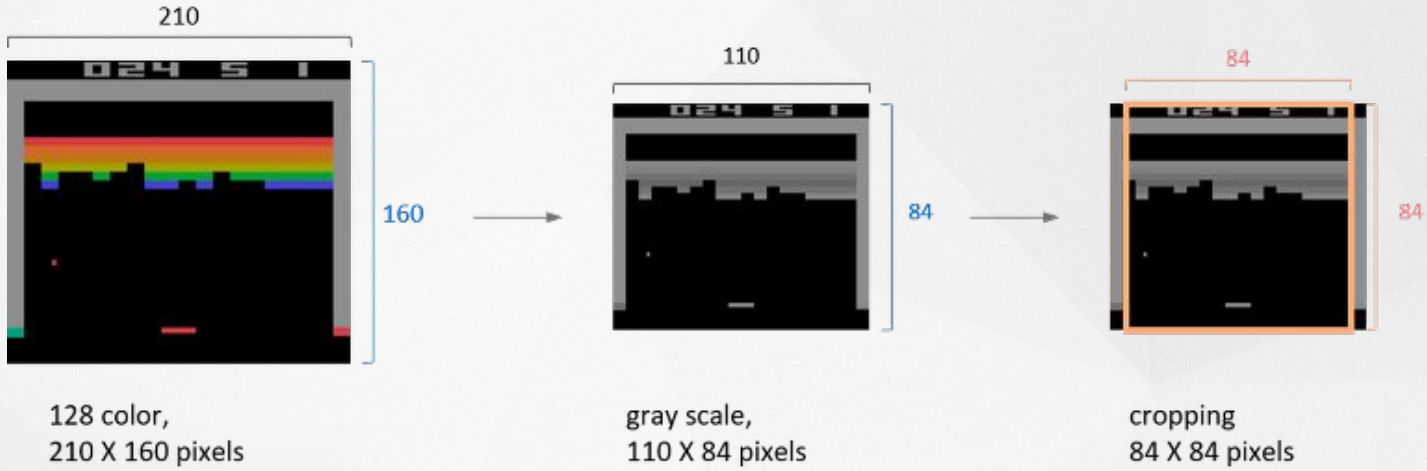
Chapter 06. 스스로 전략을 짜는 강화학습 (Reinforcement Learning)

DQN



Function Approximation

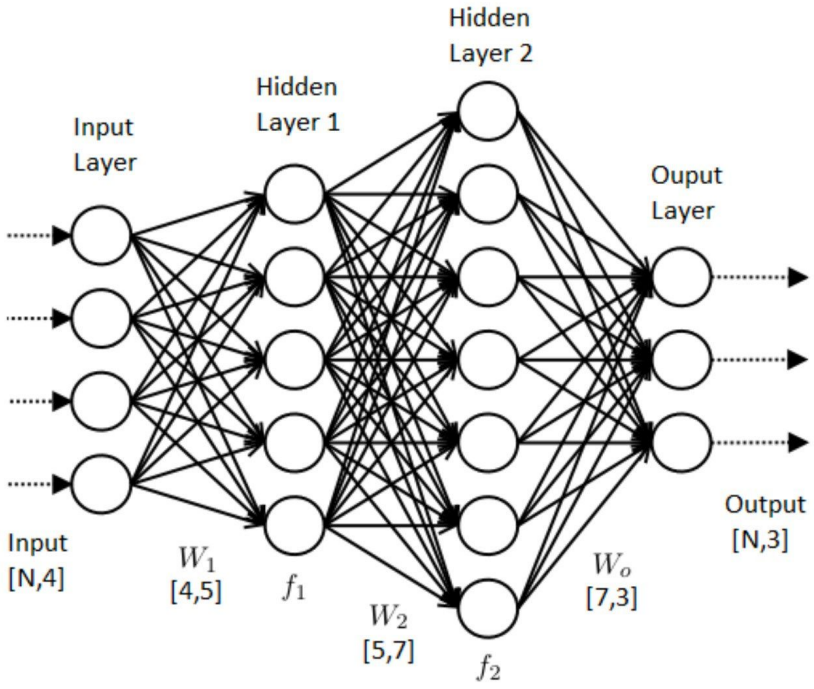
https://sumniya.tistory.com/17?category=781573

ax^3+bx^2+cx+d

parameter(a,b,c,d)

