PLAYING ATARI WITH DEEP REINFORCEMENT LEARNING

빅데이터융합학과
2021512018 박소미

CONTENT

- Abstract
- Introduction
- Background
- Deep Reinforcement Learning
- Experiment
- Result
- Conclusion

ABSTRACT

과연 딥러닝 기술을 강화학습에 적용해도 효과가 있을 것인가?

DeepMind's Deep Q-learning playing Atari Breakout

Google DeepMind

Publication

Playing Atari with Deep Reinforcement Learning

Download View publication

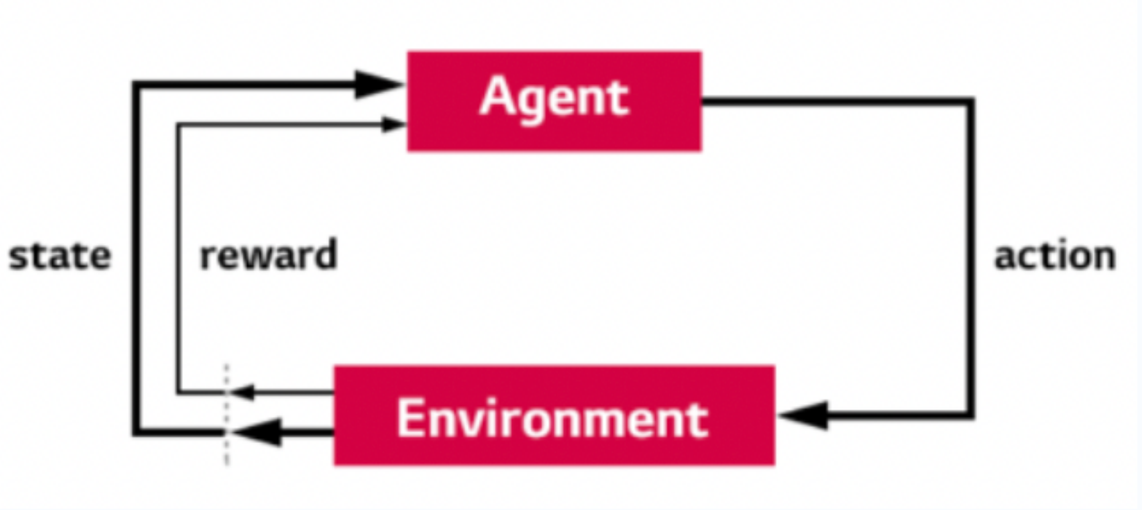
Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards.

INTRODUCTION

- Deep Q-learning : 강화학습 + 딥러닝
- 딥러닝을 적용하기 위해 해결해야 할 강화학습 데이터의 특징
  1. 희박하고 노이즈가 많으며 딜레이가 있는 신호로서 품질이 좋지 않은 데이터
  2. 높은 상관관계의 데이터
  3. agent가 새로운 action 취할 때마다 바뀌는 데이터의 분포

BACKGROUND



Markov Decision Process
(MDP)

- Agent는 action, state, reward의 시퀀스로 Environment와 상호작용 함
- Agent는 현재의 장면만을 관찰하기에 전체 상황 이해하기 힘듦

action의 sequence 관찰하여 학습해서
미래 보상을 극대화시키는 방식으로 action을 선택함 (MDP)

BACKGROUND

Reward

$$R_t = \sum_{t`=t}^{T} r^{t`-t} r_{t`}$$

Action Value Function

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma Q_i(s', a') \,\middle|\, s, a \right]$$

