Google Developer Group
Incheon National University

Playing Atari with Deep Reinforcement Learning

GDGoC INU AI Part Paper Seminar
김재호 윤시원

Contents

1. Introduction
2. Background
3. Deep Reinforcement Learning
4. Preprocessing and Model Architecture
5. Experiments
6. Conclusion

1. Introduction

1. Introduction

- 목표: 다양한 Atari game을 잘하는 강화학습 모델 제작
- Atari game 예시:

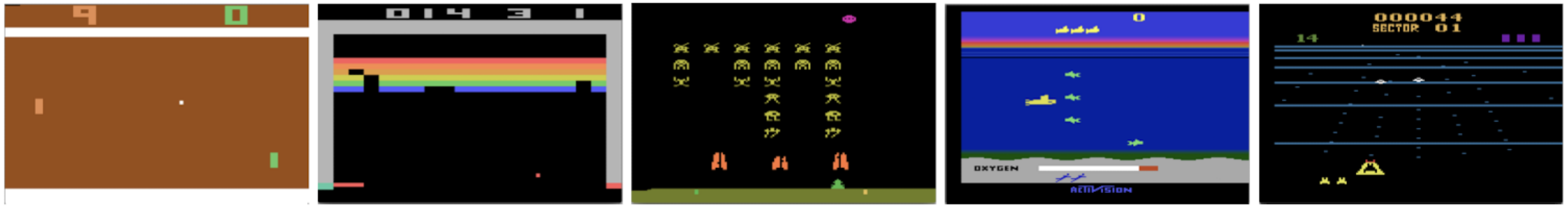


Figure 1. Atari games

https://youtu.be/V1eYniJ0Rnk?feature=shared&t=24

2. Background

2. Background

Reinforcement Learning Architecture

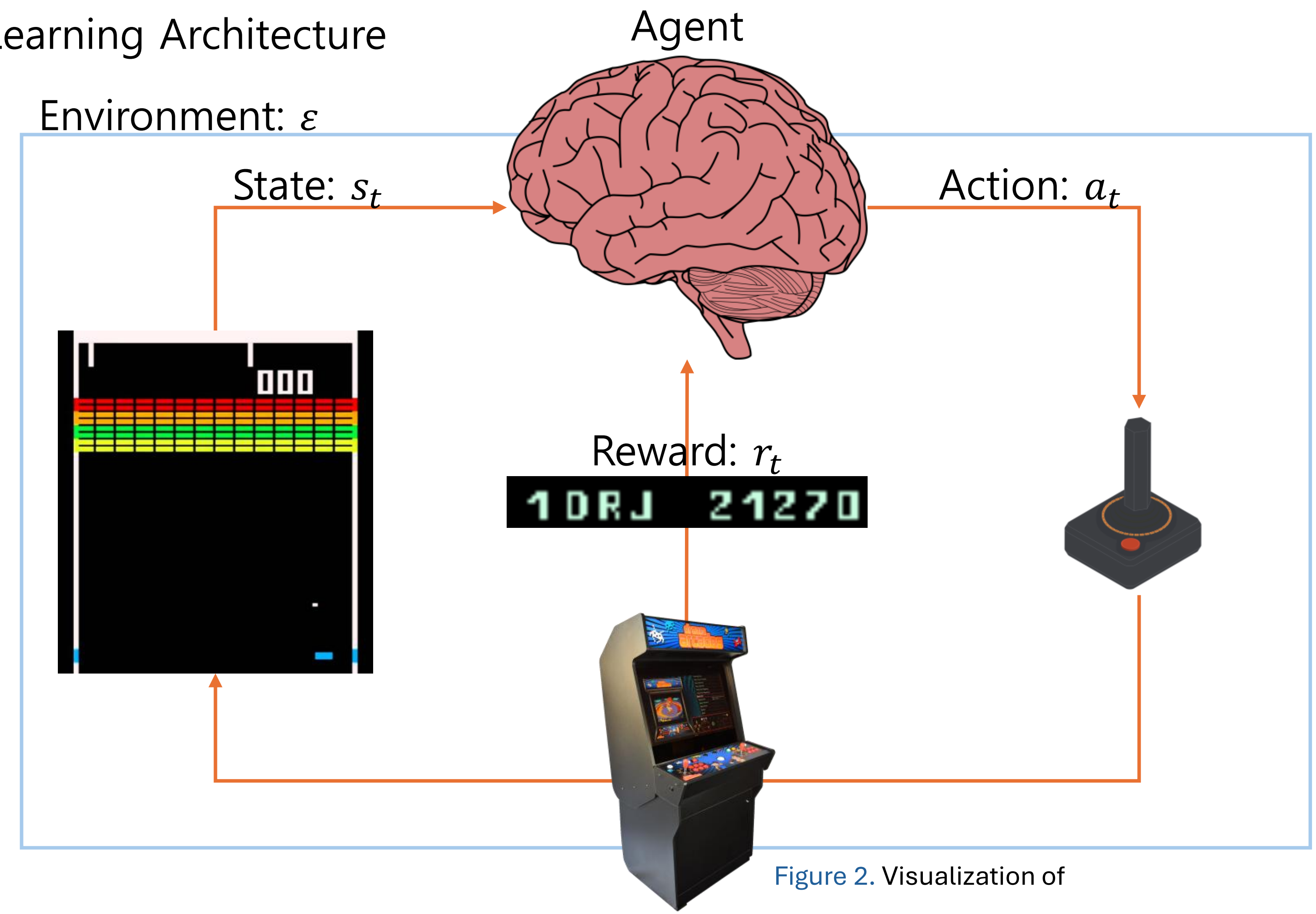


Figure 2. Visualization of

Image: $x_t \in \mathbb{R}^d$, a vector of raw pixel values representing the current screen

State: $s_t = x_1, a_1, x_2, \cdots, a_{t-1}, x_t$

2. Background

목표: return을 최대화하는 action하기

Return:

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$

T: game terminates

γ: time-step, $\gamma \in [0, 1]$ $\longrightarrow$ 1. 발산 방지 2. 빠르게 reward 받는 것을 선호

$\longrightarrow$ return 값은 미래 보상에 의존

