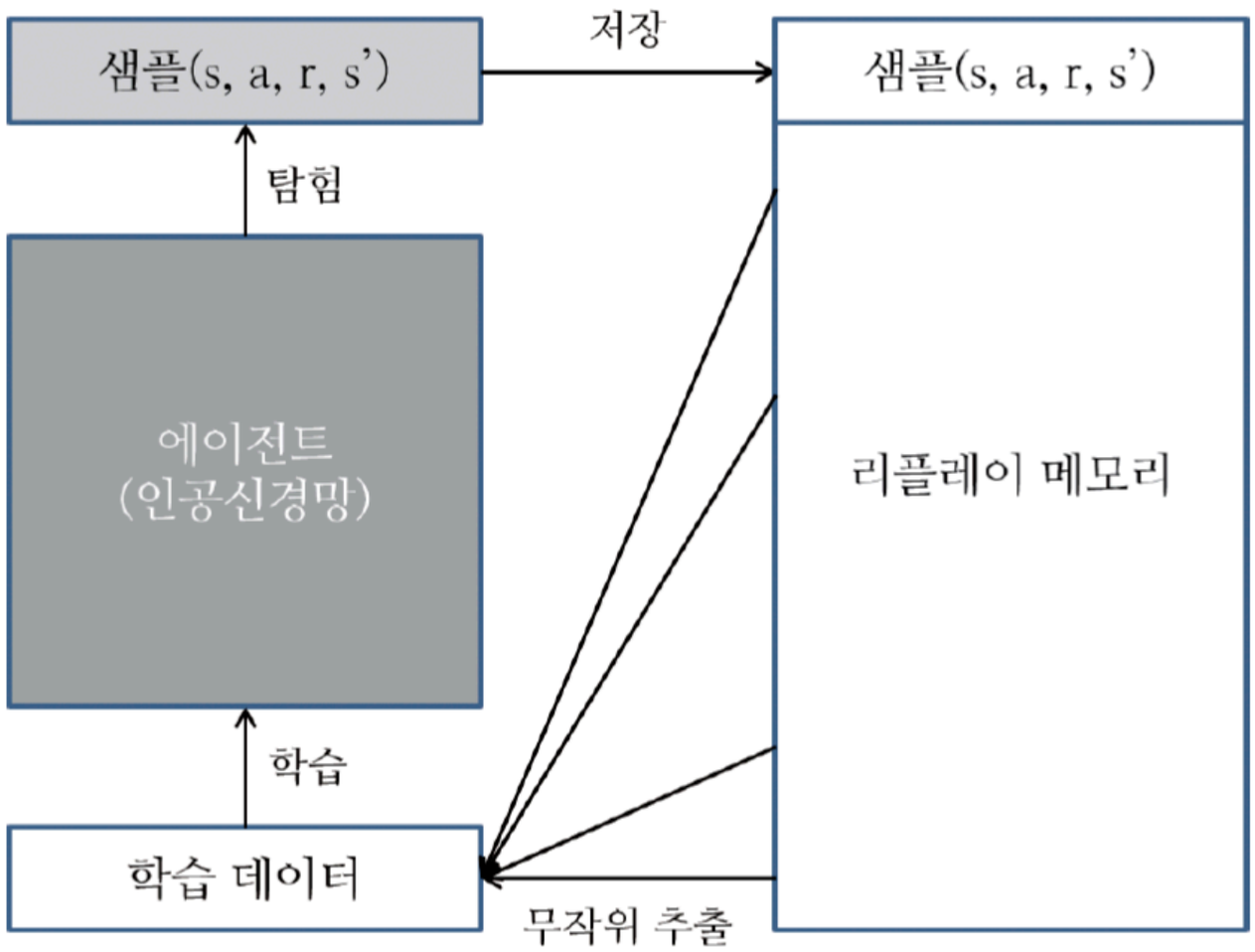
$$MSE\ error: \left(r + \gamma \max_{a'} q_{\theta^-}(s',a') - q_\theta(s,a)\right)^2$$