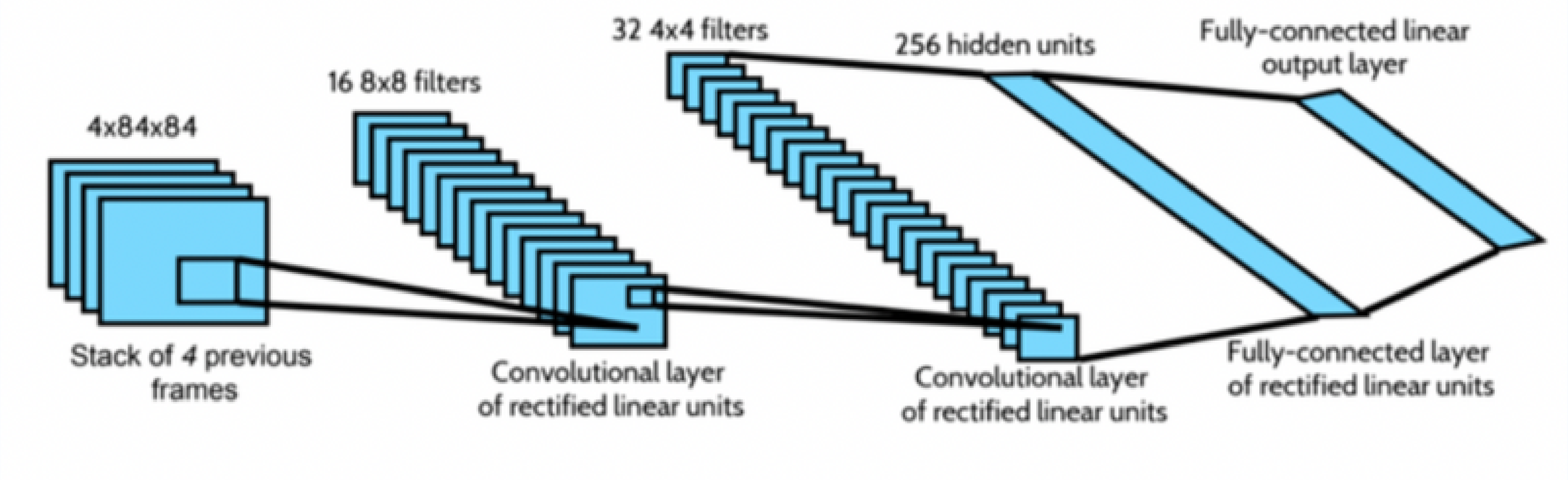
Loss Function

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_i - Q(s,a;\theta_i) \right)^2 \right], \text{where, } y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \,\middle|\, s, a \right]$$

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s,a;\theta_i) \right) \nabla_{\theta_i} Q(s,a;\theta_i) \right]$$

