

# 파이썬과 케라스로 배우는 강화학습

저자 이웅원

RLCode

# 목차

---

1. 저자 소개
2. 책 소개
3. 강화학습이란 무엇인가
4. DQN 이해하기
5. 앞으로의 강화학습

# 책 소개

# 파이썬과 케라스로 배우는 강화학습



<http://wikibook.co.kr/reinforcement-learning/>

**왜 책을 쓰게 되었는가**

- DCULab 자율주행드론



**PID 계수 자동 튜닝을 위해 강화학습을 배워보자**

- David Silver의 RL Course 강의



<https://www.youtube.com/watch?v=2pWv7GOvuf0>

## • 홈페이지에 강화학습 포스팅

졸업
Swift Programming ▾
PyProject - Web Data ▾
생각하는레고 ▾
VRGlass ▾
Python기초 ▾
안티드론 ▾
PyProject - IoT ▾
알고리즘 트레이딩 ▾
OpenRL ▾
▶ 자유게시판
▶ 연구노트
▶ 자료실
회로스터디 ▾
범죄통계분석 ▾
Unreal Lab ▾
...

### 강화학습 그리고 OpenAI - 1: Introduction to OpenAI

👤 이용원 🕒 2016.07.01 20:50 👁 조회 수 : 8676 📌추천:18

2016.07.12 Leewoongwon

## Reinforcement Learning 그리고 OpenAI

### <Contents>

#### 1. [Introduction to OpenAI](#)

#### 2-1. [Intro to Reinforcement Learning \(1\) MDP & Value Function](#)

#### 2-2. [Intro to Reinforcement Learning \(2\) Q Learning](#)

#### 3-1. [CartPole with Deep Q Learning \(1\) CartPole example](#)

#### 3-2. [CartPole with Deep Q Learning \(2\) DQN](#)

#### 3-3. [CartPole with Deep Q Learning \(3\) TensorFlow](#)

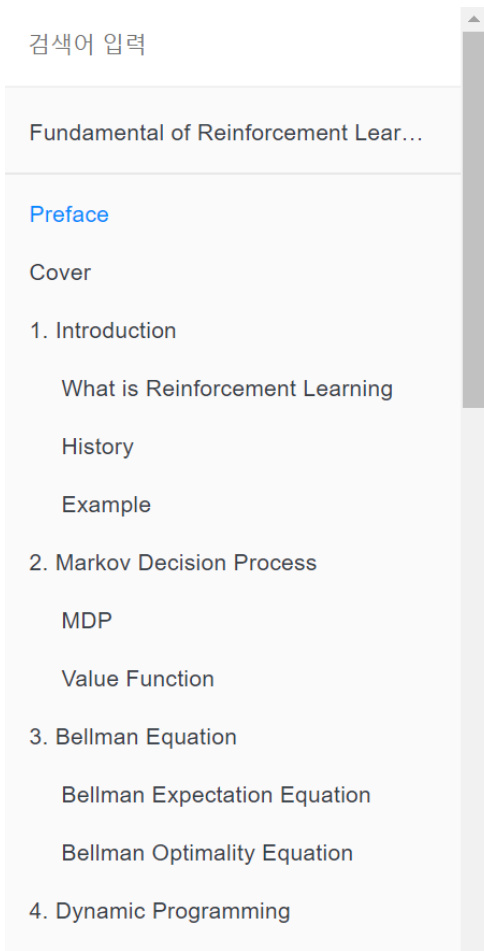
#### 3-4. [CartPole with Deep Q Learning \(4\) Code review](#)

#### 4-1. [CartPole with Policy Gradient \(1\) Policy Gradient](#)

#### 4-2. [CartPole with Policy Gradient \(2\) Code review](#)



## • 이론을 집중적으로 설명



A screenshot of a GitBook interface showing the table of contents for the book 'Fundamental of Reinforcement Learning'. The interface includes a search bar at the top, the book title, and a list of chapters and sections. The 'Preface' section is highlighted in blue.

검색어 입력
Fundamental of Reinforcement Lear...
<b>Preface</b>
Cover
1. Introduction
What is Reinforcement Learning
History
Example
2. Markov Decision Process
MDP
Value Function
3. Bellman Equation
Bellman Expectation Equation
Bellman Optimality Equation
4. Dynamic Programming

## Fundamental of Reinforcement Learning

### Author : Woong won, Lee

- (주)제이마플 연구원
- RLCode(Reinforcement Learning Code)
- DCULab(Drone Control and Utiliaztion Laboratory), 모두의연구소
- Mechanical Engineering, Yonsei University

### How to start

2016년 초부터 모두의 연구소의 자율주행 드론 연구실 DCULab의 연구실장을 맡아서 드론을 연구해오고 있었습니다. 그러던 중에 "드론의 제어에 사용되는 PID 계수를 자동으로 맞춰주는 방법이 있을까?"라는 생각이 들었고 찾아보니 Reinforcement learning을 접하게 되었습니다. 마침 모두의 연구소에서 강화학습 스터디가 시작되었고 그 때부터 David Silver교수님의 강의를 듣기 시작했습니다

# 부족한 것 필요한 것 해야 할 것

---

학습 = 이론 + 실습



학습 = 이론 + 실습

이론과 실습 두 마리 토끼를 잡는 책을 집필해보자!

# 책이 다루는 내용

---

- 강화학습의 배경과 개념
- 강화학습을 위한 기본적인 이론: MDP, 벨만 방정식, 다이내믹 프로그래밍
- 고전 강화학습 알고리즘: 몬테카를로, 살사, 큐러닝
- 인공지능경망을 이용한 강화학습 알고리즘: 딥살사, REINFORCE, DQN, 액터-크리틱, A3C
- 강화학습 알고리즘 구현 및 설명: 그리드월드, 카트폴, 아타리게임