4. Target Q-network

update의 네트워크를 분리

일정 주기마다 현재 네트워크를 업데이트

DQN 도식



DQN 학습과정

1. 상태에 따른 행동선택

2. 선택한 행동으로 환경에서 1 time step 진행

3. 환경으로부터 다음 상태(S') 보상(R') 받음

4. 샘플([s, a, r, s']을 replay memory에 저장

5. memory에서 random sampling -> mini-batch update

6. 일정 주기마다 Target network update

DQN 세부사항

1. 이미지 preprocessing

2. 4 images in 1 history

3. 30 no-op

4. Clip

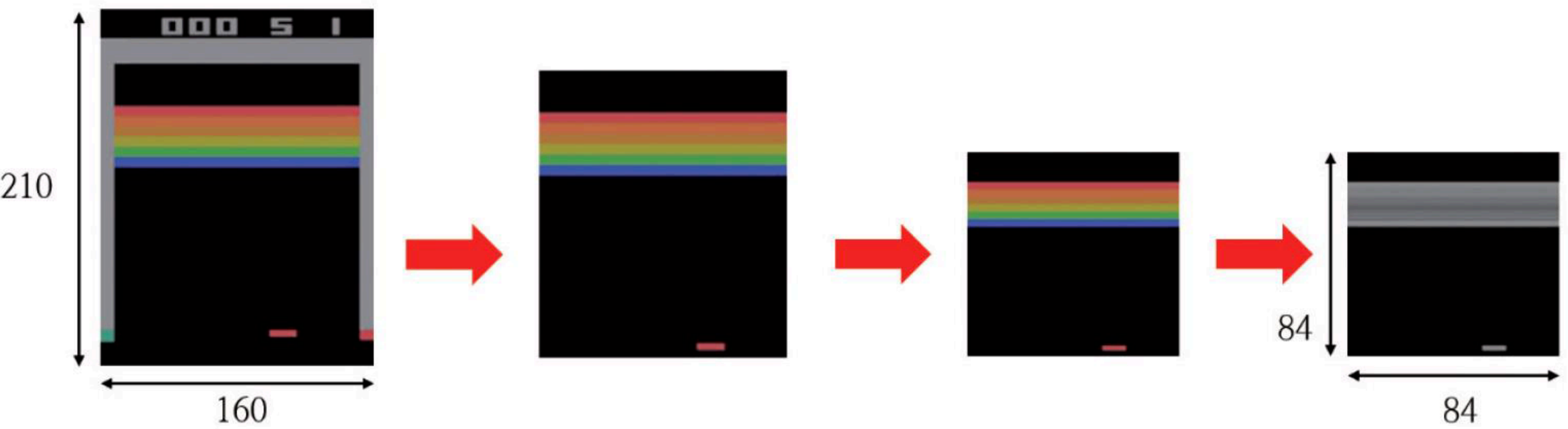
5. Huber loss

이미지 preprocessing

1. 이미지 preprocessing

Gray-scale : (210, 160, 3) -> (210, 160, 1)

