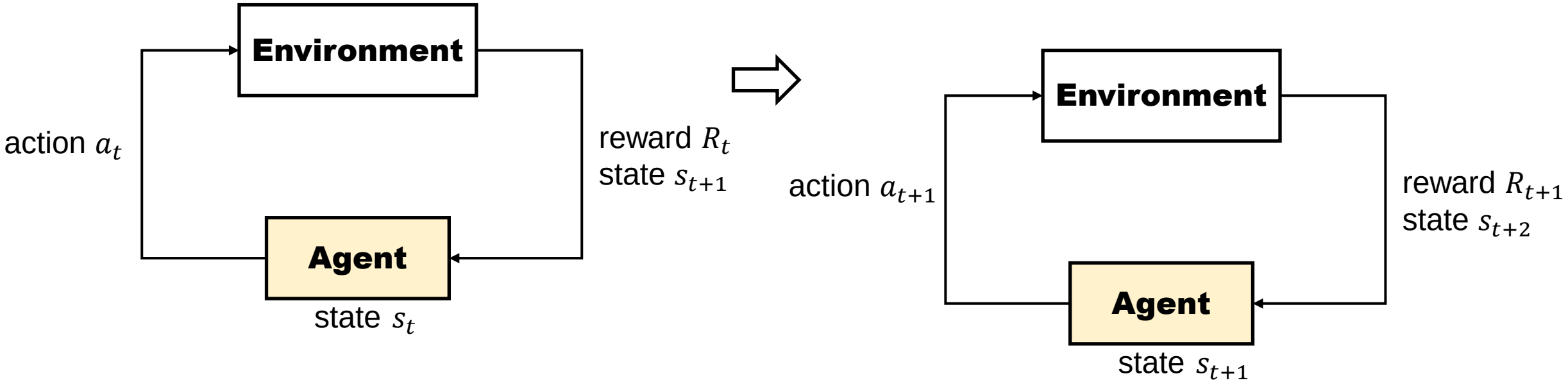
Deep Q-Network

# Playing Atari with Deep Reinforcement Learning

Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).
참고 유튜브: https://www.youtube.com/watch?v=cvctS4xWSaU

# Reinforcement Learning




- **Reinforcement Learning (강화 학습)**

강화학습은 현재의 상태(State)에서 어떤 행동(Action)을 취하는 것이 최적인지를 학습하는 것이다. 행동을 취할 때마다 외부 환경에서 보상(Reward)이 주어지는데, 이러한 보상을 최대화 하는 방향으로 학습이 진행된다.

$s_t$: 현재화면 이미지
$a_t$: 왼쪽으로가기, 오른쪽으로가기, 가만히있기
$R_t$: 게임스코어의 변화

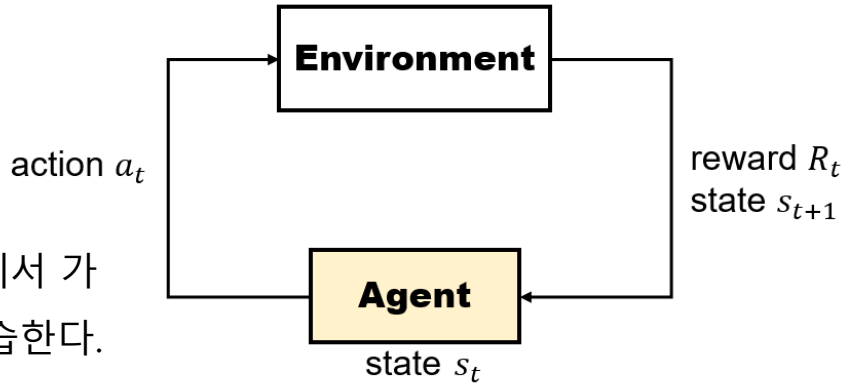
# Q-Learning

- **강화학습의 목표**

현재 time $t$ 에서 total future reward $G_t$(=return)을 최대화하는 것

$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots$

$\gamma \in [0,1]$는 discount factor




- **Q-Learning**

강화 학습 알고리즘 중의 하나. 현재 상태 $s_t$ 에서 특정 행동 $a_t$ 를 취하는 것이 전체 게임에서 가져다 줄 효용(total future reward)의 기대값(Q-value)을 예측하는 함수인 Q function을 학습한다.