$\longrightarrow$ return 값을 예측하는 optimal action-value function인 Q^* 사용(학습): Q-learning

Optimal action-value function(Q function):

$$Q^*(s, a) = \mathbb{E}_{s' \sim \varepsilon}[r + \gamma Q^*(s', a') \mid s, a]$$

$r + \gamma Q^*(s', a')$: R_t의 점화식 표현

변환된 목표: Q function 값을 최대화하는 action하기

2. Background

Q-learning: 총 {‘state 수’ × ‘action 수’}개의 Q 함수 필요

	→	←	↑	↓
Start	0	0	0	0
Idle	0	0	0	0
Hole	0	0	0	0
End	0	0	0	0

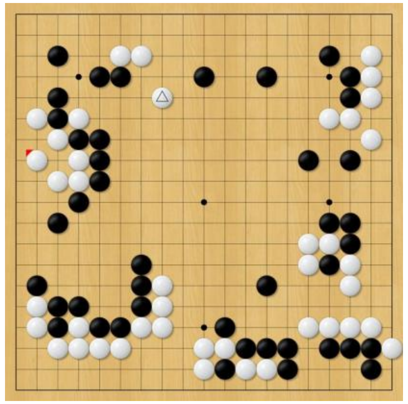


Figure 3. Q 함수 행렬 및 바둑

⟶ Q-Network

⟶ Network를 통해 Q 함수를 학습

$Q(s,a;\theta) \approx Q^*(s,a)$: Weights θ as a Q-Networks

⟶ $L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim\rho(\cdot)}[(y_i - Q(s,a;\theta_i))^2]$, each iteration i : 오차가 정규분포에 따라 모델링 가능하다고 가정

⟶ $\rho(s,a)$: behaviour distribution ⟶ 일반적으로 uniform distribution 사용

2. Background

ϵ-greedy Algorithm: 가끔 랜덤한 시도를 진행

$$a = \begin{cases} \max_a Q(s, a; \theta), & p \leq \epsilon \\ random\ action, & otherwise \end{cases}, \qquad random\ value\ p \in [0, 1]$$