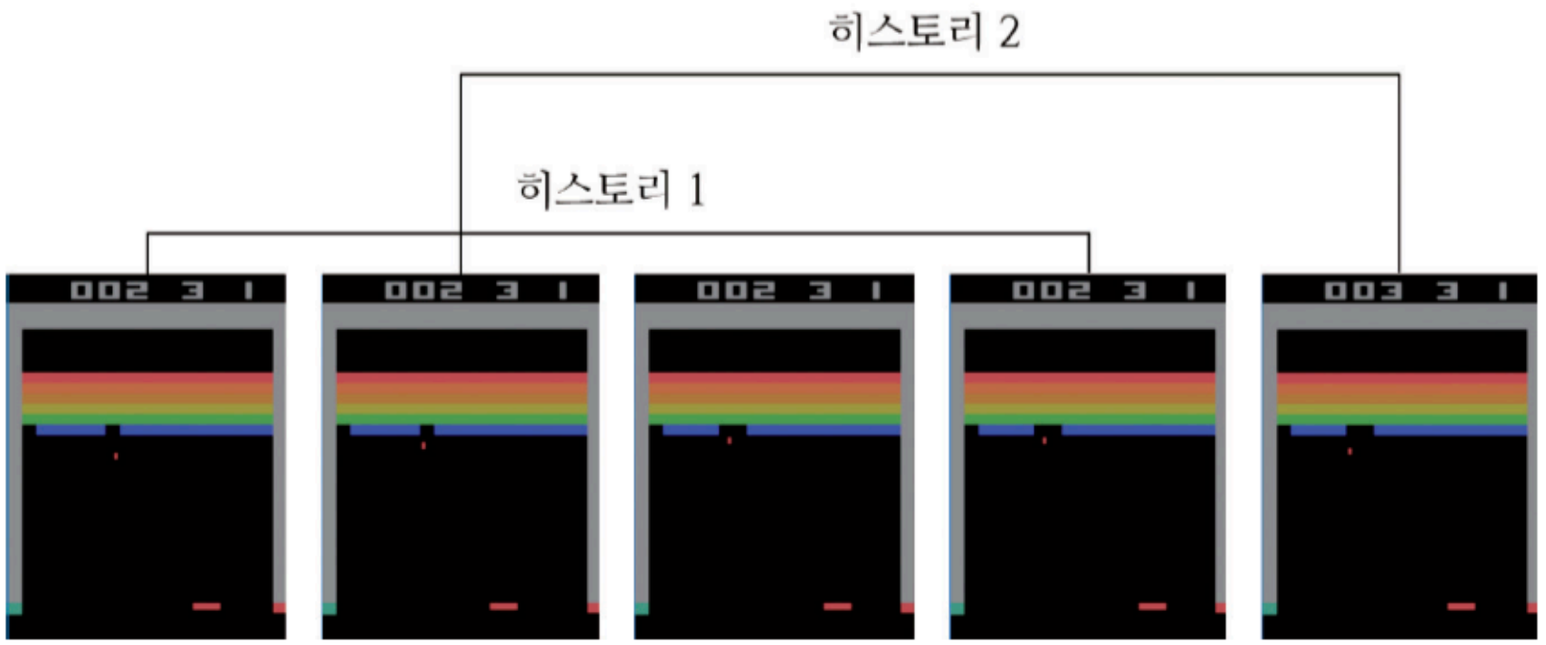
Resize : (210, 160, 1) -> (84, 84, 1)



4 images in 1 history

2. 4 images in 1 history

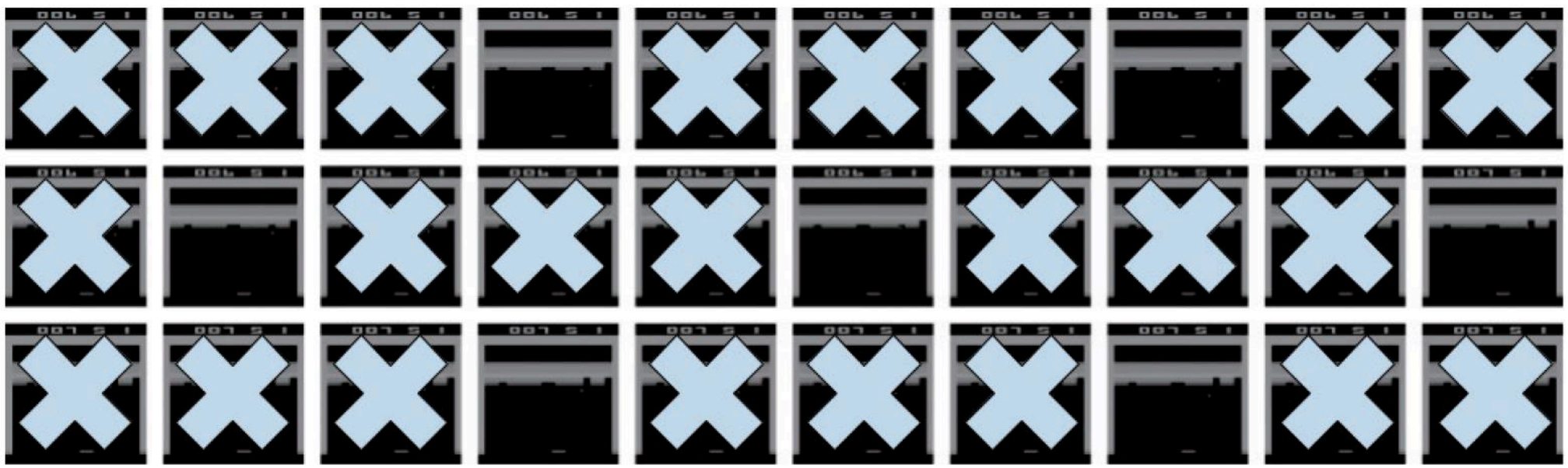
속도 정보를 포함하기 위해 연속된 4개 이미지를
하나의 history로 네트워크에 input