Q-value = $\mathbb{E}[G_t] = Q(s_t, a_t)$

- **Policy**

policy는 $s_t$에서 가능한 action중 하나를 선택할 확률에 대한 분포 $\mathrm{P}(a_t|s_t)$ 를 말함.

- **Q-Learning의 목표**

$(s_t, a_t)$ pair에 대한 Q-value를 추정함.

그리고 현재 state $s_t$에서의 optimal policy를 찾음.

optimal policy는 현재 $s_t$에서 $Q(s_t, a_t)$를 max로 만드는 $a_t$를 항상 선택하는 policy $\pi(a_t|s_t)$ 를 말함.

학습이 완료되면, 현재 state $s_t$에서 $Q(s_t, a_t)$를 max 로 하는 $a_t$를 선택함으로써 강화학습의 목표를 달성할 수 있음.

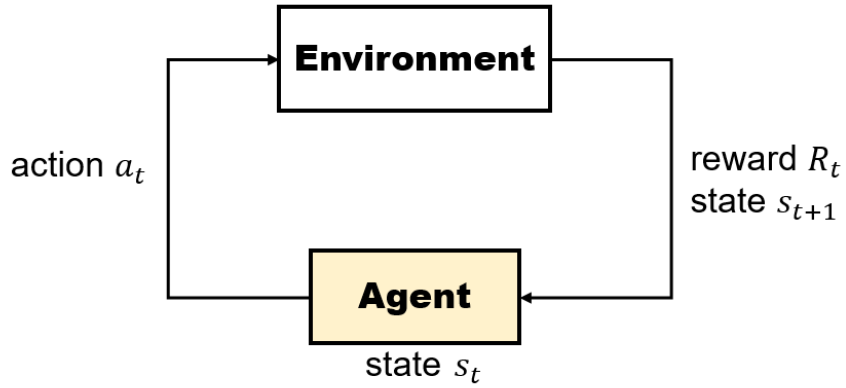
# Q-Learning

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots; \quad \mathbb{E}[f(x)] = \int_x f(x)\mathrm{P}(x)dx$$

- **Q-value의 수학적 표현**

$$Q(s_t, a_t) = \mathbb{E}[G_t] = \int_{s_{t+1}:a_\infty} G_t \, \mathrm{P}(s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, \cdots \mid s_t, a_t) \, ds_{t+1}:a_\infty$$

→ 계산 불가능!




- **Q-value 식을 약간 변경**

$$= \int_{s_{t+1}:a_\infty} G_t \, \mathrm{P}(s_{t+2}, a_{t+2}, \cdots \mid \cancel{s_t}, \cancel{a_t}, s_{t+1}, a_{t+1}) \, \mathrm{P}(s_{t+1}, a_{t+1} \mid s_t, a_t) \, ds_{t+1}:a_\infty$$

(그리고 $G_t = R_t + \gamma G_{t+1}$이므로)

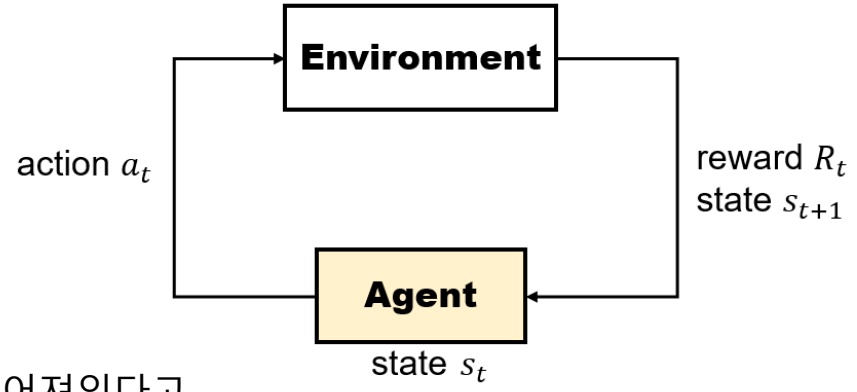
$$= \int_{s_{t+1}, a_{t+1}} \{ \int_{s_{t+2}:a_\infty} (R_t + \gamma G_{t+1}) \, \mathrm{P}(s_{t+2}, a_{t+2}, \cdots \mid s_{t+1}, a_{t+1}) \, ds_{t+2}:a_\infty \} \, \mathrm{P}(s_{t+1}, a_{t+1} \mid s_t, a_t) \, ds_{t+1}, a_{t+1}$$