Deep Reinforcement Learning

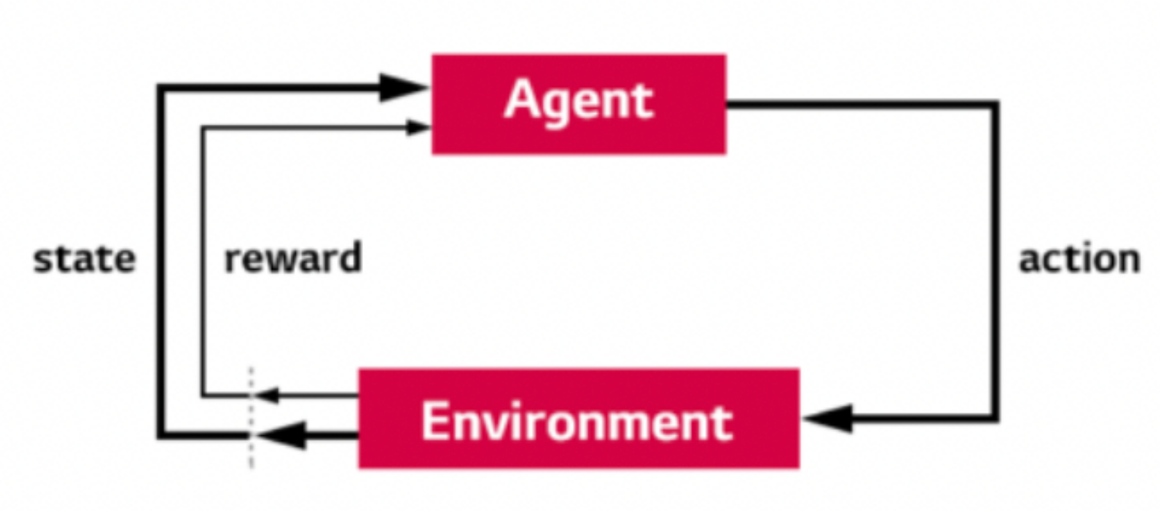
Deep Q-Network Architecture : CNN + RL

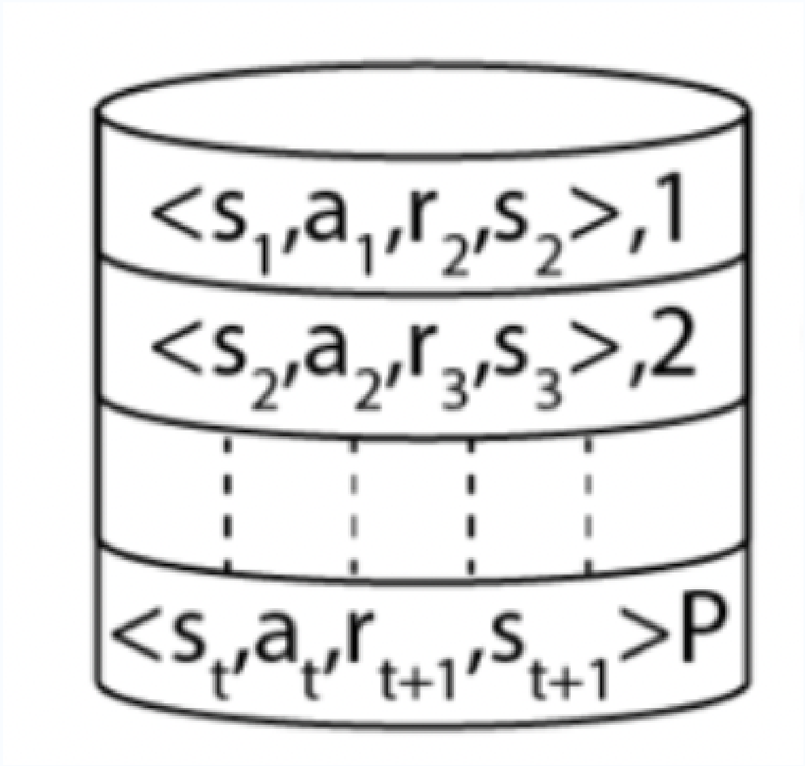


output은 input state에 대한 개별 action 예측한 Q-value

Deep Reinforcement Learning

Experience Replay





데이터의 높은 상관관계를 극복하고
학습을 안정적으로 만듦

Deep Reinforcement Learning

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