강화학습이란 무엇인가

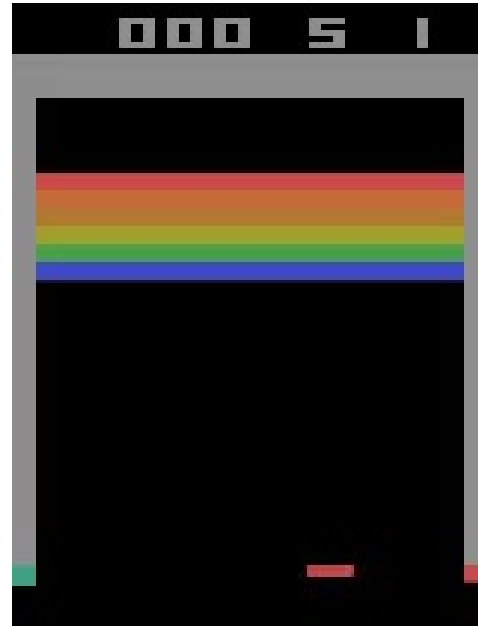
강화학습 → DeepMind → AlphaGo



<http://www.popsci.com/googles-alphago-ai-defeats-lee-se-dol-at-game-go>

# 강화학습의 예시

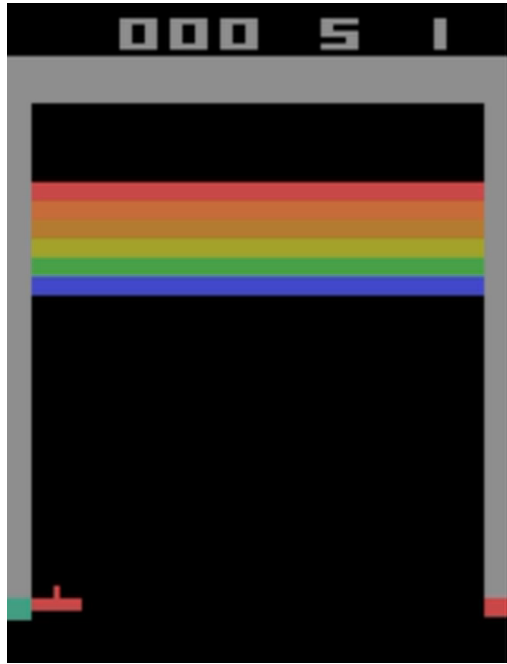
DeepMind → Atari → DQN



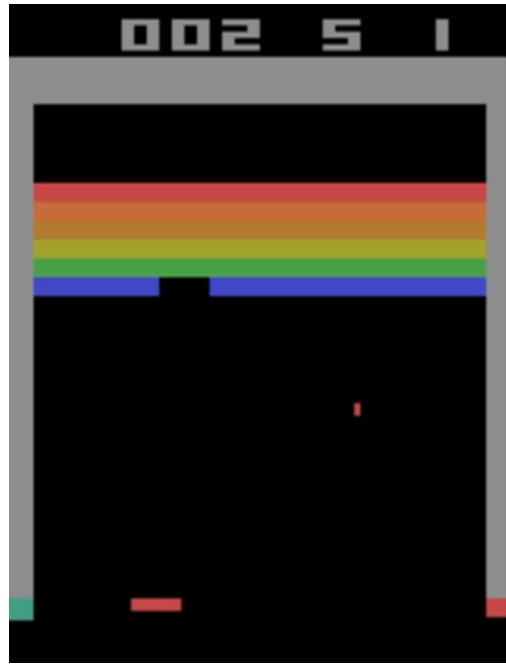
게임 화면으로 게임 플레이하는 법을 학습

# 강화학습의 예시

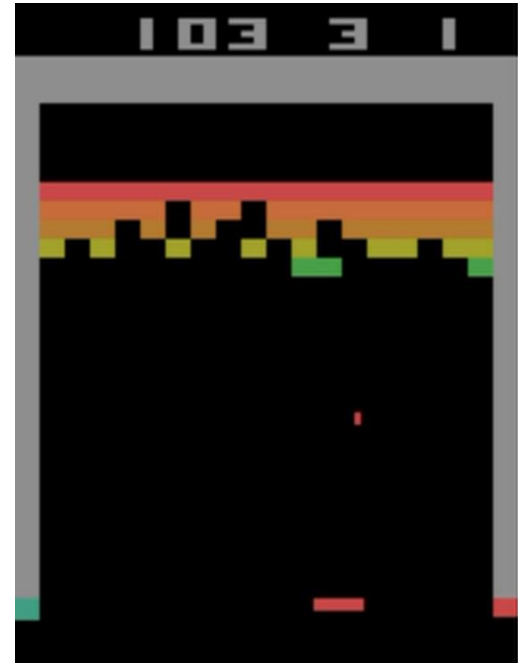
## Atari Breakout 학습 과정



1000 에피소드 학습 후



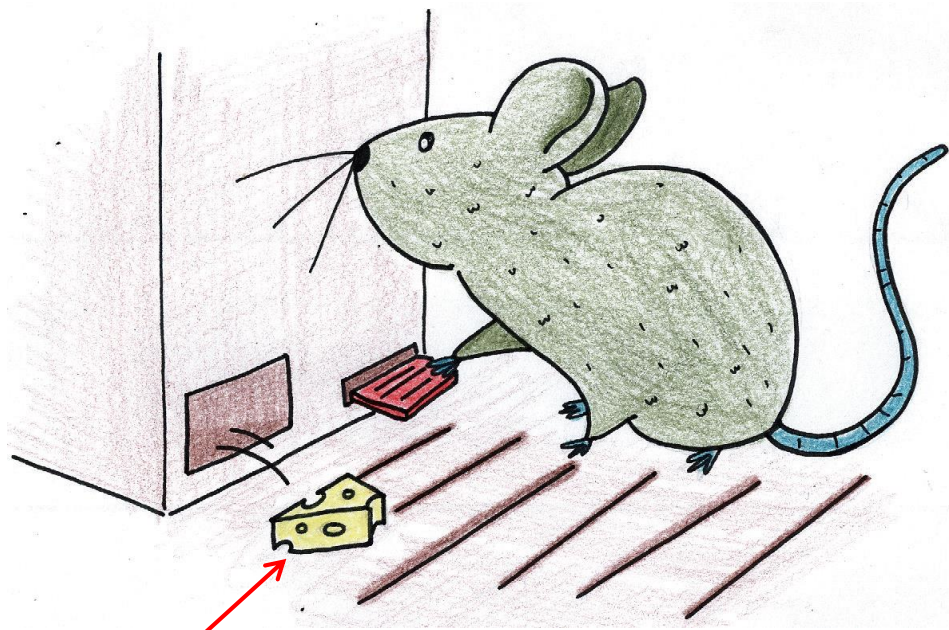
3000 에피소드 학습 후



5000 에피소드 학습 후

# 행동심리학의 강화이론

- 스키너의 쥐 실험

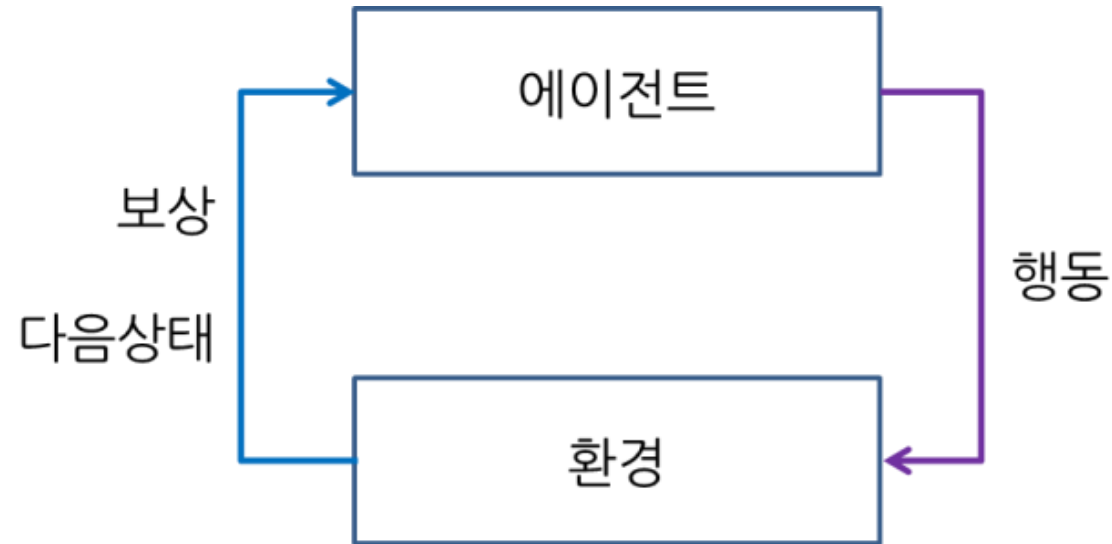


**보상**을 받는 행동의 확률 증가 → **강화**



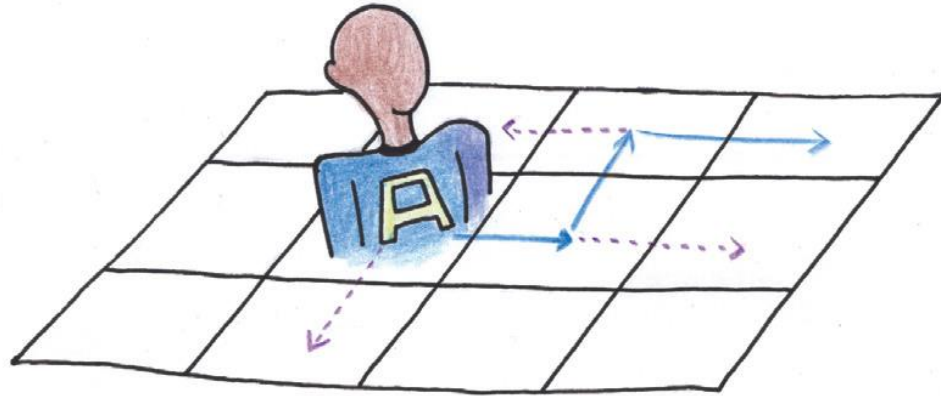
# 학습을 위해 필요한 것

- 핵심은 에이전트와 환경의 상호작용



모르는 단어! [에이전트, 환경] [상태, 행동, 보상]

**에이전트** → 상태를 관찰, 행동을 선택, 목표지향



an autonomous, goal-directed entity which observes and acts upon an environment - 위키피디아