$$= Q(s_{t+1}, a_{t+1})$$

$$= \int_{s_{t+1}, a_{t+1}} (R_t + \gamma \, Q(s_{t+1}, a_{t+1})) \, \mathrm{P}(s_{t+1}, a_{t+1} \mid s_t, a_t) \, ds_{t+1}, a_{t+1}$$

→ 식이 조금 간단해 졌지만, 여전히 계산할 수 없다.

# Q-Learning




- **Q-value를 계산하려면?**

$$Q(s_t, a_t) = \int_{s_{t+1}, a_{t+1}} (R_t + \gamma\, Q(s_{t+1}, a_{t+1}))\, \mathrm{P}(s_{t+1}, a_{t+1} \mid s_t, a_t)\, ds_{t+1}, a_{t+1}$$

여기서 $s_{t+1}$ 와 모든 가능한 action $a_{t+1}$ 에 대해 optimal한 $Q(s_{t+1}, a_{t+1}) = Q^*(s_{t+1}, a_{t+1})$가 주어져있다고 가정해보자.

$Q^*(s_{t+1}, a_{t+1})$ 는 $(s_{t+1}, a_{t+1})$ 이후의 모든 sequence에서 optimal한 policy를 따랐을 때의 Q-value이다. 즉, True Q-value 이다. 그러면

$Q(s_t, a_t) = \int_{s_{t+1}, a_{t+1}} (R_t + \gamma\, Q^*(s_{t+1}, a_{t+1}))\, \mathrm{P}(s_{t+1}, a_{t+1} \mid s_t, a_t)\, ds_{t+1}, a_{t+1}$ 는 계산할 수 있다.

- **$Q^*$ 를 얻으려면?**

함수의 기대값을 얻으려면: $\mathbb{E}[f(x)] = \int_x f(x)\mathrm{P}(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$ (Monte-Carlo 방식)

$Q^*$ 를 얻으려면 가능한 모든 $(s_t, a_t)$ pair의 $Q(s_t, a_t)$에 대해 아래 업데이트 과정을 충분히 많이 수행하여 $Q^*(s_t, a_t)$로 수렴하게 만든다.

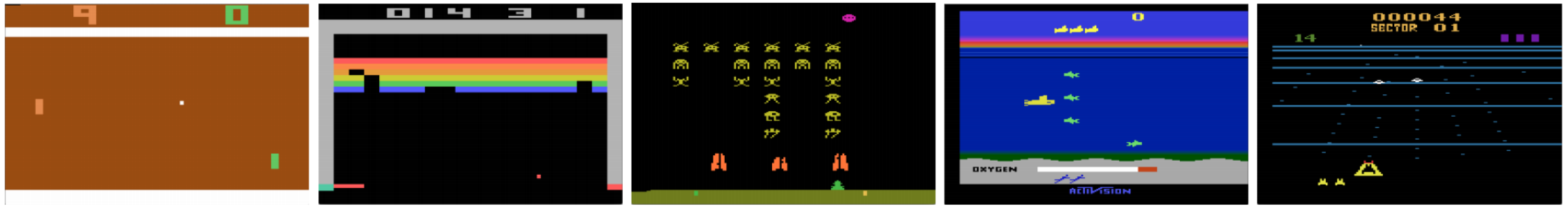
$$Q(s_t, a_t) \leftarrow (1 - \alpha)\, Q(s_t, a_t) + \alpha\, (R_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}))$$ (여기서 $\alpha \in [0, 1]$)

$Q(s_t, a_t)$가 수렴하면, $Q(s_t, a_t)$를 $Q^*(s_t, a_t)$로 여긴다.