3. Deep Reinforcement Learning

3. Deep Reinforcement Learning

Experience Replay: 메모리를 사용하여 성능 향상

Experience: $e_t = (s_t, a_t, r_t, s_{t+1})$

Replay Buffer: Data-set $\mathcal{D} = e_1, \cdots, e_N$: 이전 경험들을 저장

Deep Q-learning

State Representation: $\phi(s_t) = stacked[x_{t-3}, x_{t-2}, x_{t-1}, x_t]$

$\longrightarrow$ DQN의 입력 값으로 전처리 되었음 $\longrightarrow$ 최근 4개 상황의 프레임을 저장

$\phi(s_t) \longrightarrow$

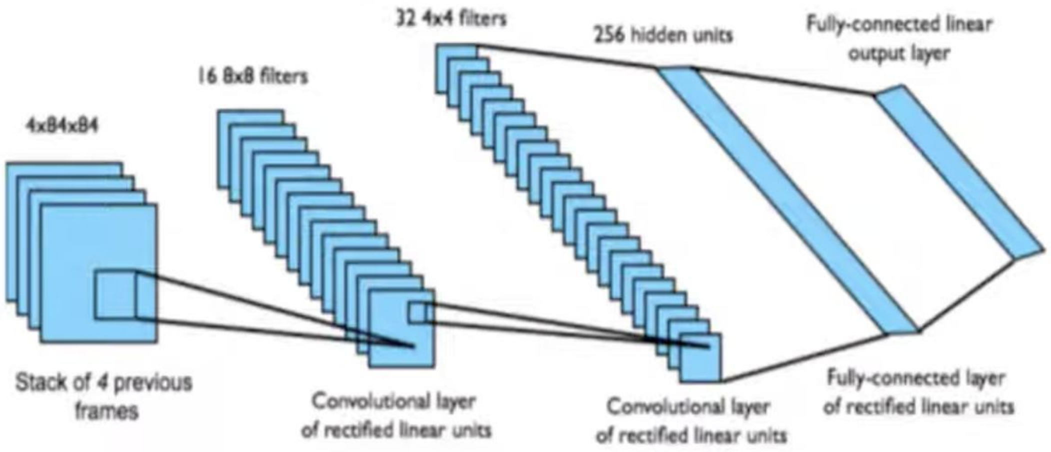


Figure 4. Network architecture

3. Deep Reinforcement Learning

Deep Q-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** How many times the agent plays the game
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$ $= stacked[x_1, x_1, x_1, x_1]$
  for $t = 1, T$ **do** Game start
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ ϵ –greedy algorithm: select action
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
1) Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
2) 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
  Learning
  end for Equation 3: $\nabla_{\theta_i} L_i(\theta_i)$
end for

1. State: $s_t = x_1, a_1, x_2, \cdots, a_{t-1}, x_t$
2. Transition $(\phi_t, a_t, r_t, \phi_{t+1})$: $e_t: (s_t, a_t, r_t, s_{t+1}) \rightarrow (\phi_t, a_t, r_t, \phi_{t+1})$, state representation

3. Deep Reinforcement Learning

DQN vs Q-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 $\phi_1 = \phi(s_1)$ → 데이터 재활용하여 효율적 학습 가능
  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 $\mathcal{D}$ → 랜덤으로 minibatch를 불러와서 근방의 큰 상관관계를 낮춤: 유사한 상황에서 반복된 action이 일어나는 것을 방지
    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 $\boxed{3}$
  end for
end for