**환경** → 에이전트를 제외한 나머지



판단하는 아이라는 주체를 빼고 길과 자전거와 아이의 몸 또한 환경이 된다

**무엇을 관찰?**

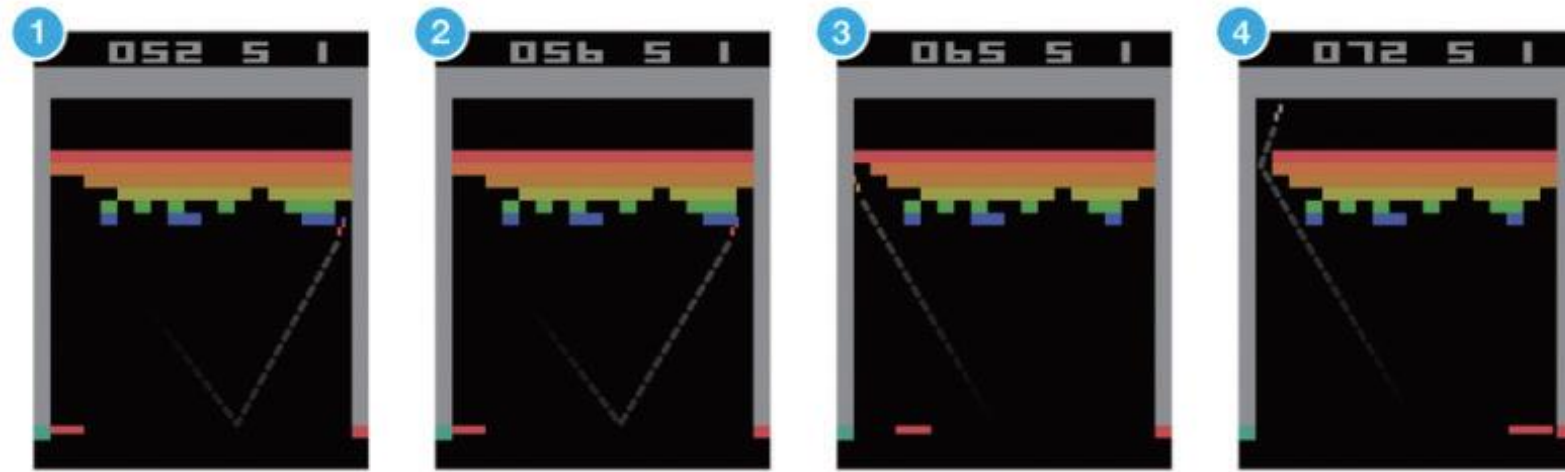
**상태(state), 보상(reward)**

**상태(s)** → 현재 상황을 나타내는 정보



에이전트가 탁구를 치려면 탁구공의 위치, 속도, 가속도와 같은 정보가 필요

**보상(r) → 행동의 좋고 나쁨을 알려주는 정보**

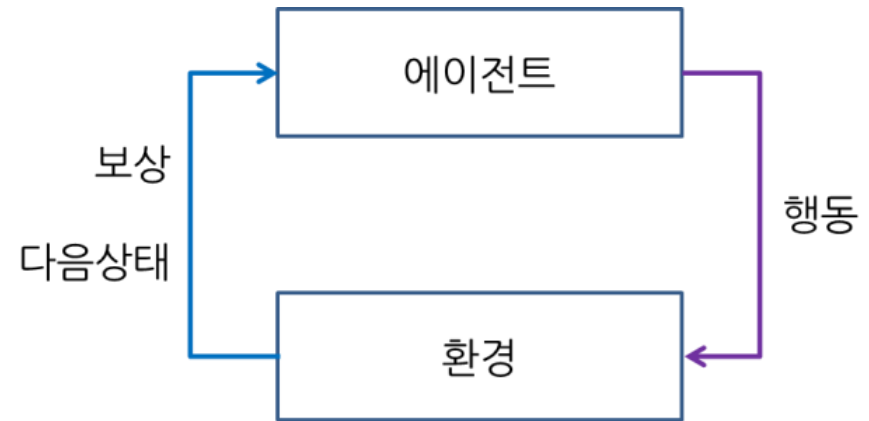


<https://www.intelnervana.com/demystifying-deep-reinforcement-learning/>

**보상은 에이전트가 달성하고자 하는 목표에 대한 정보를 담고 있다**

# 에이전트와 환경의 상호작용 과정

1. 에이전트가 환경에서 자신의 상태를 관찰
2. 그 상태에서 **어떠한 기준**에 따라 행동을 선택
3. 선택한 행동을 환경에서 실행
4. 환경으로부터 다음 상태와 보상을 받음
5. 보상을 통해 에이전트가 가진 정보를 수정함



$$s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_T$$

어떻게 행동을 선택?

판단 기준의 필요 → 가치함수(Value function)



# 가치함수 (Value function)

- 만약 즉각적인 보상만을 고려해서 행동을 선택한다면?



이 행동만이 좋은 행동이고 나머지는 아니다?

# 가치함수 (Value function)

---

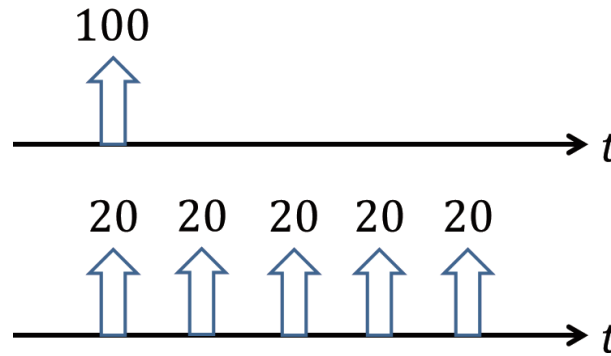
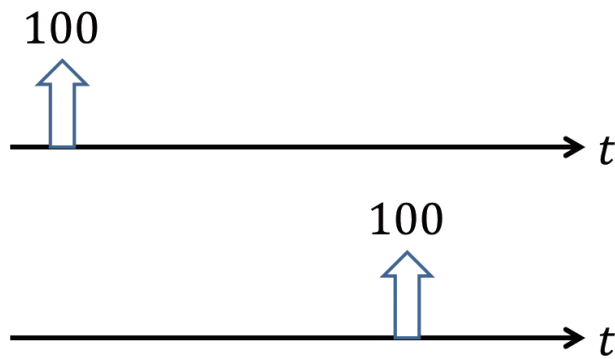
- 보상은 딜레이(delay)된다
- 어떤 행동이 그 보상을 얻게 했는지 명확하지 않다

그렇다면 앞으로 받을 보상을 싹 다 더해보자  
현재 시간 =  $t$

$$\text{보상의 합} = R_{t+1} + R_{t+2} + \cdots + R_T$$

# 가치함수 (Value function)

- 세 가지 문제점 → 감가율의 도입(discount factor)



$$\begin{aligned} 0.1 + 0.1 + \dots &= \infty \\ 1 + 1 + \dots &= \infty \end{aligned}$$

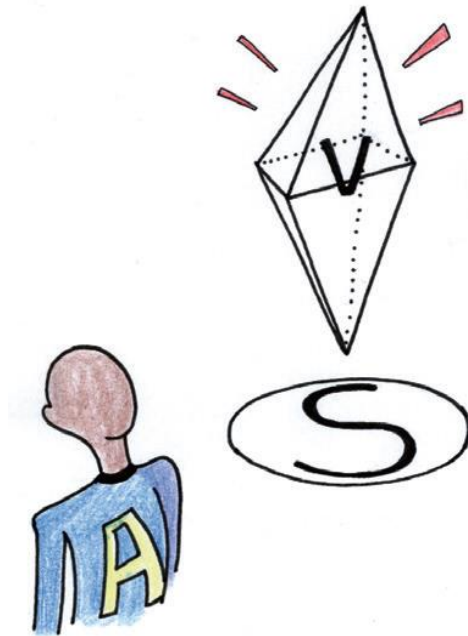
감가율  $0 \leq \gamma \leq 1$

$$\text{보상의 합} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-t-1} R_T$$

# 가치함수(Value function)

- 하지만 아직 보상을 받지 않았는데...? 미래에 받을 보상을 어떻게 알지?

지금 상태에서 미래에 받을 것이라 기대하는 보상의 합 = 가치함수

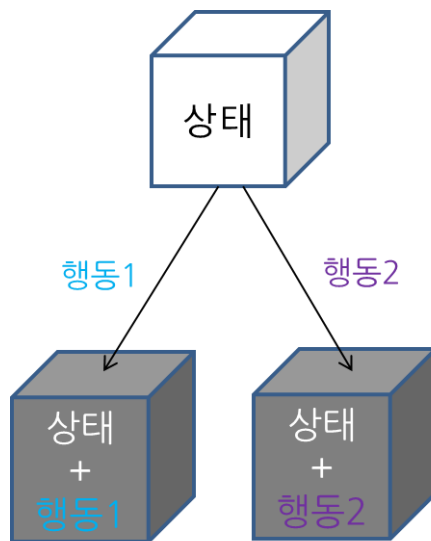


가치함수  $v(s) = E[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s]$

# 큐함수(Q function)

- 하지만 내가 알고 싶은 건 '어떤 행동이 좋은가'인데?

지금 상태에서 이 행동을 선택했을 때 미래에 받을 것이라 기대하는 보상의 합  
= 큐함수



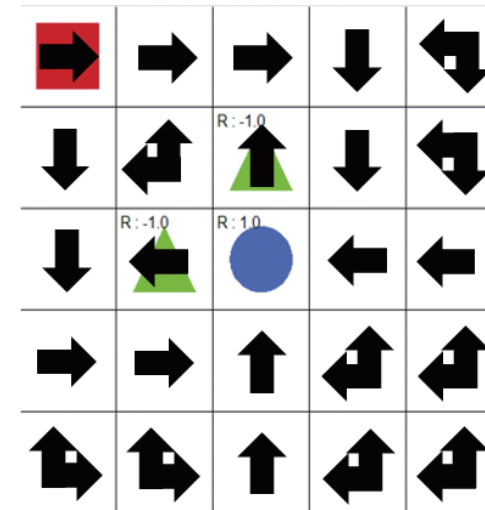
$$\text{큐함수 } q(s, a) = E[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s, A_t = a]$$

# 정책(Policy)

- 미래에 대한 기대  $\rightarrow$  내가 어떻게 행동할 것인지를 알아야 함
- 각 상태에서 에이전트가 어떻게 행동할 지에 대한 정보

상태  $s$ 에서 행동  $a$ 를 선택할 확률

정책  $\pi(a|s) = P[A_t = a | S_t = s]$



가치함수  $v_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s]$

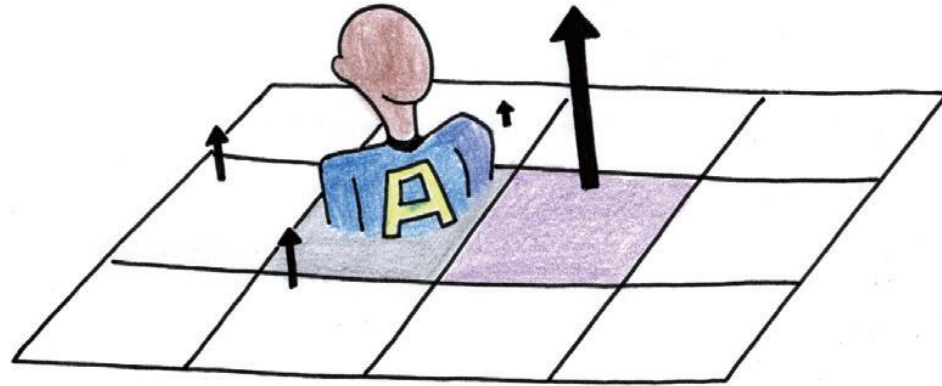
큐함수  $q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s, A_t = a]$