# Deep Q-Network

- **DQN = Deep Q Network = Q-Learning + DNN**

강화학습을 사용하여 Atari 2600 game을 플레이하는 Agent를 만들기 위해 개발됨.
State의 수가 무한대에 가까운 game 환경 특성을 극복하기 위해 DNN을 사용함.




Screen shots from five Atari 2600 Games: (Left-to-right) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

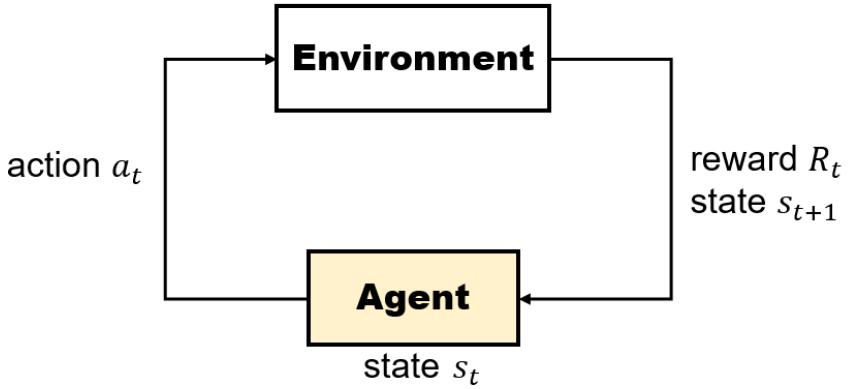
# Deep Q-Network

- **Data**

Experience replay 기법을 사용하여 데이터를 수집함.

Agent가 게임을 플레이하며, 각 time-step(매 frame) $t$에서 Agent의 experience $e_t = (s_t, a_t, R_t, s_{t+1})$ 를 수집하여 replay memory에 저장함.

수행할 action $a_t$ 은 $\epsilon$-greedy 방식으로 선택함. ($\epsilon$ 값의 확률로 랜덤 $a_t$ 를 선택하고, $1-\epsilon$ 확률로 $\underset{a_t}{\text{argmax}} Q(s_t, a_t)$를 선택함)




memory에 한계가 있기 때문에 마지막 $N(=1{,}000{,}000)$개의 experience를 사용함.

$s_t$ : 게임 화면 이미지를 gray-scale로 변경하고, 84x84 크기로 rescale함. 그런 다음 현재에서부터 과거 4 frame 까지의 이미지를 stack 하여 $s_t$ 로 사용함.