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$

$$\hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$



Function Approximation

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

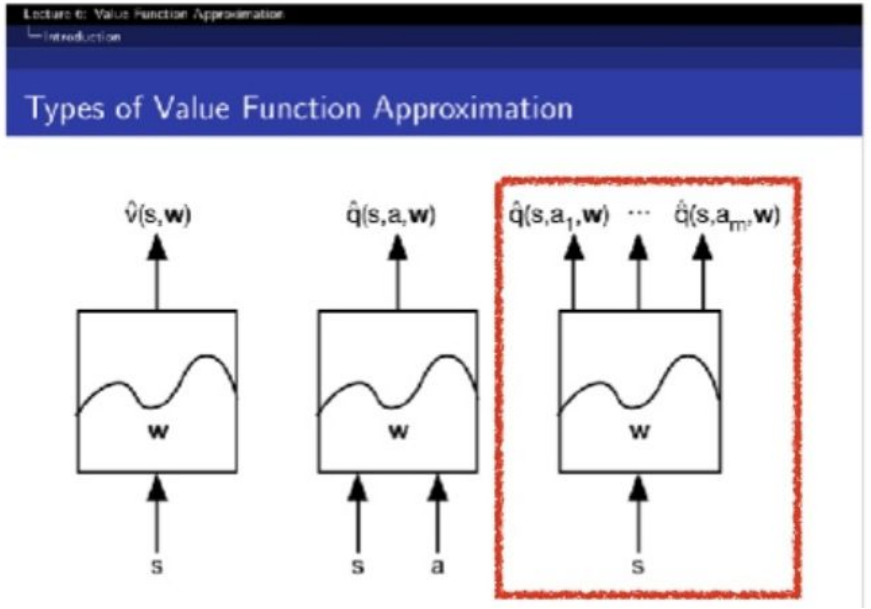
- Reinforcement learning agents가 실제 세계의 복잡도를 가진 문제에서 잘 작동하기 위해서는:
  - 상당한 고차원의 sensory inputs으로부터 representation을 잘 얻어낼 수 있어야 한다.
  - 얻어낸 representation으로 과거의 경험을 일반화하여 새로운 상황에서도 잘 적용할 수 있어야한다.

➡ RL의 유용성은 아주 제한적인 도메인(e.g. 저차원의 state-space를 가진 도메인)에 머물러 있다.

Function Approximation

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

- Deep Convolutional Neural Network가 non-linear function approximator로써 이례적인 성능을 보이고 있다.
- CNN 구조를 이용하여 raw sensory data를 입력으로하는 action-value function의 근사함수를 만들어보면 어떨까?



Function Approximation

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

강화학습에서 action-value(Q) function을 나타내기 위해 non-linear function approximator를 사용하였을 경우 수렴이 보장되지 않는 것으로 알려져 있다.

Lecture 6: Value Function Approximation
└Incremental Methods
└Convergence

Convergence of Prediction Algorithms

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	TD(0)	✓	✓	✗
	TD(λ)	✓	✓	✗
Off-Policy	MC	✓	✓	✓
	TD(0)	✓	✗	✗
	TD(λ)	✓	✗	✗

Problem of RL

- **Issue 1.** 성공적인 Deep Learning applications는 hand-labelled training data set을 요하는데, Reinforcement Learning에서는 오로지reward를 통해 학습이 이루어지고, 그 reward도 sparse하고 noisy 심지어는 delay되어 주어진다.