**큐함수를 통해 어떻게 행동을 선택?**

**그냥 큰 놈 골라**

# 탐욕정책(greedy policy)

- 지금 상태에서 선택할 수 있는 행동 중에 큐함수가 가장 높은 행동을 선택



탐욕정책  $\pi'(s) = \operatorname{argmax}_a q_{\pi}(s, a)$



어떻게 학습? **큐함수의 업데이트**

# 벨만 방정식(Bellman equation)

- 에이전트는 모든 (상태, 행동)에 대해서 큐함수를 가진다 → 일종의 기억
- 그렇다면 현재의 큐함수를 다음 타임스텝의 큐함수로 표현할 수 있지 않을까?

$$\text{큐함수 } q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s, A_t = a]$$

너무 먼 미래에 대해서 기대를 품기보다는 가까운 미래에 대해서 구체적인 기대를 품기로 했다

$$q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma(R_{t+2} + \dots) | S_t = s, A_t = a]$$

$$q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

## 벨만 기대 방정식(Bellman expectation equation)

# 살사(SARSA)

- 벨만 기대 방정식 → 큐함수 업데이트 식

$$q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

미래에 대해서 기대만 하기보다는 실제로 부딪혀보면서 학습하기로 했다

**현재 큐함수  $\leftarrow$  보상 + 감가율  $\times$  다음 큐함수**

$$q(s, a) \leftarrow r + \gamma q_{\pi}(s', a')$$

점진적인 큐함수의 업데이트

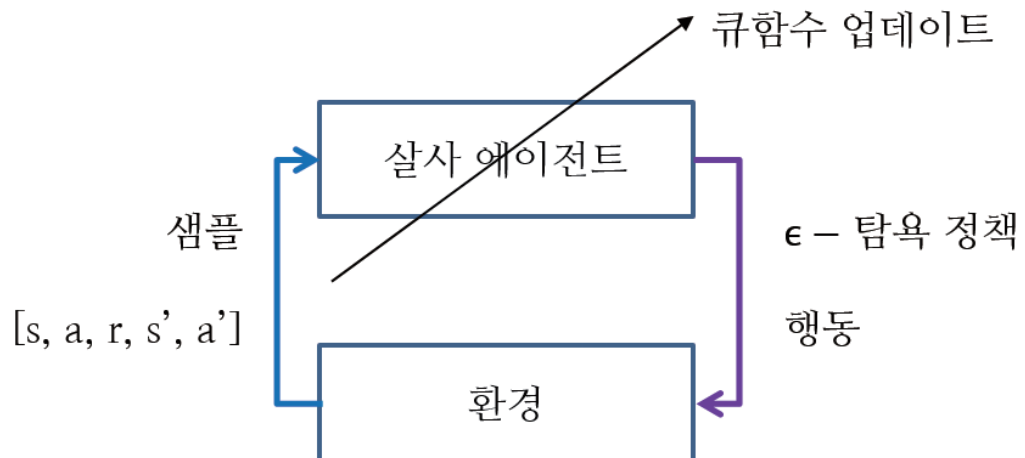
$$q(s, a) = q(s, a) + \alpha(r + \gamma q(s', a') - q(s, a))$$

# 살사(SARSA)

- 현재 큐함수를 업데이트하기 위해서는  $(s, a, r, s', a')$ 이 필요

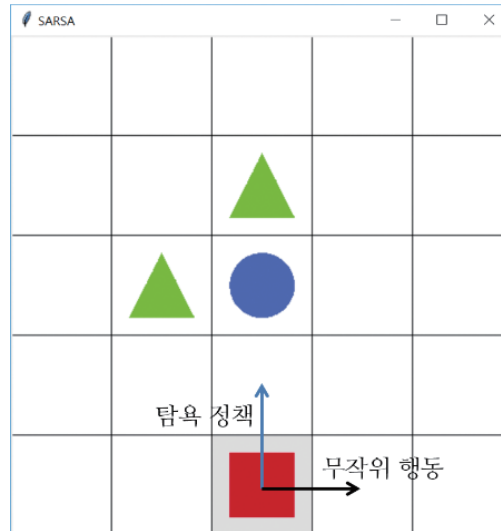
→ 살사(SARSA)

$$q(s, a) = q(s, a) + \alpha(r + \gamma q(s', a') - q(s, a))$$



## $\epsilon$ - 탐욕정책

- 탐욕 정책의 Exploration problem → 일정한 확률로 랜덤하게 행동 선택



$$\text{탐욕정책 } \pi'(s) = \operatorname{argmax}_a q_{\pi}(s, a)$$

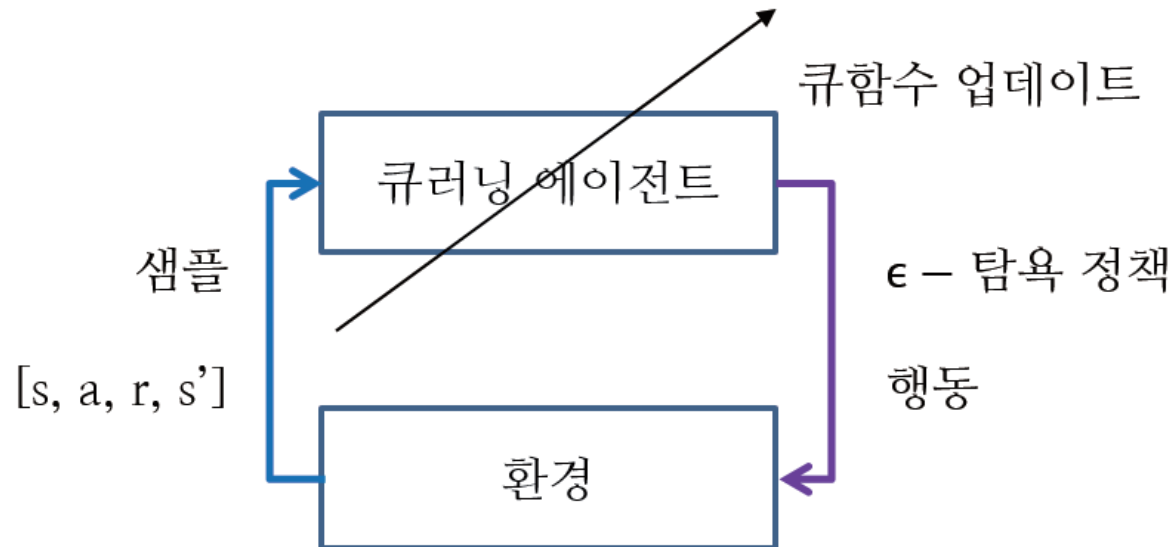
$$\epsilon\text{-탐욕정책 } \pi(s) = \begin{cases} a^* = \operatorname{argmax}_a q(s, a), & 1 - \epsilon \\ a \neq a^*, & \epsilon \end{cases}$$

# 큐러닝(Q-Learning)

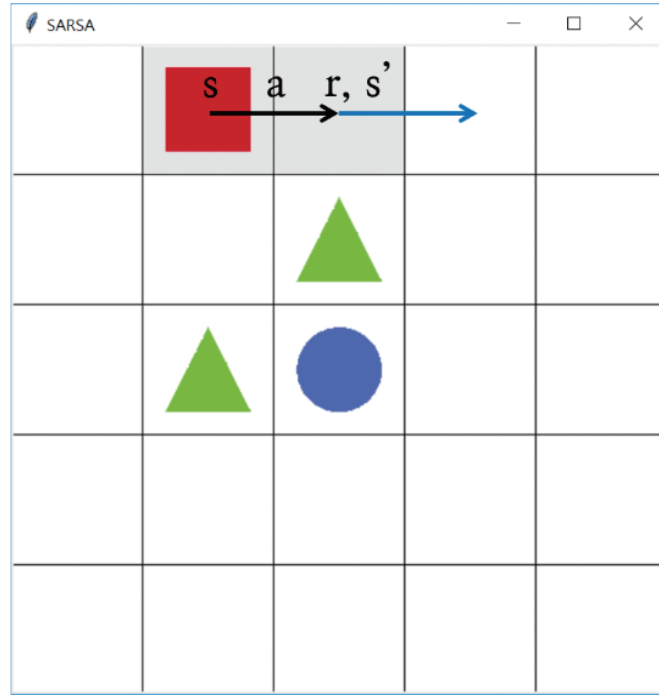
- 기왕 기억을 활용하는 김에 좋은 기억을 활용해보자

→ 다음 큐함수 중에서 가장 값이 큰 큐함수를 이용해서 현재 큐함수를 업데이트  
(Q-Learning)