$a_t$ : 가능한 모든 조작

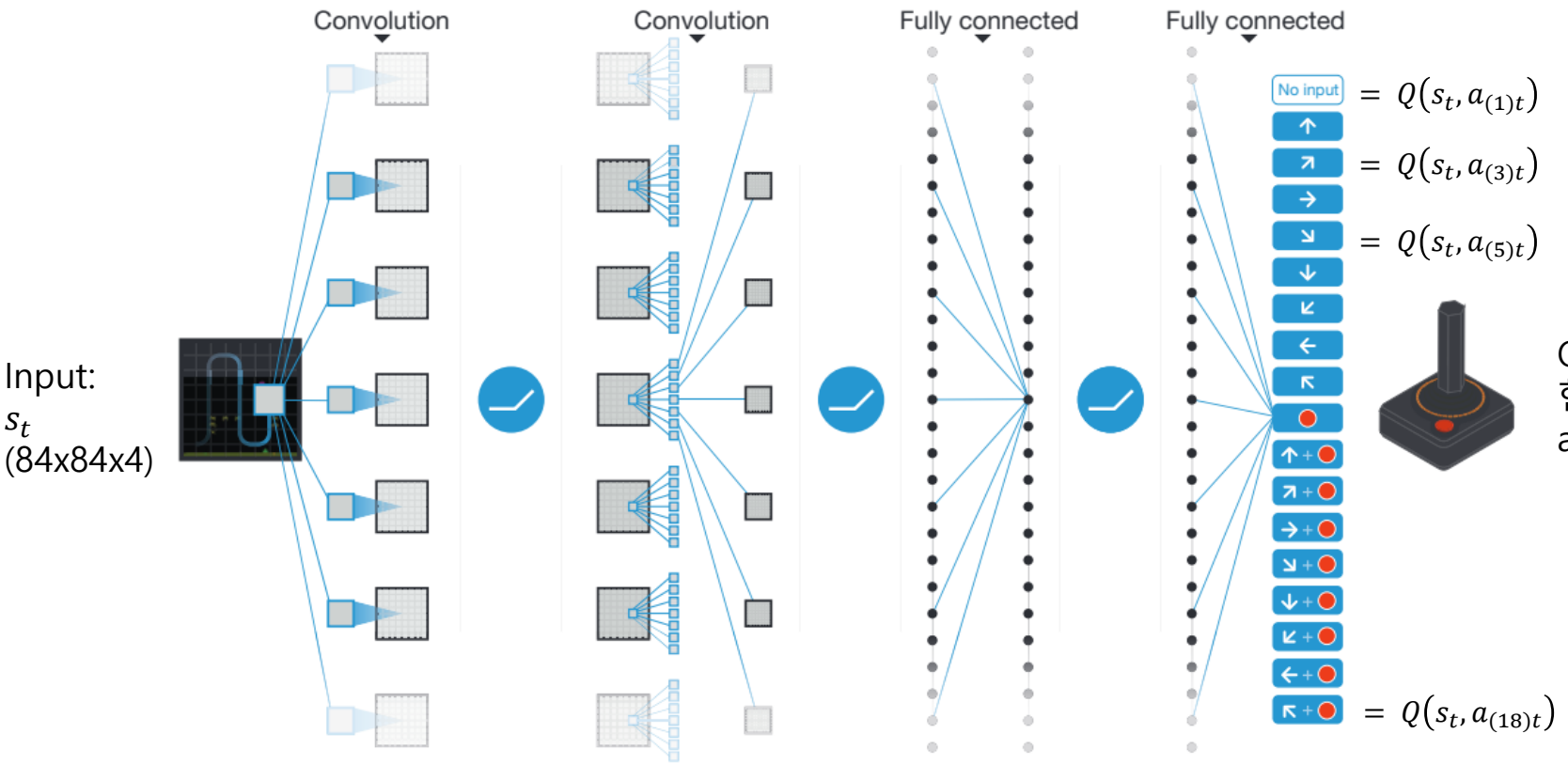
$R_t$ : {-1, 0, 1}

- **Input**

Replay memory에서 mini-batch(=32) 만큼의 데이터를 무작위로 샘플링하여 Input으로 사용함.

# Deep Q-Network

- **Deep Q-Network Architecture**




Output:
현재 state에서 가능한 모든 action에 대한 Q-value

CNN을 통해 Q-value를 regression 하는 이유는, 게임 환경에서의 state의 수는 무한대에 가깝기 때문이다.
제한된 수의 state를 사용하여 Q-value를 학습하고, 만약 본 적 없지만 학습된 state와 비슷한 state가 입력되었을 경우 학습된 Q-value와 비슷한 Q-value를 출력한다.

네트워크가 학습이 잘 되었다면, 현재 state $s_t$에서 action $a'_t = \underset{a_t}{\text{argmax}} Q(s_t, a_t)$를 선택하면 최선의 선택이다.

# Deep Q-Network

- **Training Deep Q-Network**

$$L(\theta) = \mathbb{E}[\left(y_i - Q(s_t, a_t\,;\theta)\right)^2]$$
where $y_i = R_t + \gamma \max_{a_{t+1}} Q(s_{t+1}\,, a_{t+1}\,; \theta^-)$