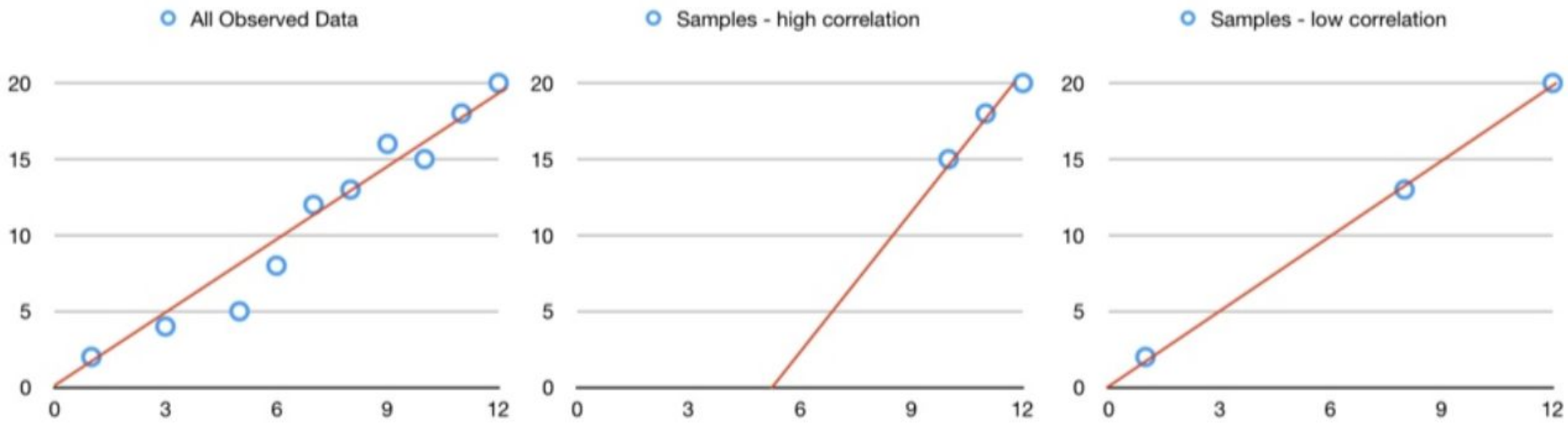
- **Issue 2.** Deep Learning에서는 data sample이 i.i.d. 분포를 가정하지만, Reinforcement Learning에서는 현재 state가 어디인지에 따라 갈 수 있는 다음 state가 결정되기때문에 state간의 correlation이 크다. 즉, data간의 correlation이 크다.

Problem of RL

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

- Correlation between samples

강화학습에서의 학습데이터는 시간의 흐름에 따라 순차적으로 수집되고,
이 순차적인 데이터는 근접한 것들끼리 높은 correlation을 띄게된다.



만약에 이 순차적인 데이터를 그대로 입력으로 활용하게 되면 입력이미지
들 간의 높은 correlation에 의해 학습이 불안정해질 것이다.

Problem of RL

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

- **Non-stationary targets**

MSE(Mean Squared Error)를 이용하여 optimal action-value function을 근사하기 위한 loss function을 다음과 같이 표현할 수 있다.

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma max_{a'} Q(s', a'; \theta_i) - Q(s, a; \theta_i)\right)^2\right],$$

where θ_i are the parameters of the Q-network at iteration i.

이는 Q-learning target 를 근사하는 $y_i = r + \gamma max_{a'} Q(s', a'; \theta_i)$ 를 구하려는 것과 같다. 문제는 $Q(s, a; \theta_i)$ 가 Q함수에 대해 의존성을 갖고 있으므로 Q함수를 업데이트하게 되면 target y_i 또한 움직이게 된다는 것이다. 이 현상으로 인한 학습의 불안정해진다.

DQN

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

- Correlation between samples ← experience replay (replay memory)
- Non-stationary targets ← fixed Q-targets

1. raw pixel을 받아와 directly input data로 다룬 것
2. CNN을 function approximator로 이용한 것
3. 하나의 agent가 여러 종류의 Atari game을 학습할 수 있는 능력을 갖춘 것
4. Experience replay를 사용하여 data efficiency를 향상한 것

DQN

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

Replay Memory

1. Agent의 경험(experience) $e_t = (s_t, a_t, r_t, s_{t+1})$를 time-step 단위로 data set $D_t = \{e_1, \ldots, e_t\}$에 저장해 둔다.

2. 저장된 data set으로부터 uniform random sampling을 통해 minibatch를 구성하여 학습을 진행한다. $\left((s, a, r, s') \sim U(D)\right)$

   - Minibatch가 순차적인 데이터로 구성되지 않으므로 입력 데이터 사이의 correlation을 상당히 줄일 수 있다.