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



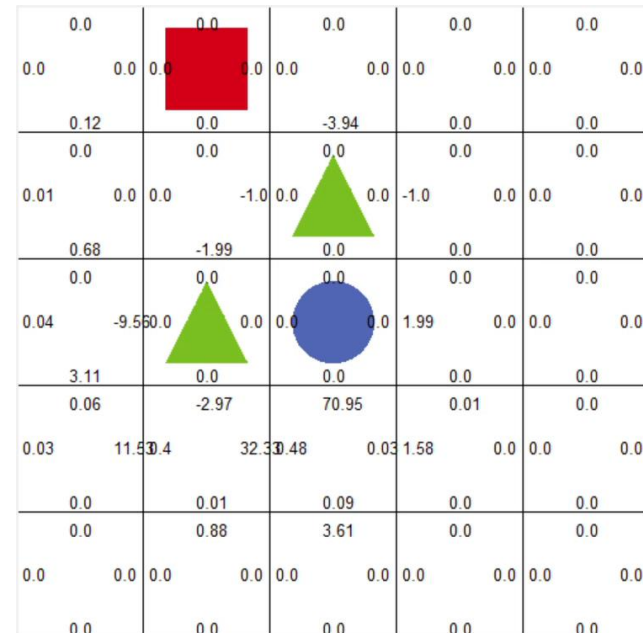
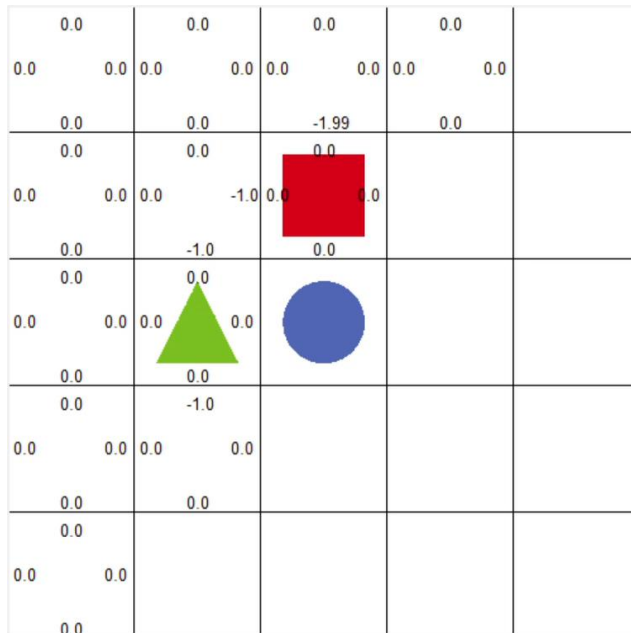
# 큐러닝(Q-Learning)



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

# 그리드월드와 큐러닝

1. 현재 상태에서  $\epsilon$ -탐욕 정책에 따라 행동을 선택
2. 선택한 행동으로 환경에서 한 타임스텝을 진행
3. 환경으로부터 보상과 다음 상태를 받음
4. 다음 상태에서  $\epsilon$ -탐욕 정책에 따라 다음 행동을 선택
5.  $(s, a, r, s')$ 을 통해 큐함수를 업데이트





# DQN 이해하기

# 브레이크아웃(Breakout)

- 상태, 행동, 보상



- 행동 : 제자리, 좌, 우, (발사)
- 보상 : 벽돌 깨 때마다 점수를 받으며 위 층의 벽돌을 깨수록 더 큰 점수를 받음
- 게임 세팅 : 1 에피소드에서 에이전트는 5개의 목숨을 가짐
- 목표 : 1 에피소드 동안 최대의 점수 얻기

# 브레이크아웃(Breakout)

- 브레이크아웃의 상태는 무엇일까? RGB 이미지

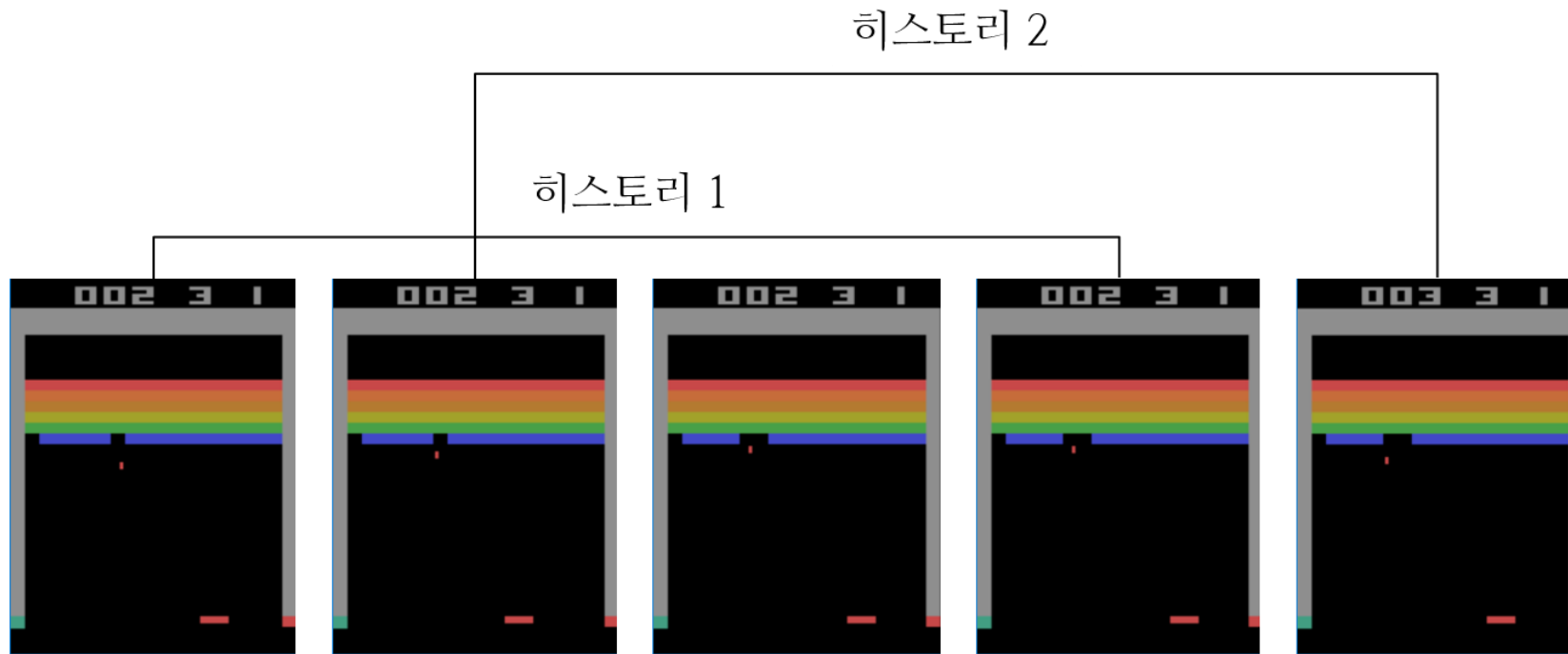


				214	208	64	154	121	122	156	56	135	
				22	36	39	170	232	243	139	178	240	82
				20	18	246	3	91	145	93	118	48	150
				21	115	99	90	116	1	206	164	49	240
12				69	52	139	142	164	203	14	204	121	42
22				57	160	15	53	134	211	118	27	49	232
19				15	221	237	159	234	162	141	185	235	44
14				86	154	109	145	165	254	25	63	75	75
16				11	140	115	241	202	246	109	102	244	159
3				16	104	148	80	13	241	0	114	12	46
7				20	104	31	239	120	157	243	247	87	123
83				59	60	51	245	224	2	233	62	66	64
12				24	35	48	81	38	18	129	196	20	67
61				22	61	174	84	165	34	24	164	46	106
15				14	108	162	180	95	89	33	123	36	128
19				91	44	147	194	189	43	65	222	205	160
35				23	92	34	27	201	127	25	182	36	14
21				55	16	39	83	195	88	135	186	96	161
76				20	109	53	119	15	199	58	30	94	129
15				24	217	123	213	132	104	151	178	73	
10													
1													
52				70	98	195	236	176	234	194	225		

모든 상태의 큐함수를 다 저장하고 업데이트하는 방식으로는 불가능

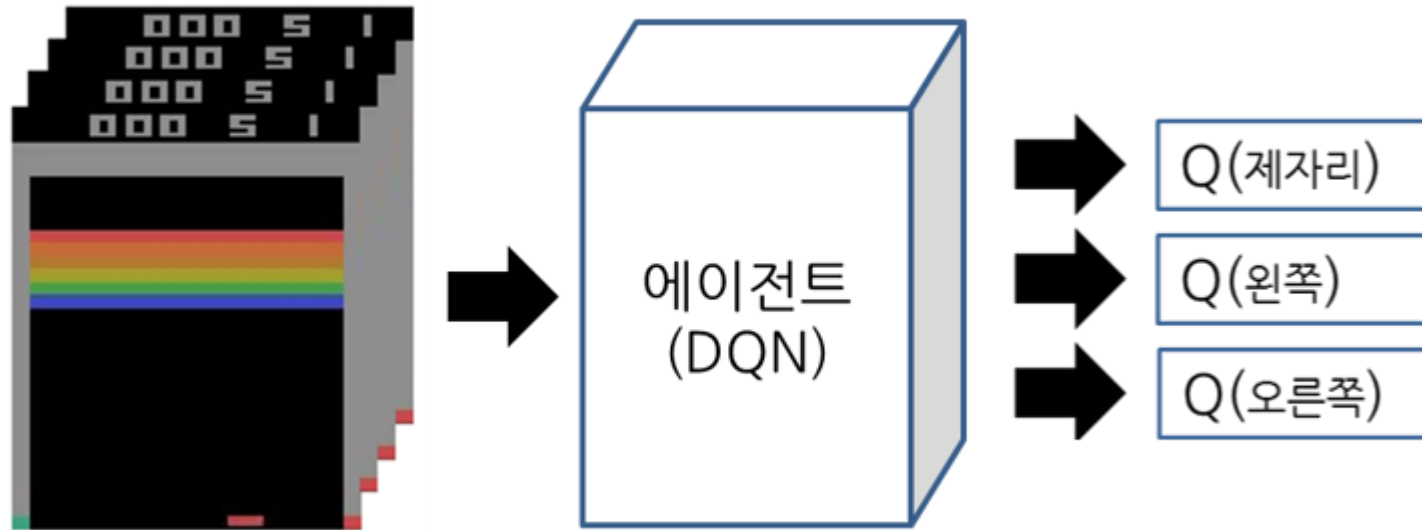
# 브레이크아웃(Breakout)