$$\theta \;\coloneqq \theta \; - \alpha \frac{dL(\theta)}{d\theta}$$

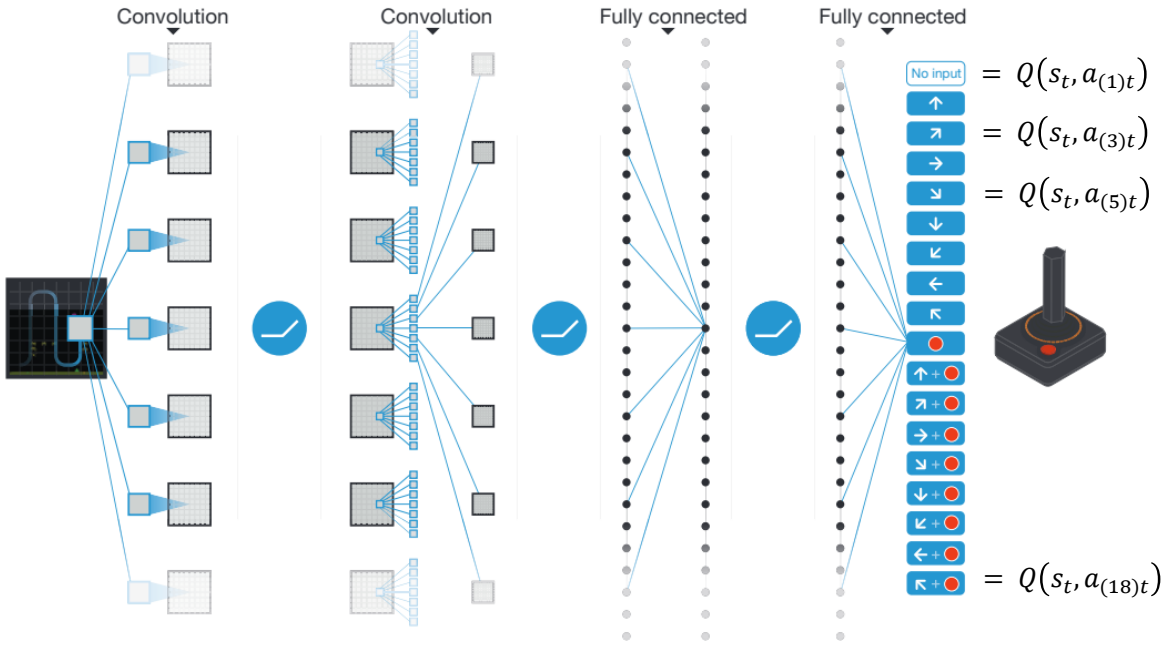
$\theta$: Deep Q-Network의 파라미터. 네트워크의 학습 목표. Q-value를 추정하는 function 이다.
$\theta^-$: $\theta$를 그대로 복사한 파라미터. $\theta^-$파라미터는 학습되지 않으며, 매 C step (=10000 step) 마다 $\theta$를 그대로 덮어씌운다.

**Training**
1. 네트워크에 현재 state $s_t$를 입력하고, $\epsilon$-greedy 방식으로 수행할 action을 선택한 다음 이에 대한 Q-value $Q(s_t, a_t\,;\theta)$를 얻는다.

2. 그런 다음 다음 state $s_{t+1}$를 고정된 파라미터 $\theta^-$ 를 사용하는 네트워크에 입력하여 true(라고 여기는) Q-value $y_i = R_t + \gamma \max_{a_{t+1}} Q(s_{t+1}\,, a_{t+1}\,; \theta^-)$
를 계산한다.

3. 이 둘의 MSE를 줄이는 방향으로 네트워크를 학습시킨다.

(DQN은 총 1,000만 frame동안 학습하였음)