4.1. Preprocessing

4.1. Preprocessing

210 x 160 pixels

4.1. Preprocessing

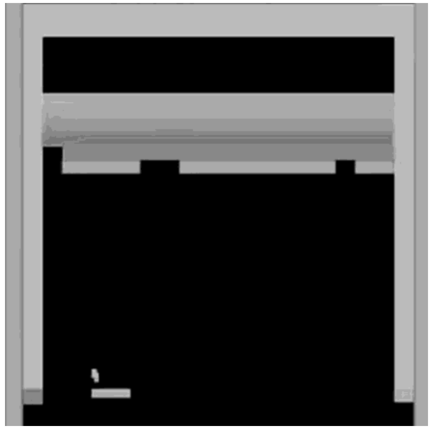
210 x 160 pixels
Gray Scale

4.1. Preprocessing

110 x 84 pixels

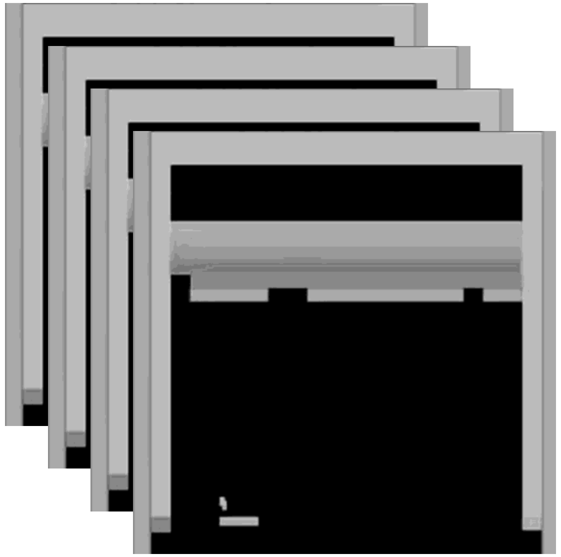
4.1. Preprocessing

84 x 84 pixels



4.1. Preprocessing

84 x 84 x 4



4.2. Model Architecture

4.2. Model Architecture

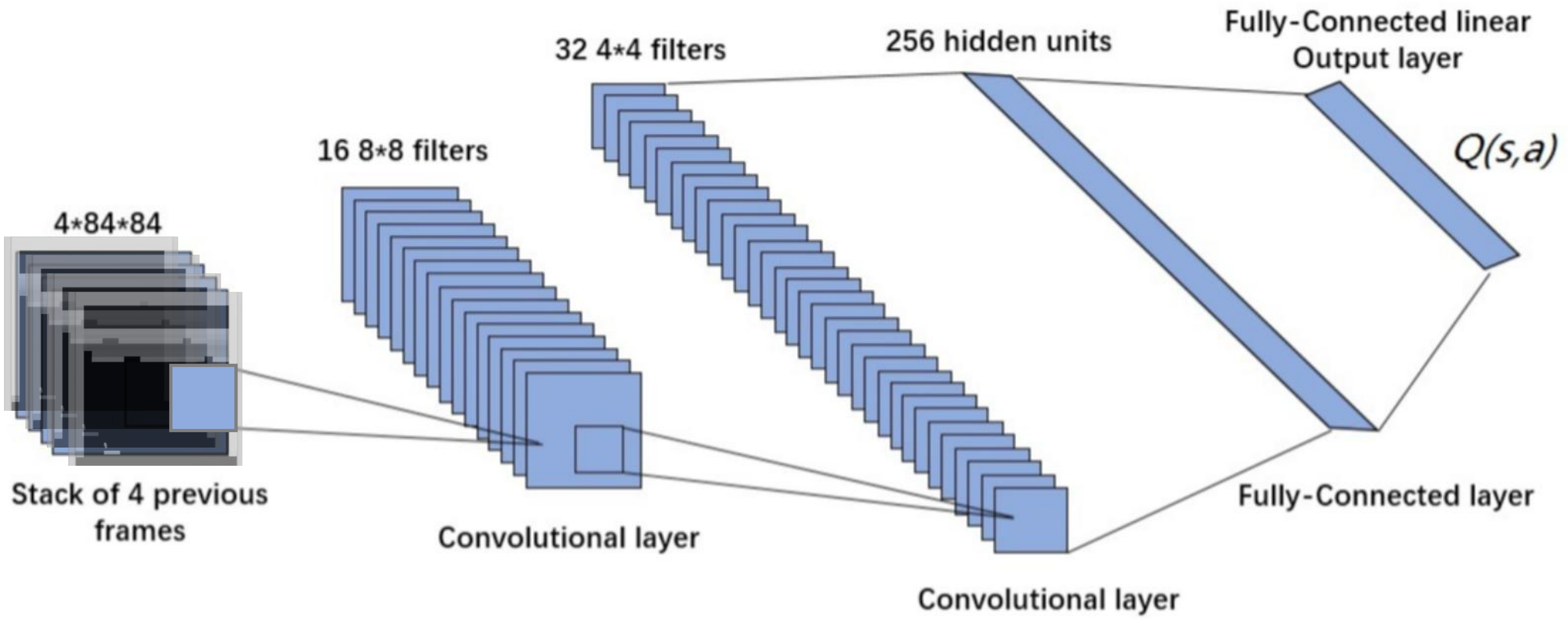
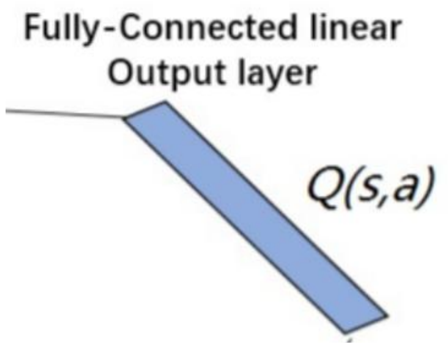