- 브레이크아웃의 상태는 무엇일까? RGB 이미지 4장으로 이루어진 히스토리



# 브레이크아웃(Breakout)

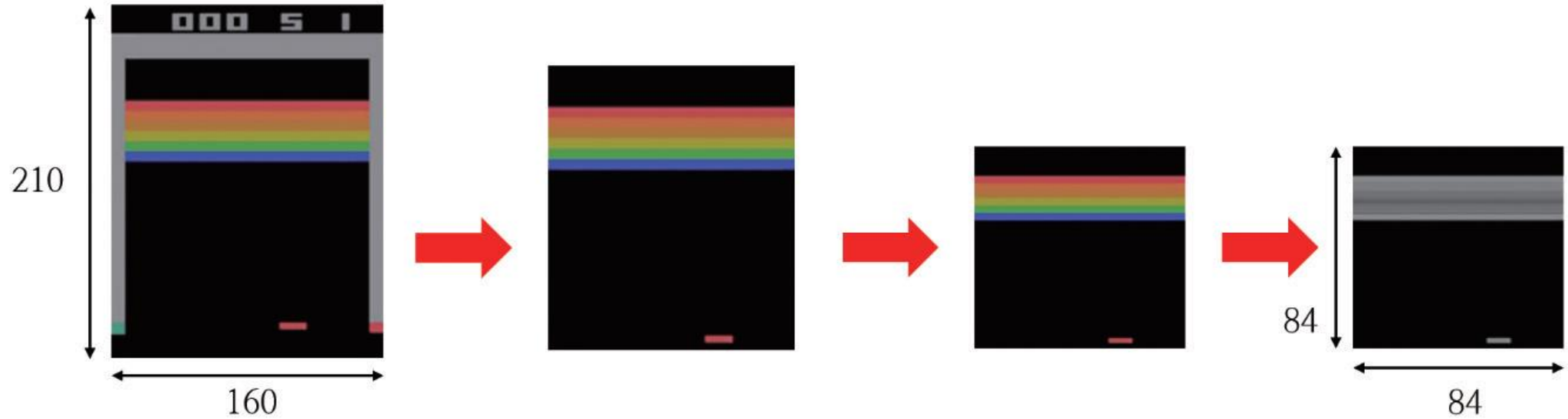
- 큐함수를 인공지능망으로 근사하자!



상태(히스토리) → 큐-신경망 → 각 행동에 대한 큐함수 값

# 브레이크아웃(Breakout)

- 브레이크아웃 이미지의 전처리 과정

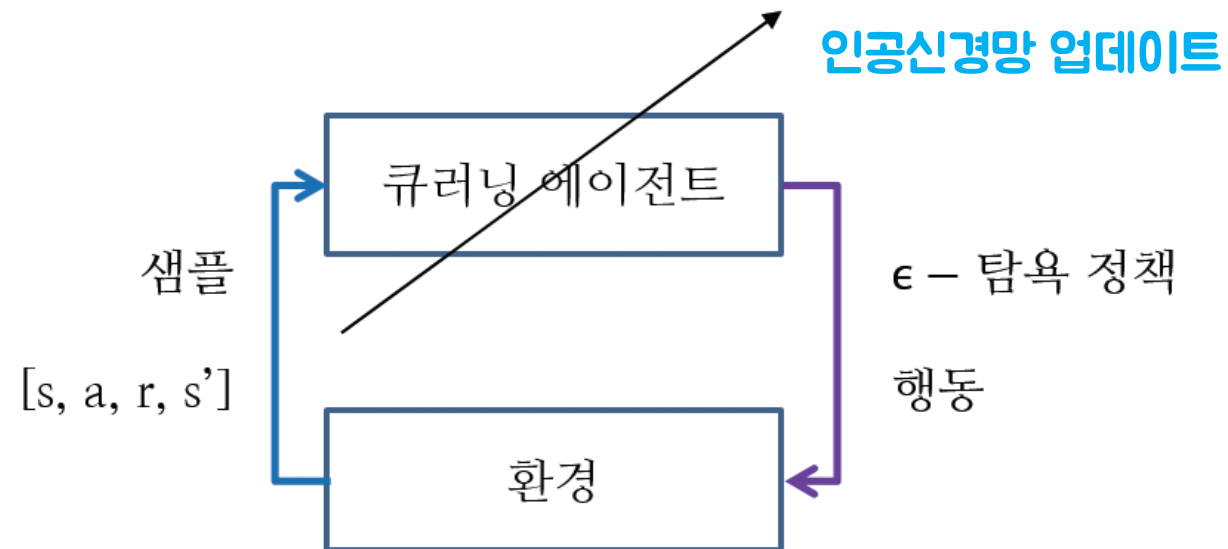


Rescale + grayscale

# 인공신경망 업데이트

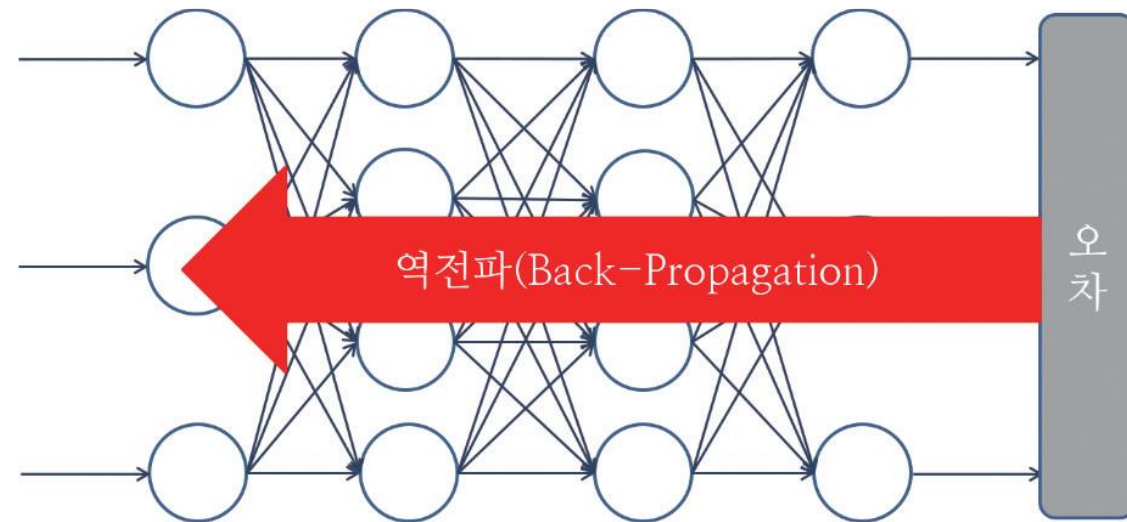
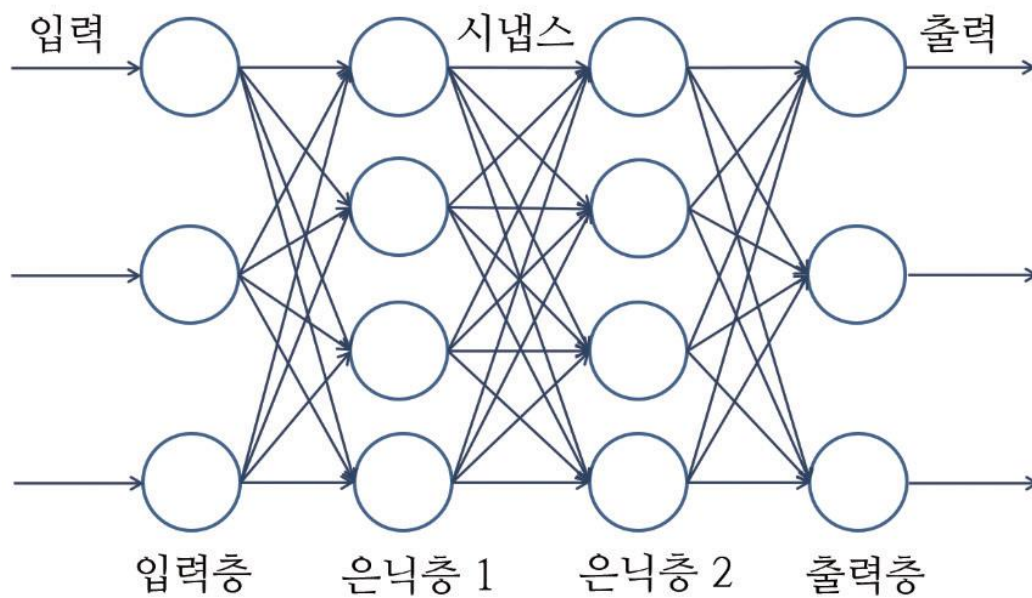
- 큐러닝의 큐함수 업데이트 식

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



# 인공신경망 업데이트

- (정답 - 예측)의 오차를 이용해서 인공신경망에 역전파  
→ 인공신경망의 업데이트





# 인공신경망 업데이트

- 큐러닝의 큐함수 업데이트 식

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

- 큐함수를 인공신경망(parameter  $\theta$ )로 근사

$$q_{\theta}(s, a) = q_{\theta}(s, a) + \alpha(r + \gamma \max_{a'} q_{\theta}(s', a') - q_{\theta}(s, a))$$

- 큐함수가 아닌 큐함수를 근사한 인공신경망을 업데이트(MSE loss function)