Experiment

게임 실험 환경 조건은 모두 동일하게 세팅

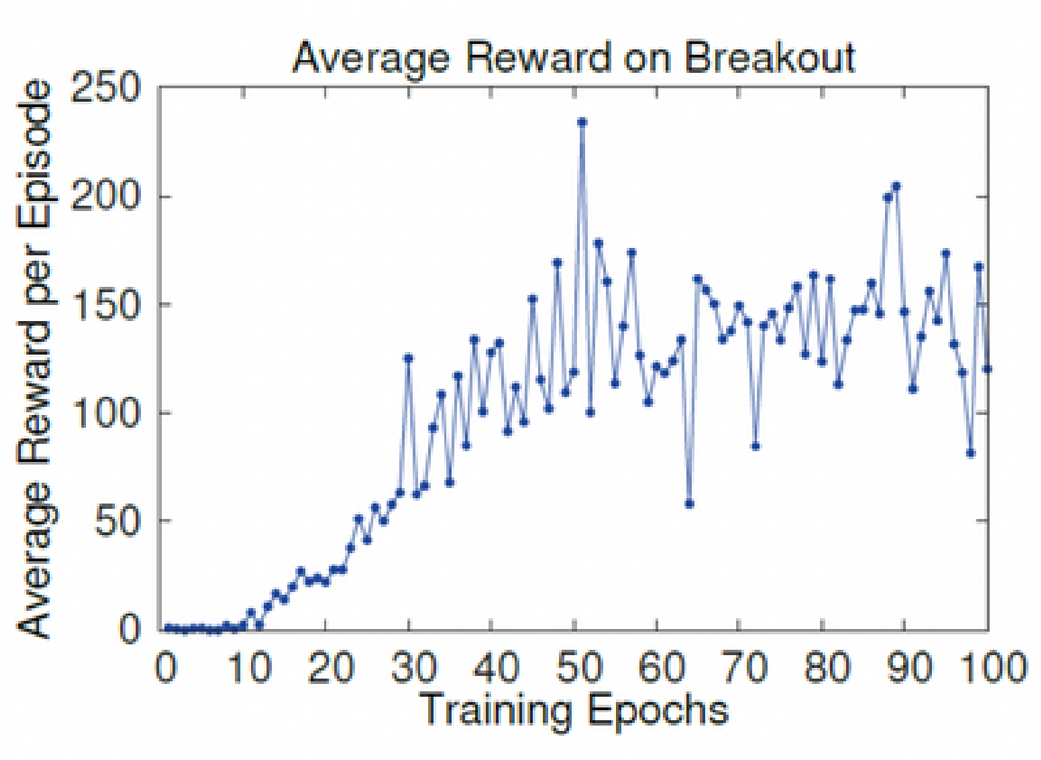
Result

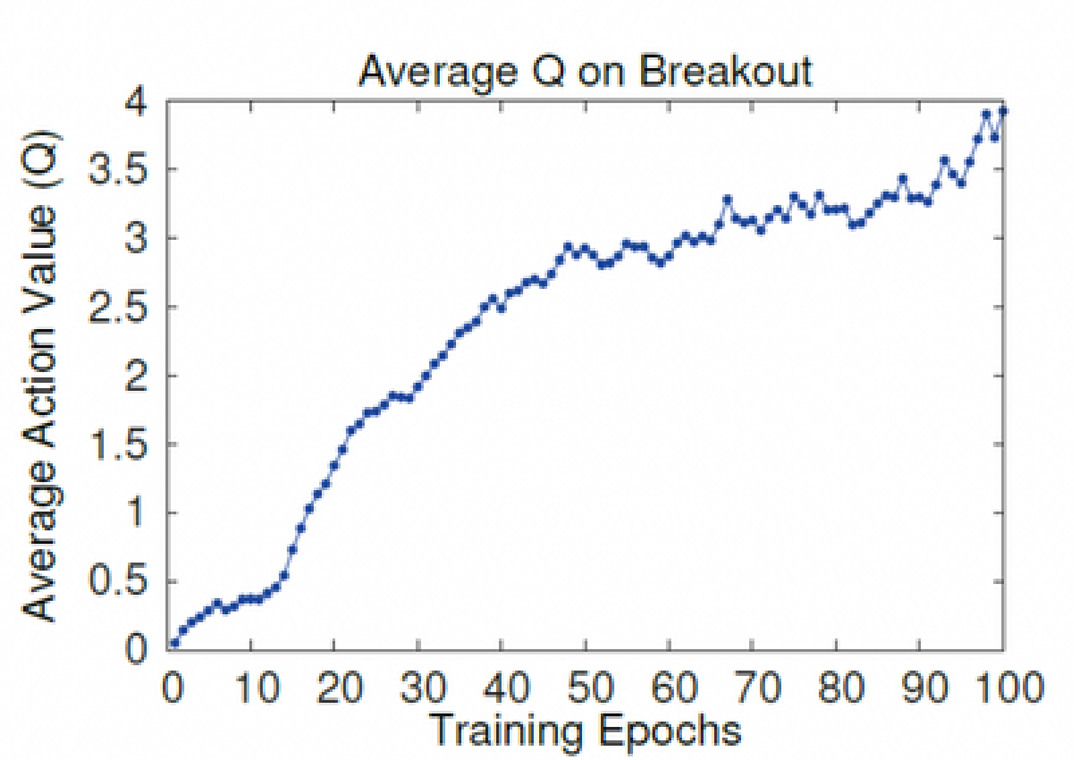
1. DQN이 우수한 성적을 받고 있음을 확인 할 수 있다

	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
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Result

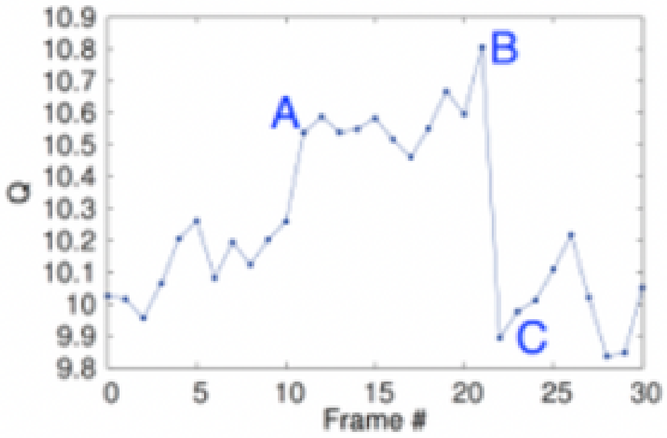
2. Q Value값을 통해 반복이 거듭될 수록 학습이 잘 되는 것을 알 수 있다