Figure 2.1: DQN structure

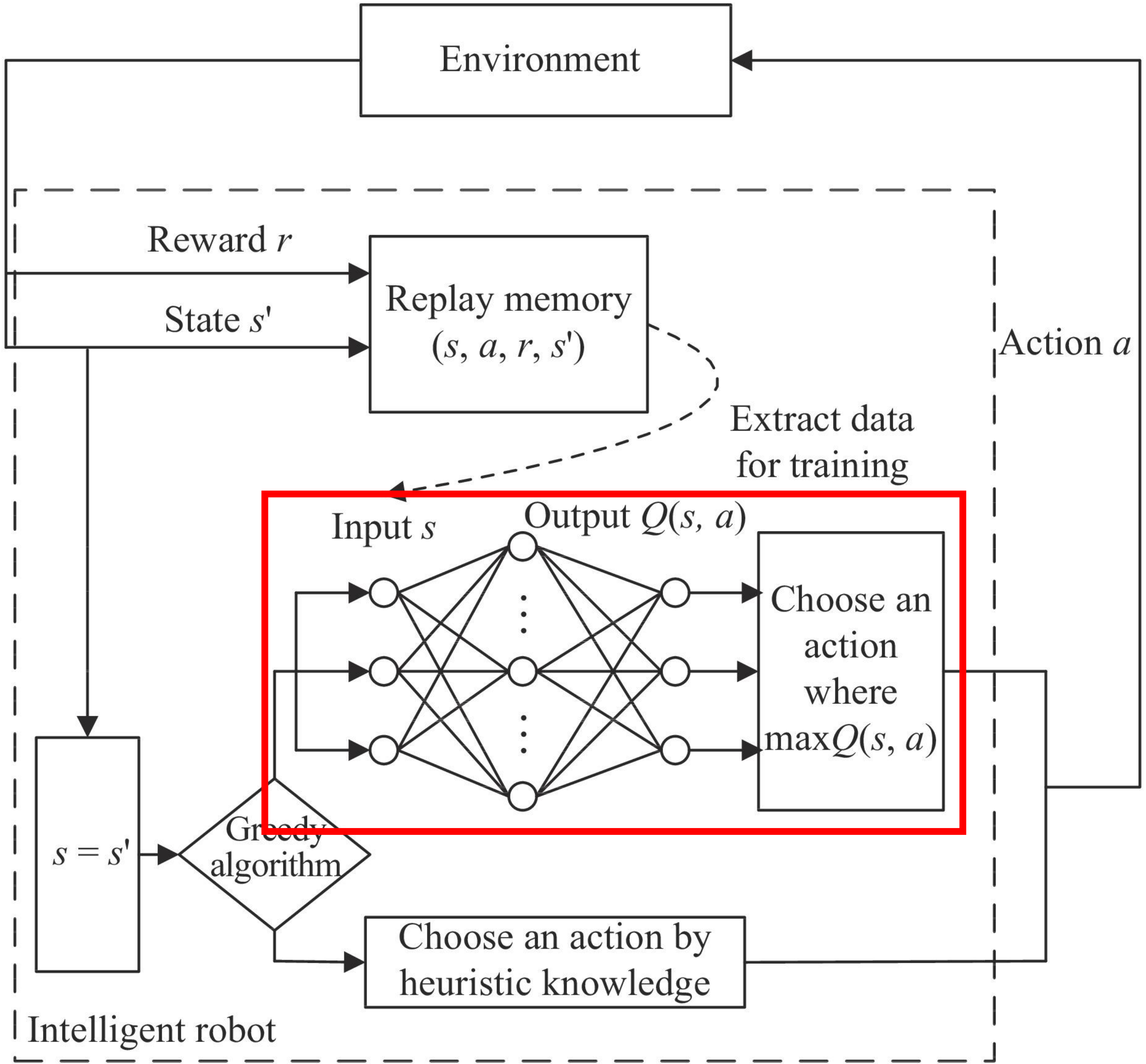
4.2. Model Architecture



e.g.

- a_1 : $move\ left$ — $Q(s, a_1) = 2.5$
- a_2 : $move\ right$ — $Q(s, a_2) = 3.8$
- a_3 : $fire$ — $Q(s, a_3) = 1.0$
- a_4 : $nothing$ — $Q(s, a_4) = 0.2$