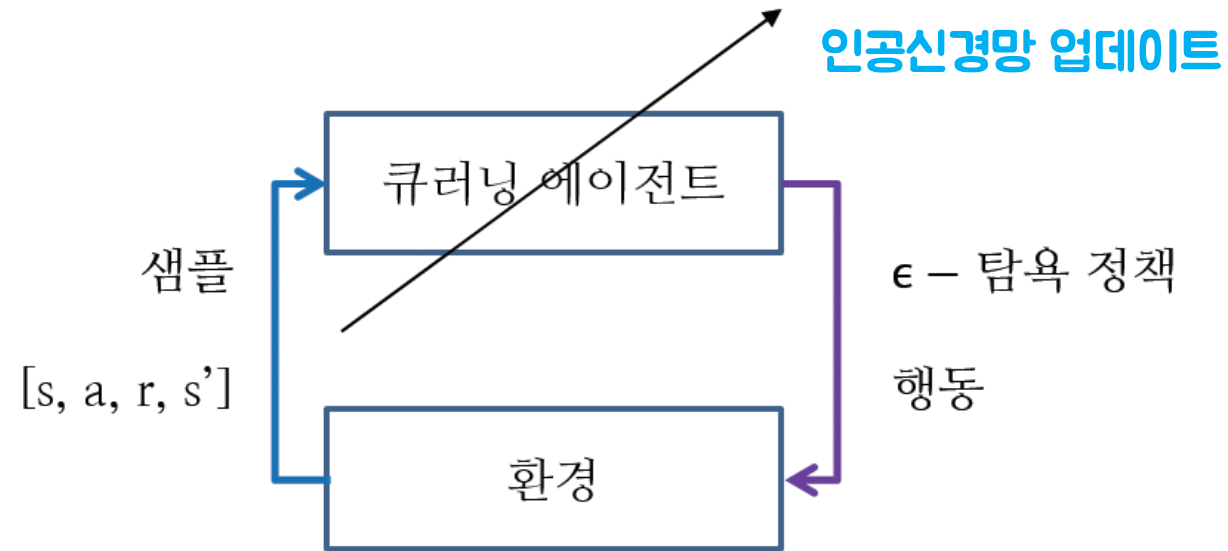
$$MSE = \left( \underbrace{r + \gamma \max_{a'} q_{\theta}(s', a')}_{\text{정답}} - \underbrace{q_{\theta}(s, a)}_{\text{예측}} \right)^2$$

정답

예측

# 인공신경망 업데이트

- 큐-신경망 업데이트



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

- **DQN의 핵심 :**

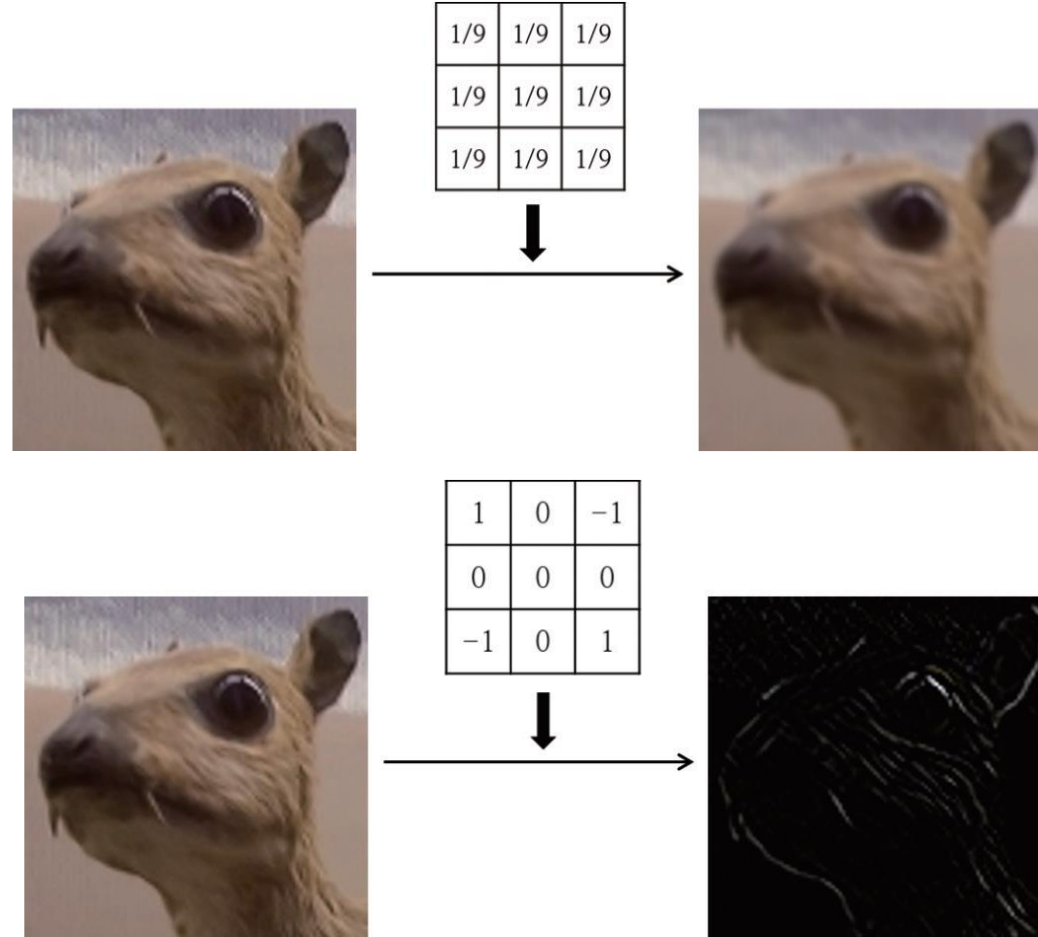
<https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf>

**(1) CNN**

**(2) Experience Replay**

**(3) Target q-network**

- 이미지 필터(커널)의 예시

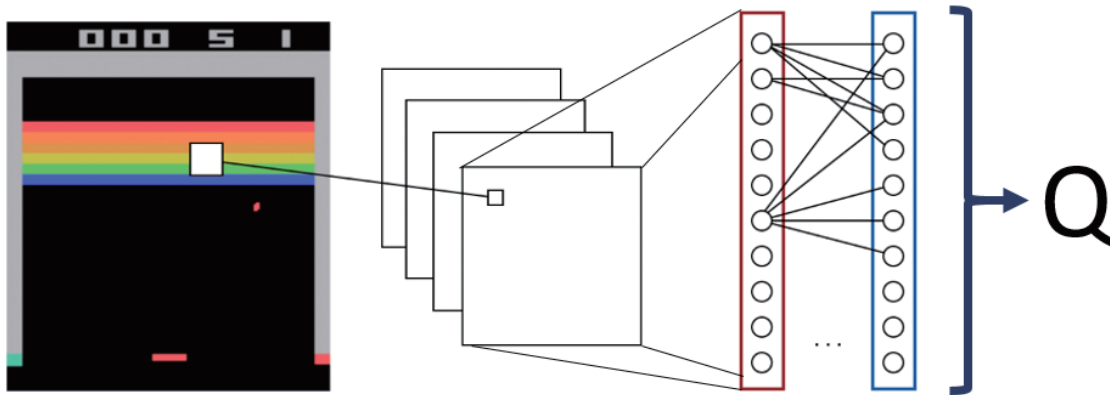


[https://en.wikipedia.org/wiki/Kernel\\_\(image\\_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

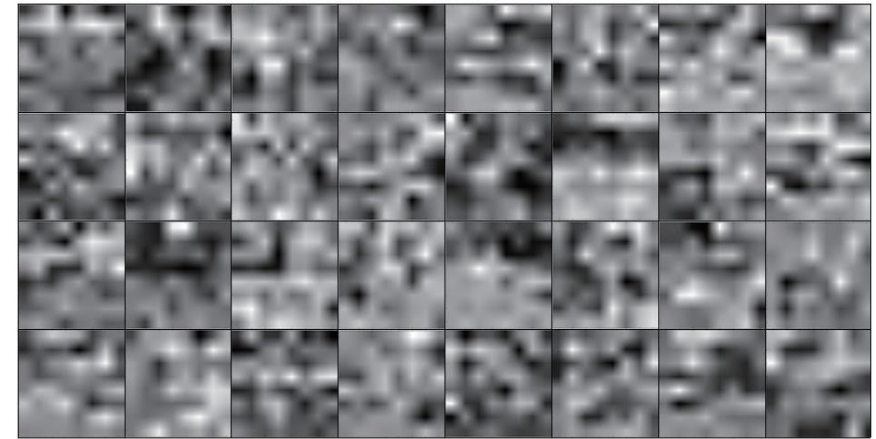
어떤 필터인지에 따라 이미지를 다양하게 변형 가능

# CNN

- 브레이크 아웃 게임 화면에 필터를 컨볼루션!