4 images in 1 history

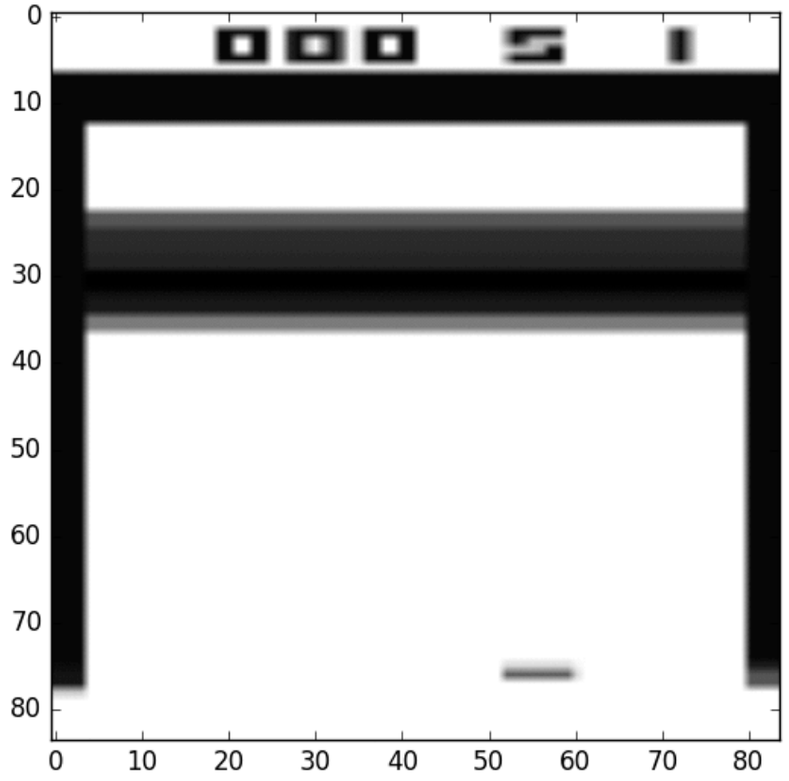
Atari game은 연속된 4개 이미지에서 큰 변화 X

학습에 4번의 이미지 중에 1개만 사용 (frame skip)