4.2. Model Architecture

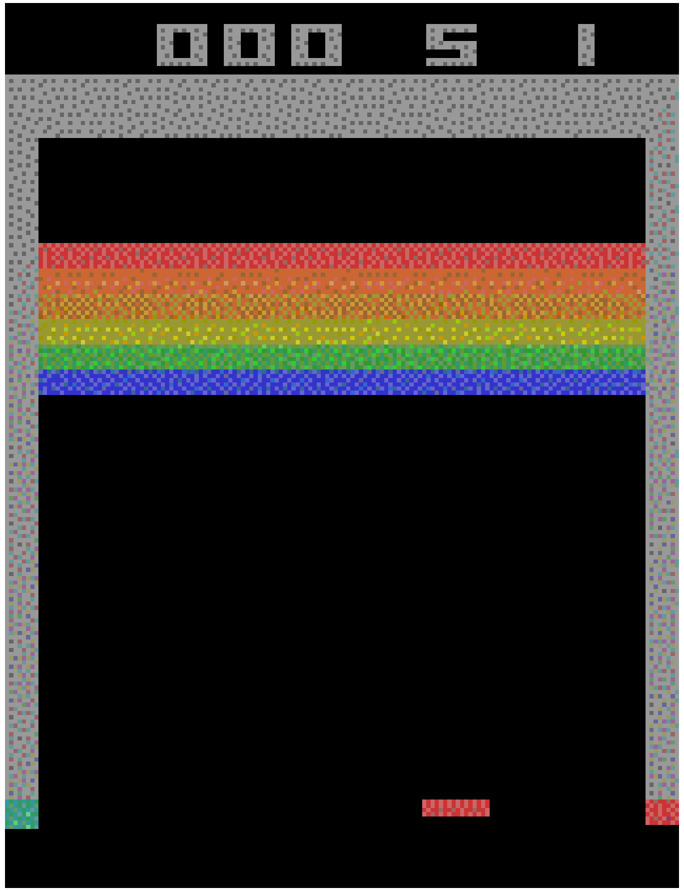


5. Experiments

평가 지표