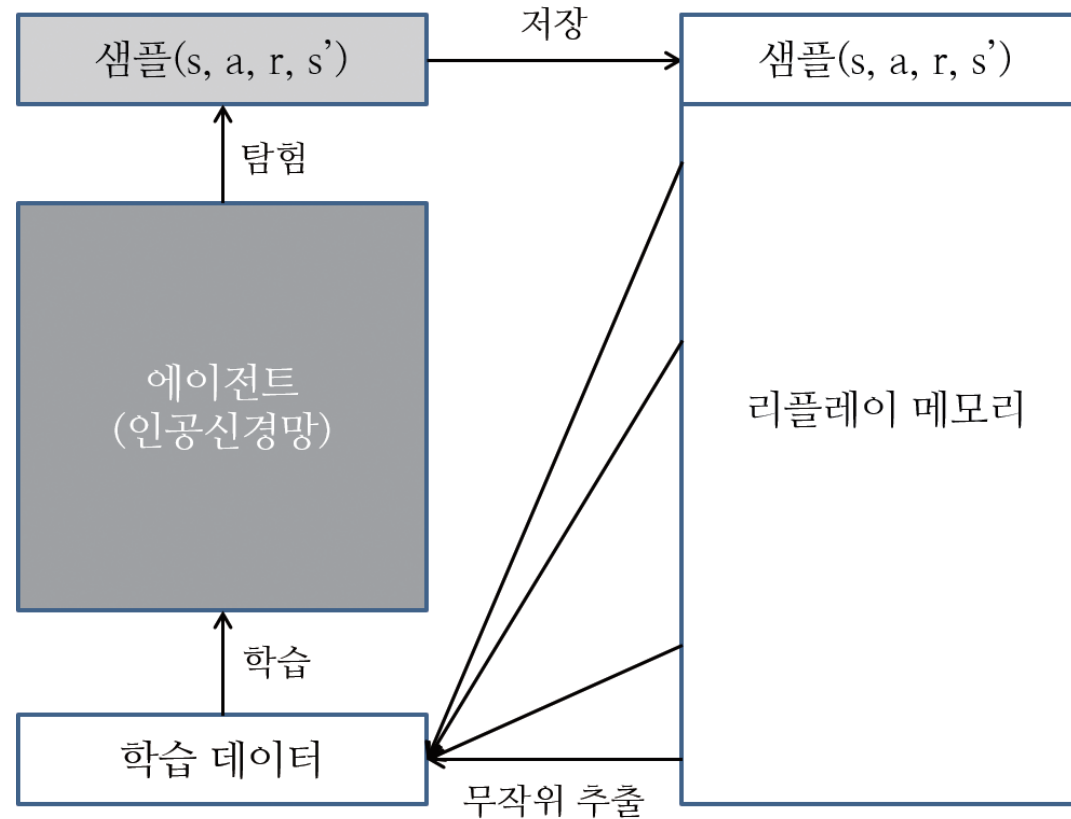
DQN의 CNN



학습된 필터의 예시

# Experience Replay

- $(s, a, r, s')$ 끼리 상관관계가 강하다 → 리플레이 메모리



# Target q-network

---

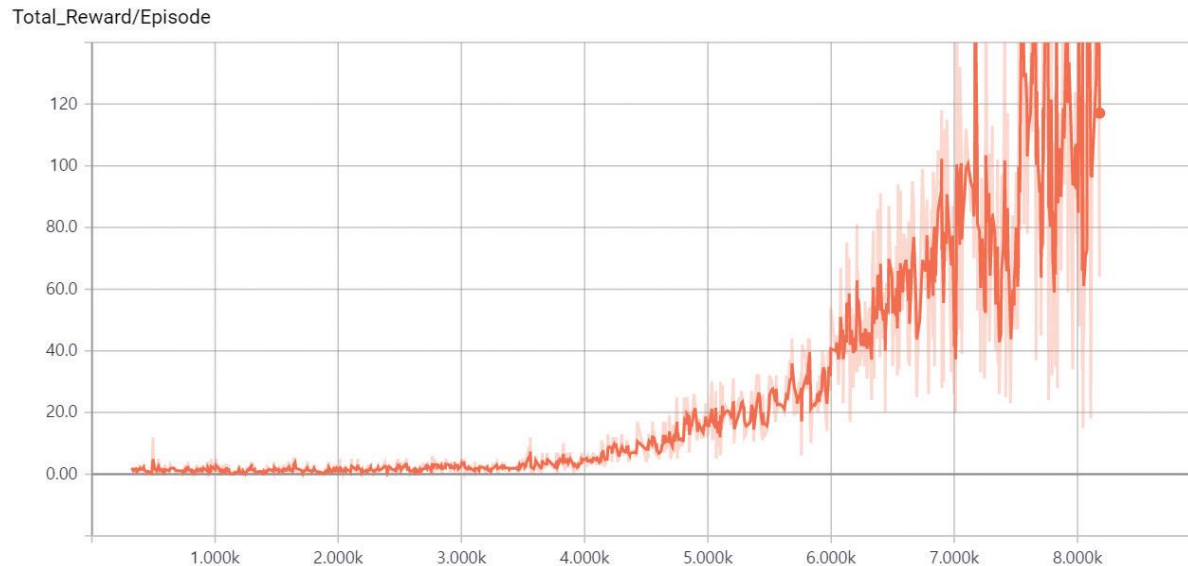
- 안정적인 학습을 위해 학습에 사용하는 큐함수의 값을  
타겟 큐-신경망( $\theta^-$ )에서 가져옴

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

- 타겟 큐-신경망을 일정한 주기마다 업데이트

# Deep Q-Learning

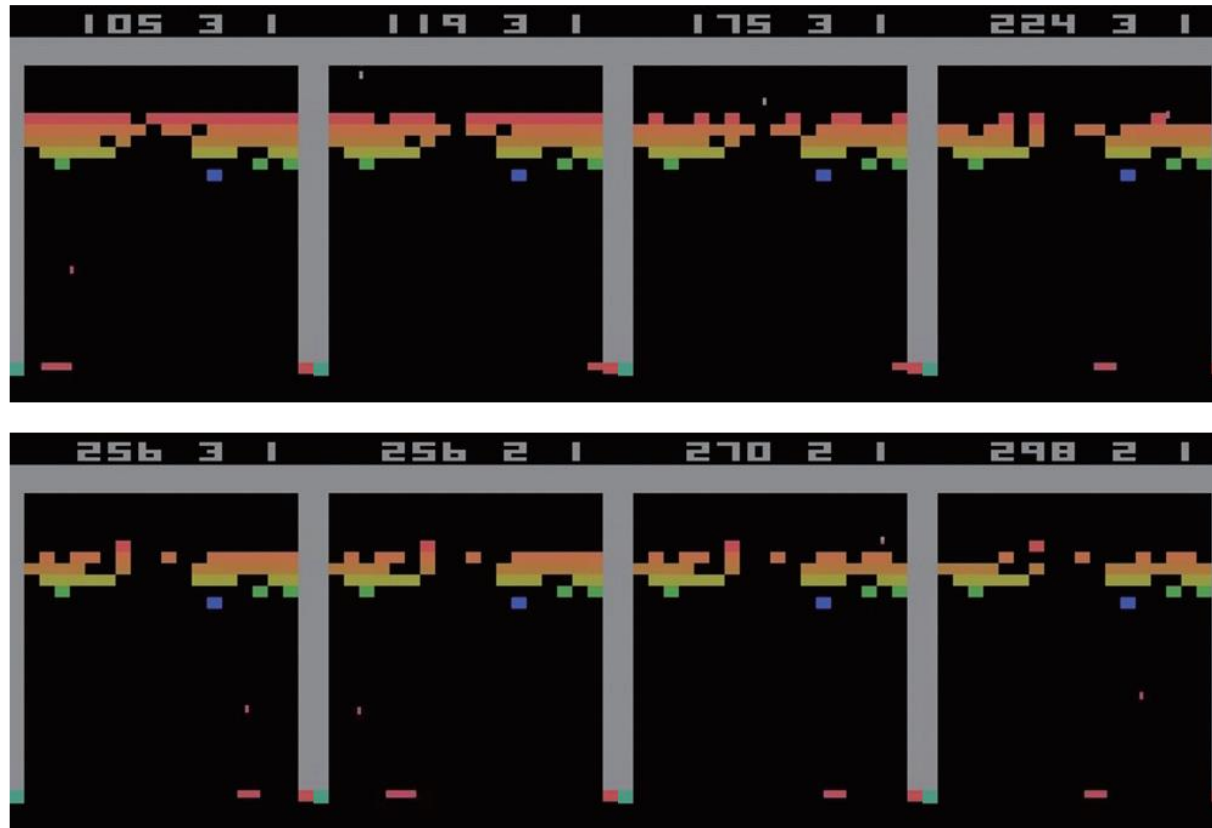
1. 상태에 따른 행동 선택
2. 선택한 행동으로 환경에서 한 타임스텝을 진행
3. 환경으로부터 다음 상태와 보상을 받음
4. 샘플( $s, a, r, s'$ )을 리플레이 메모리에 저장
5. 리플레이 메모리에서 무작위로 추출한 32개의 샘플로 학습
6. 50000 타임스텝마다 타깃네트워크 업데이트





# 학습된 에이전트

터널을 뚫어서 점수를 얻는 정책을 학습



# 앞으로의 강화학습

# 강화학습의 활용

---

- 자율주행 자동차
- 자연어 처리



<https://www.youtube.com/watch?v=cYTVXfIH0MU>



<http://www.maluuba.com/blog/2016/11/23/deep-reinforcement-learning-in-dialogue-systems>

# 강화학습의 단점

---

- 학습이 느리다
- 환경과의 상호작용 때문에 시간과 비용이 많이 들어간다
- Safe하지 않다
- 보상을 적절히 설정하기가 어렵다
- 복잡한 작업을 학습하기엔 어려움이 있다

# 강화학습의 발전 방향

---

- No MDP
  - Hierarchy
  - Multi-agent
  - Efficient learning
- 
- Supervised Learning의 한계를 극복 → 사용자의 피드백으로 학습

감사합니다