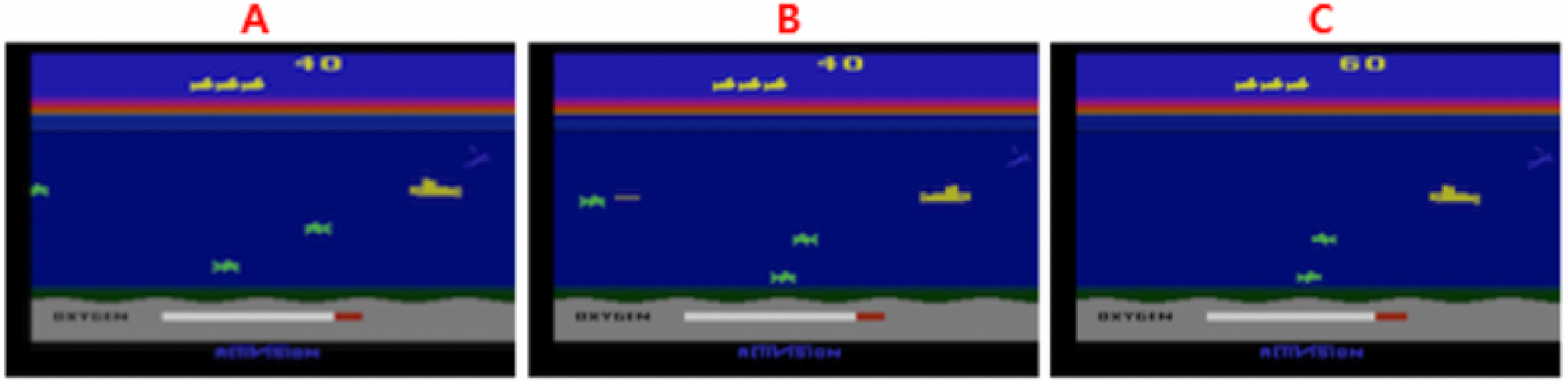




Result

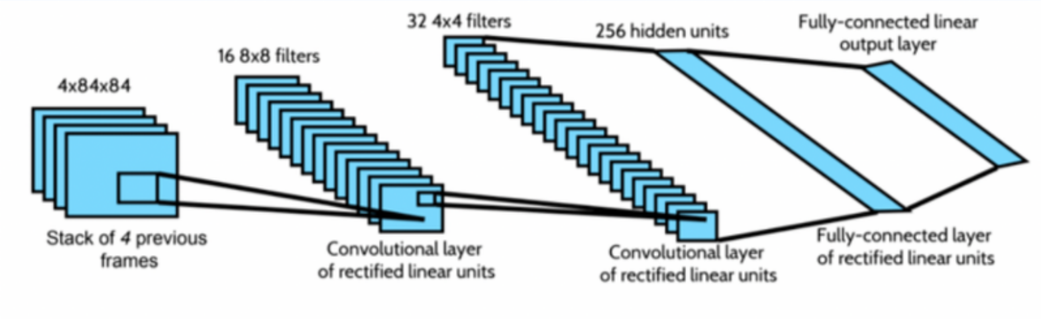
3. 게임 화면을 통해서도 Q가 잘 작동하고 있음을 확인할 수 있다





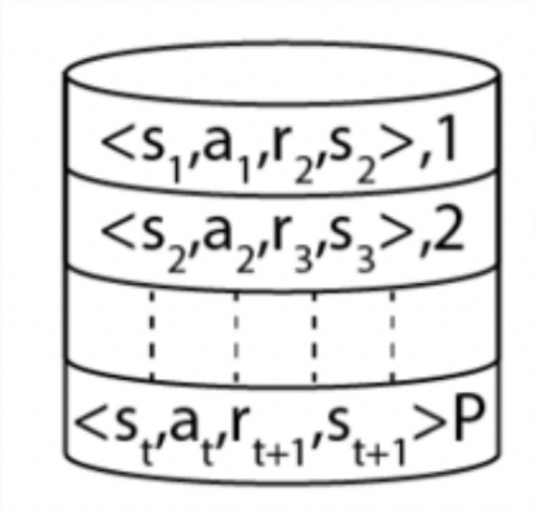
Conclusion

희박하고 노이즈가 많으며 딜레이가 있는 신호
품질이 좋지 않은 데이터



CNN기반인 DQN 알고리즘으로 학습

높은 상관관계의 데이터
불규칙한 데이터의 분포



EXPERIENCE REPLAY