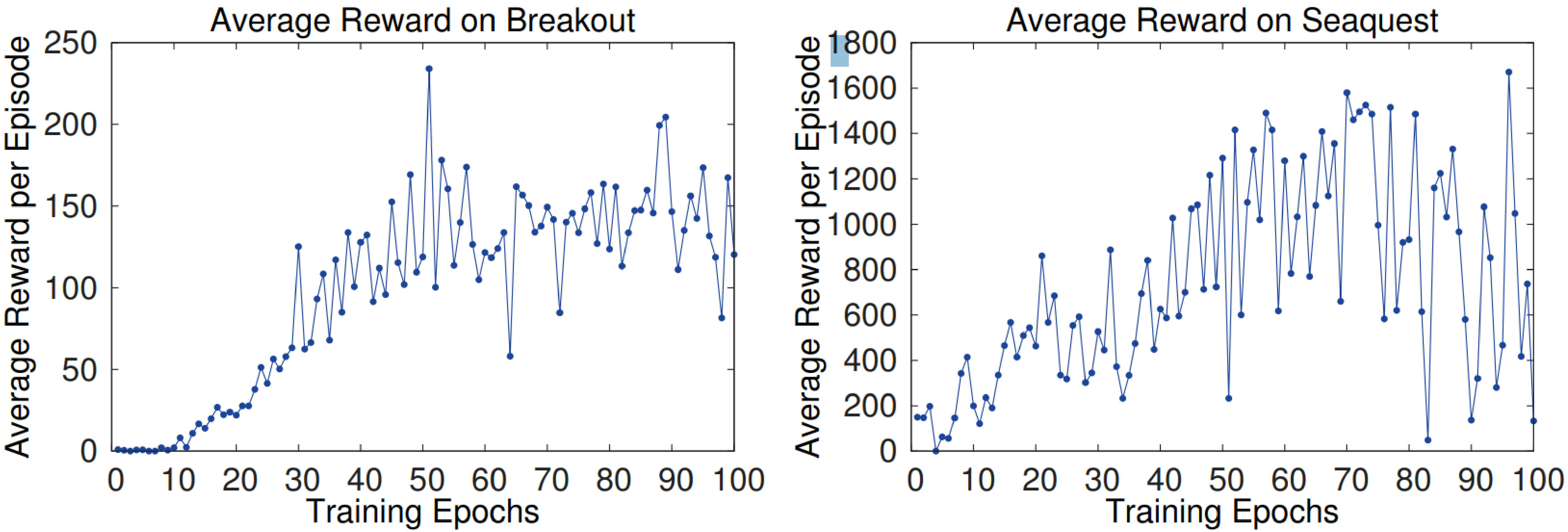
5. Experiments

Seaquest



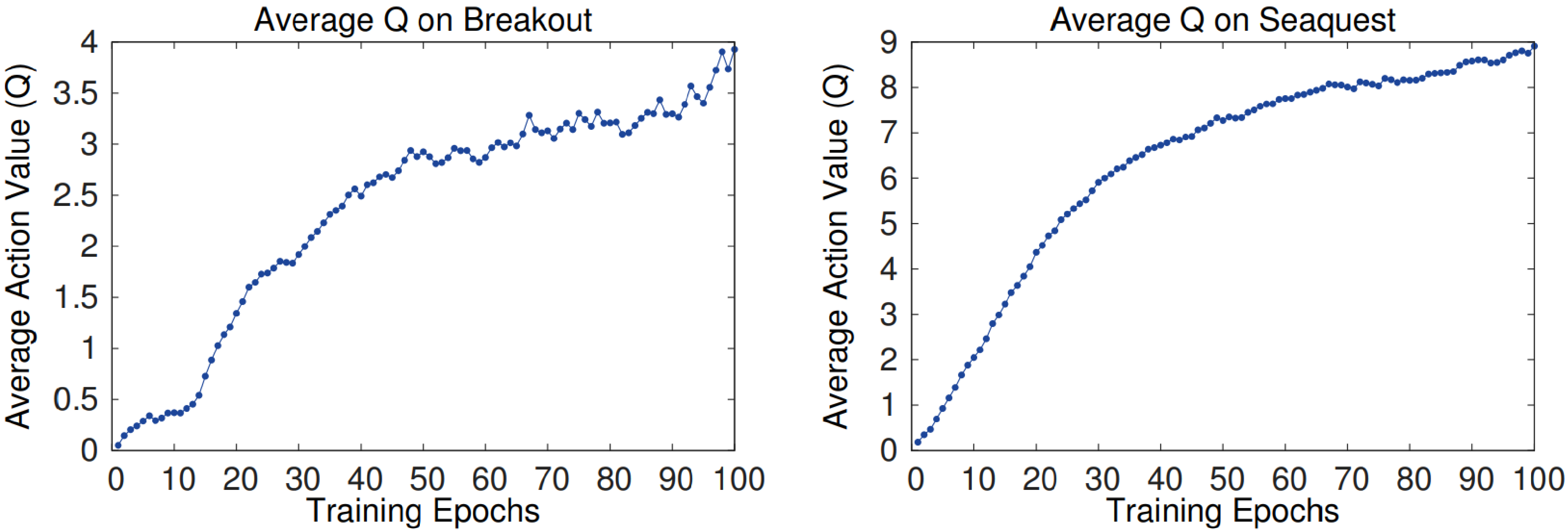