   - 과거의 경험에 대해 반복적인 학습을 가능하게 한다[6].

   - 논문의 실험에서는 replay memory size를 1,000,000으로 설정한다.

DQN

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

Fixed Q-Target

$Q(s, a; \theta)$와 같은 네트워크 구조이지만 다른 파라미터를 가진(독립적인) target network $\hat{Q}(s, a; \theta^-)$를 만들고 이를 Q-learning target y_i에 이용한다.

$$y_i = r + \gamma max_{a'} \hat{Q}(s', a'; \theta_i^-) .$$

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[\left(r + \gamma max_{a'} \hat{Q}(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right],$$

in which γ is the discount factor determining the agent's horizon, θ_i are the parameters of the Q-network at iteration i and θ_i^- are the network parameters used to compute the target at iteration i.

- Target network parameters θ_i^-는 매 C step마다 Q-network parameters(θ_i)로 업데이트된다. 즉, C번의 iteration동안에는 Q-learning update시 target이 움직이는 현상을 방지할 수 있다.

- 논문의 실험에서는 C값을 10,000으로 설정한다.

DQN

Gradient Clipping

- Loss function:

$$\left(r + \gamma max_{a'} Q(s', a'; \theta_i^- - Q(s, a; \theta_i)\right)^2$$

- 위 loss function에 대한 gradient의 절대값이 1보다 클때는 절대값이 1이 되도록 clipping해준다[5].

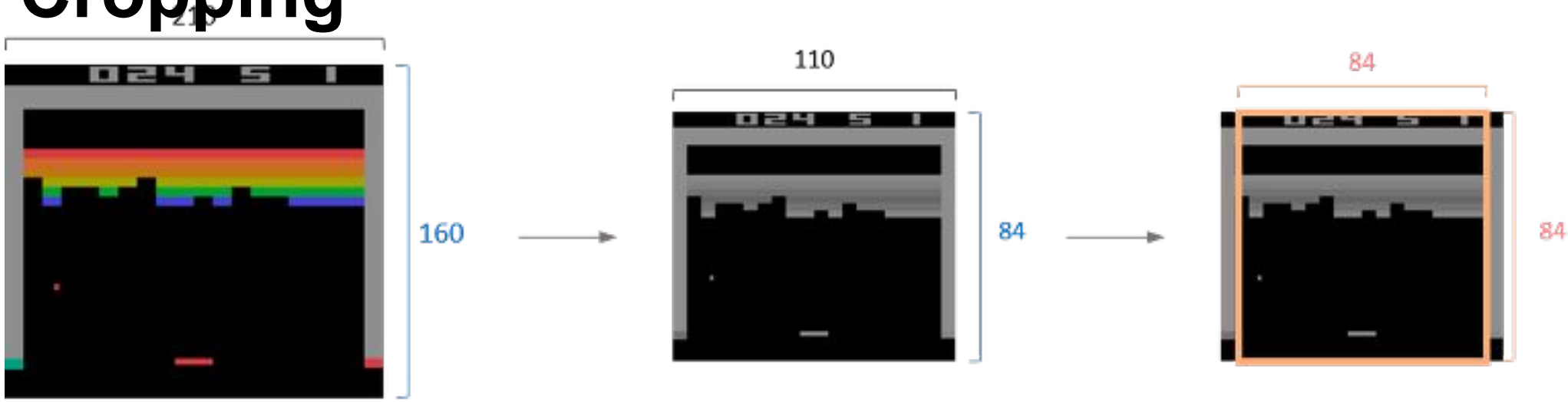
- Huber loss[10]와 기능적으로 동일하기 때문에 구현시에는 loss function을 Huber loss로 정의하기도 한다[11].