# Deep Q-Network

- **Deep Q-Network와 Q-Learning과의 관계?**

**Deep Q-Network: Training $\theta$**

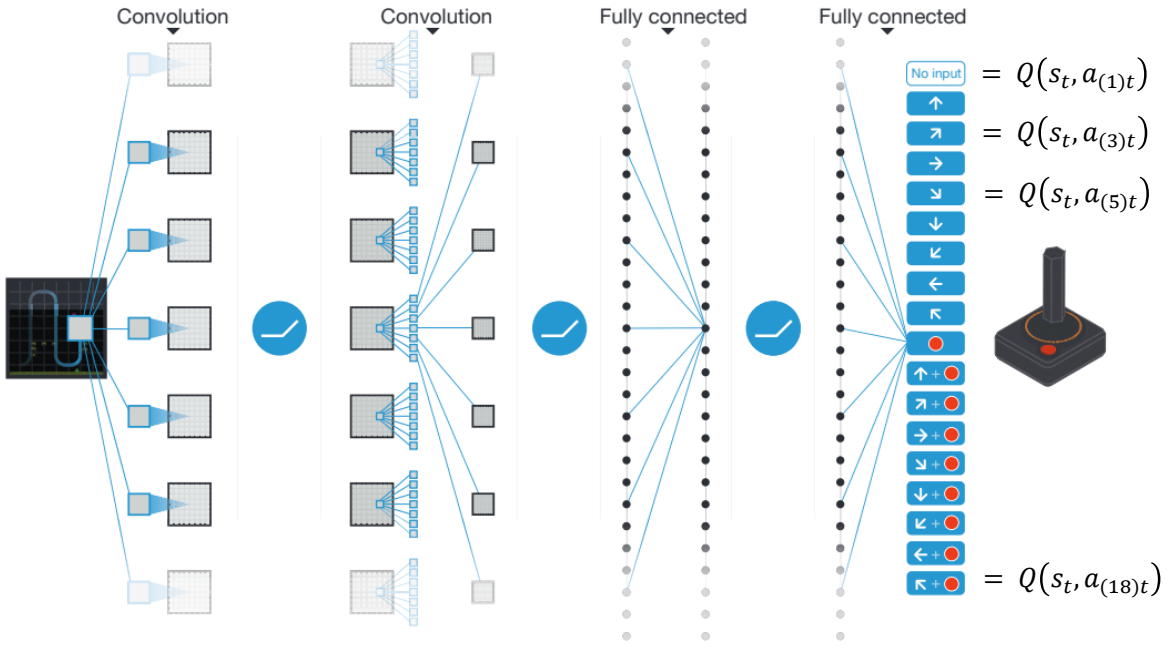
$$L(\theta) = \mathbb{E}[\left(y_i - Q(s_t, a_t\,; \theta)\right)^2]$$
where $y_i = R_t + \gamma \max_{a_{t+1}} Q(s_{t+1}\,, a_{t+1}\,; \theta^-)$

$$\theta \;\coloneqq \theta \; - \alpha \frac{dL(\theta)}{d\theta}$$

**Q-Learning: Update Q-value**

$$Q(s_t, a_t) \leftarrow \; (1-\alpha)\, Q(s_t, a_t) + \alpha \, (R_t \; + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}))$$

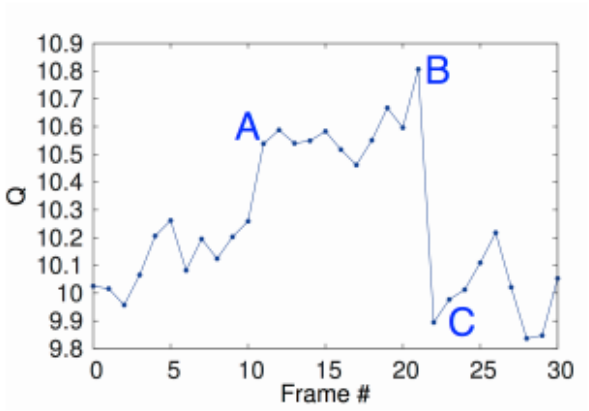
둘 다 $Q(s_t, a_t)$ 가 True Q-value로 수렴하도록 학습한다는 것은 동일.
학습 방식은 약간의 차이가 있음.




# Deep Q-Network

- **Visualization of Q-value**

  Atari game Seaquest




A

B

C

- **Performance**

| | B. Rider | Breakout | Enduro | Pong | Q*bert | Seaquest | S. Invaders |
|---|---|---|---|---|---|---|---|
| **Random** | 354 | 1.2 | 0 | −20.4 | 157 | 110 | 179 |
| **Sarsa** [3] | 996 | 5.2 | 129 | −19 | 614 | 665 | 271 |
| **Contingency** [4] | 1743 | 6 | 159 | −17 | 960 | 723 | 268 |
| **DQN** | **4092** | **168** | **470** | **20** | **1952** | **1705** | **581** |
| **Human** | 7456 | 31 | 368 | −3 | 18900 | 28010 | 3690 |