4 images in 1 history

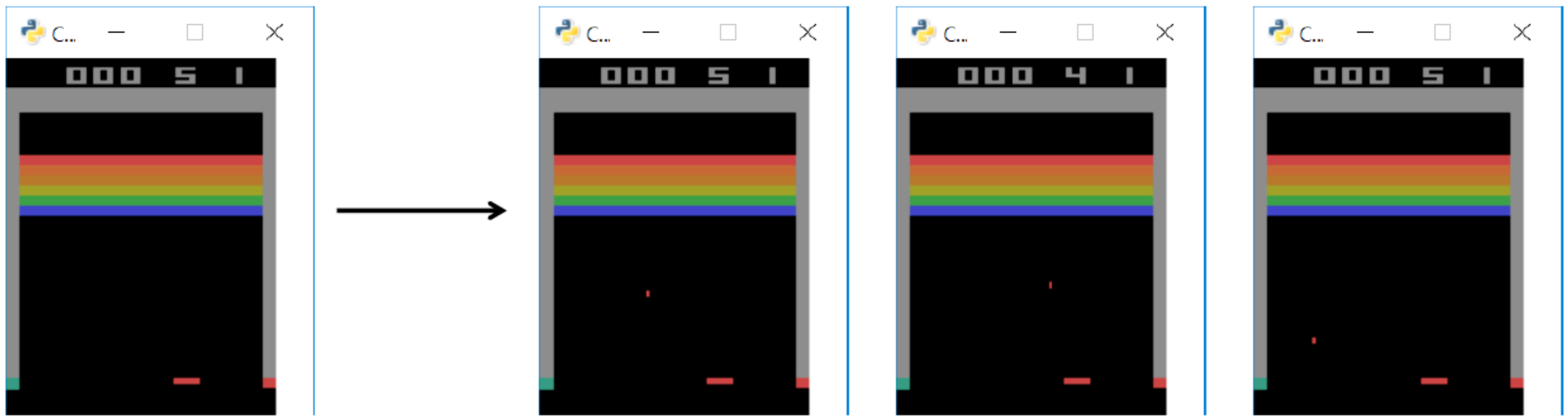
결과적으로,
Q-networt 에서 바라보는 input



30 no-op

시작 이미지가 항상 같다 -> 초반 local 에 수렴확률 높다

0-30 time-step중 랜덤으로 선택한 뒤 그동안 무행동

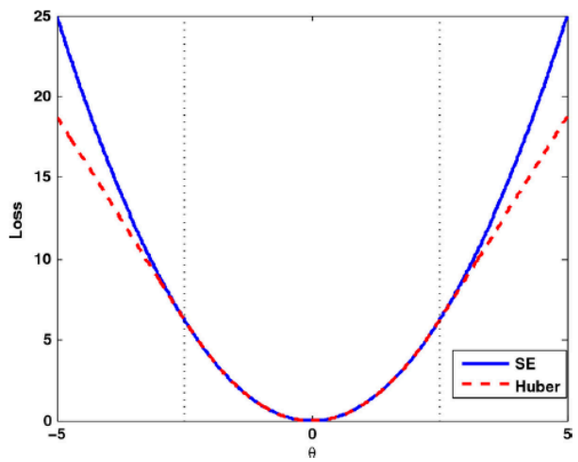


Reward clip

게임마다 다른 reward크기 -> -1<reward<1

Huber loss function

Loss 값의 variance -> 학습 불안정성에 영향



$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

```
quadratic_part = K.clip(error, 0.0, 1.0)
linear_part = error - quadratic_part
loss = K.mean(0.5 * K.square(quadratic_part) + linear_part)
```

-1<<1에서는 quadratic, 다른곳은 linear

갓mind 하이퍼파라미터

변수	값
미니 배치 크기	32
리플레이 메모리 크기	400000
히스토리 길이	4 프레임
타깃 모델 업데이트 주기	10000 스텝에 한 번
감가율	0.99
프레임 스킵	4개 화면 중 1개 사용
학습 속도(경사하강법)	0.00025
ε 관련	1부터 0.1까지 1000000 스텝 동안 감소
학습 시작	50000 스텝 후

알고리즘

1. 환경 reset, 30 no-op

2. History에 따라 행동 선택

3. 선택한 행동들로 1 time-step 진행

4. 샘플 형성 및 버퍼에 넣기

5. 50000 스텝 이상일 경우 mini-batch 추출

6. 10000 스텝마다 타겟N 업데이트

Q형님과 만나봐요!