$$\phi_{\text{hub}}(u) = \begin{cases} u^2 & |u| \leq M \\ M(2|u| - M) & |u| > M \end{cases}$$

DQN

https://sumniya.tistory.com/18?category=781573

Scale Down, Gray Scale, Cropping



128 color,
210 X 160 pixels

gray scale,
110 X 84 pixels

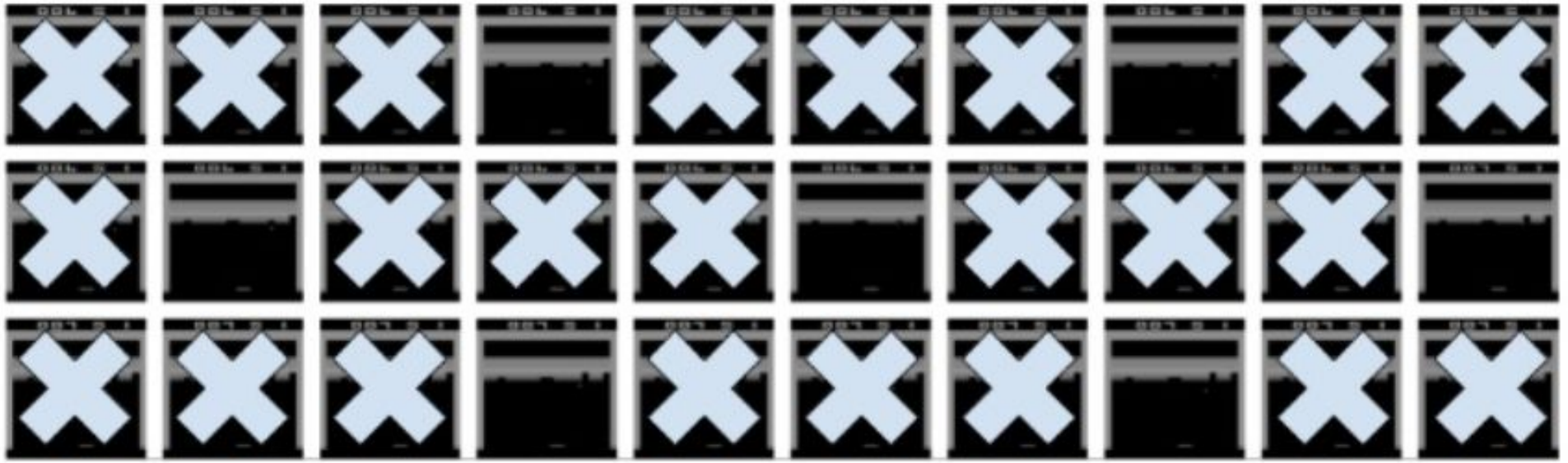
cropping
84 X 84 pixels

DQN

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

Skipped Frame

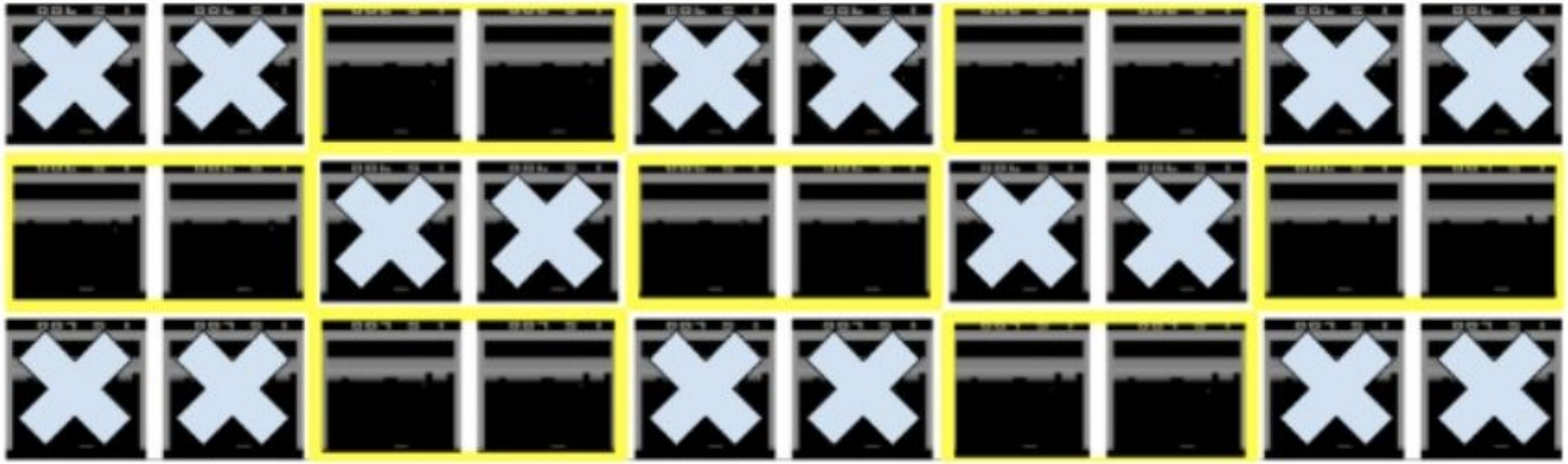
연속된 이미지중 k번째 이미지만 선택



*모든 frame을 전부 입력으로 활용하는 것은 입력 데이터 간의 correlation을 높이게 된다.

DQN

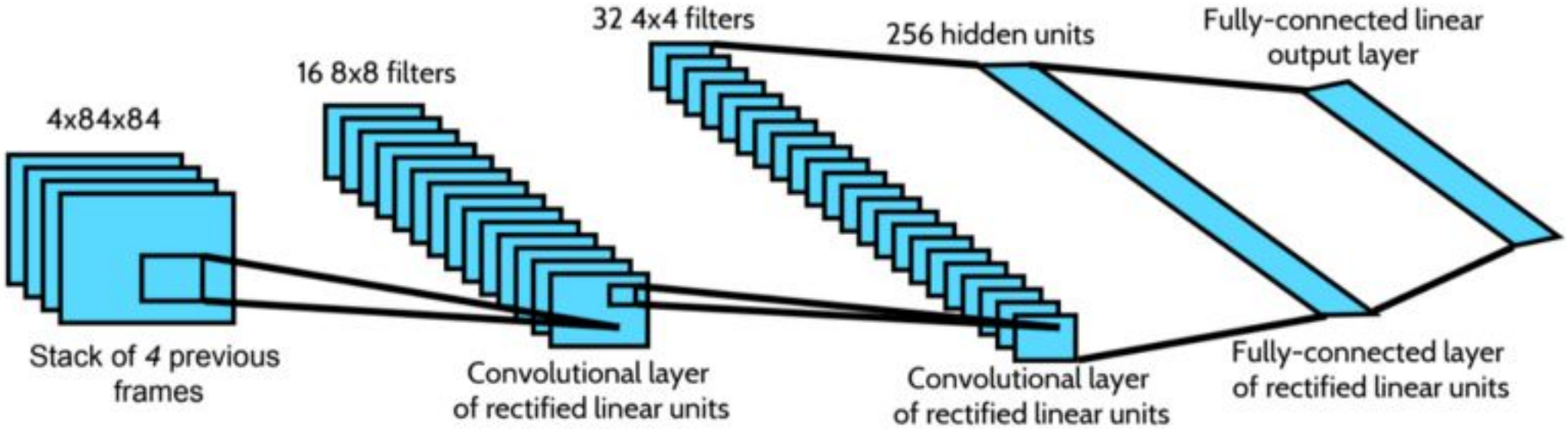
Odd, Even Frame, Pixel-wise maximum



*Atari 2600은 화면에 한 번에 표시할 수 있는 sprites가 단 5개 뿐이어서 짝수 프레임, 홀수 프레임에 번갈아서 표시하는 것으로 여러개의 sprites를 화면에 보여줄 수 있었다. 연속된 두 이미지에 대해 component-wise maximum을 취해줌으로써 이를 한 이미지에 모두 표시할 수 있다.

DQN

CNN



DQN

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

- *Thank you*