Breakout

5. Experiments



평균 보상 값

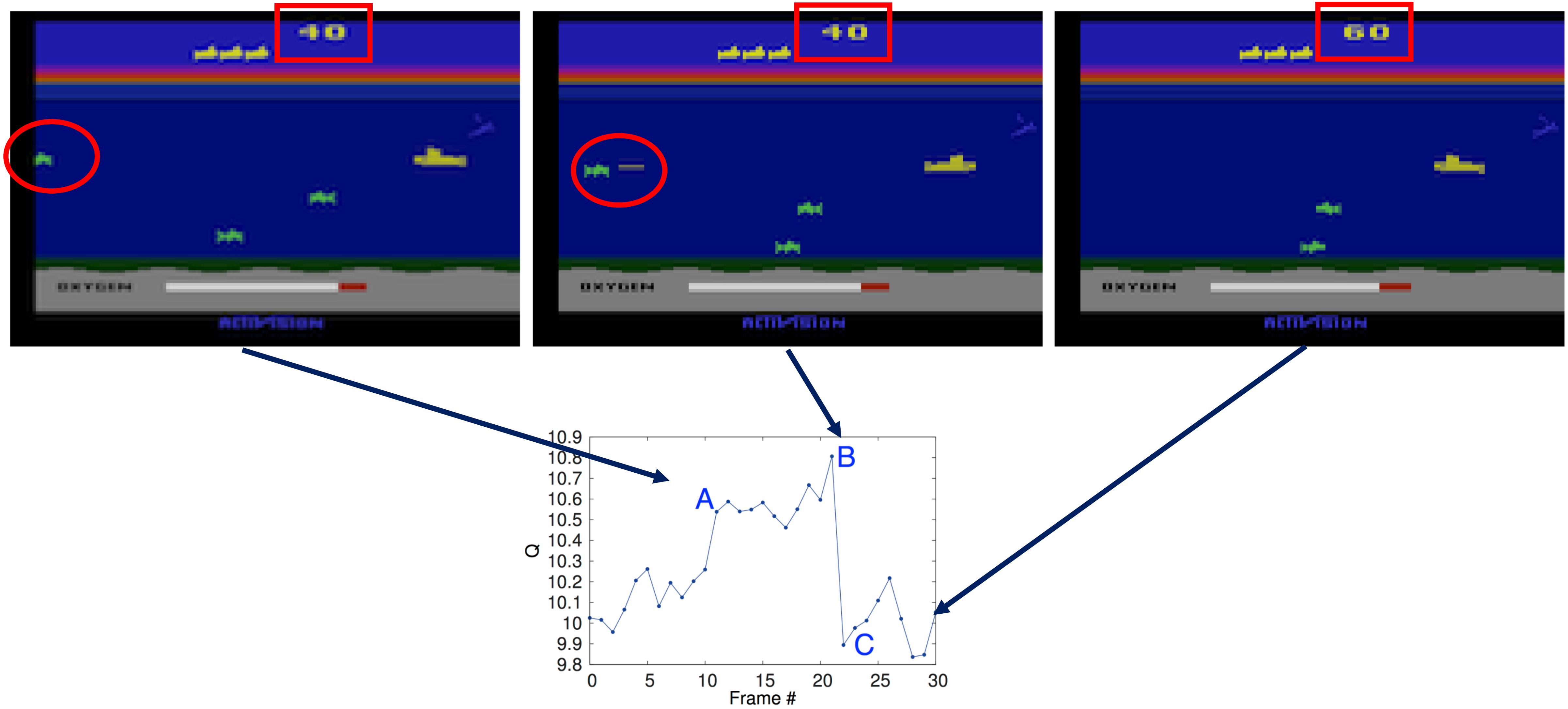
5. Experiments



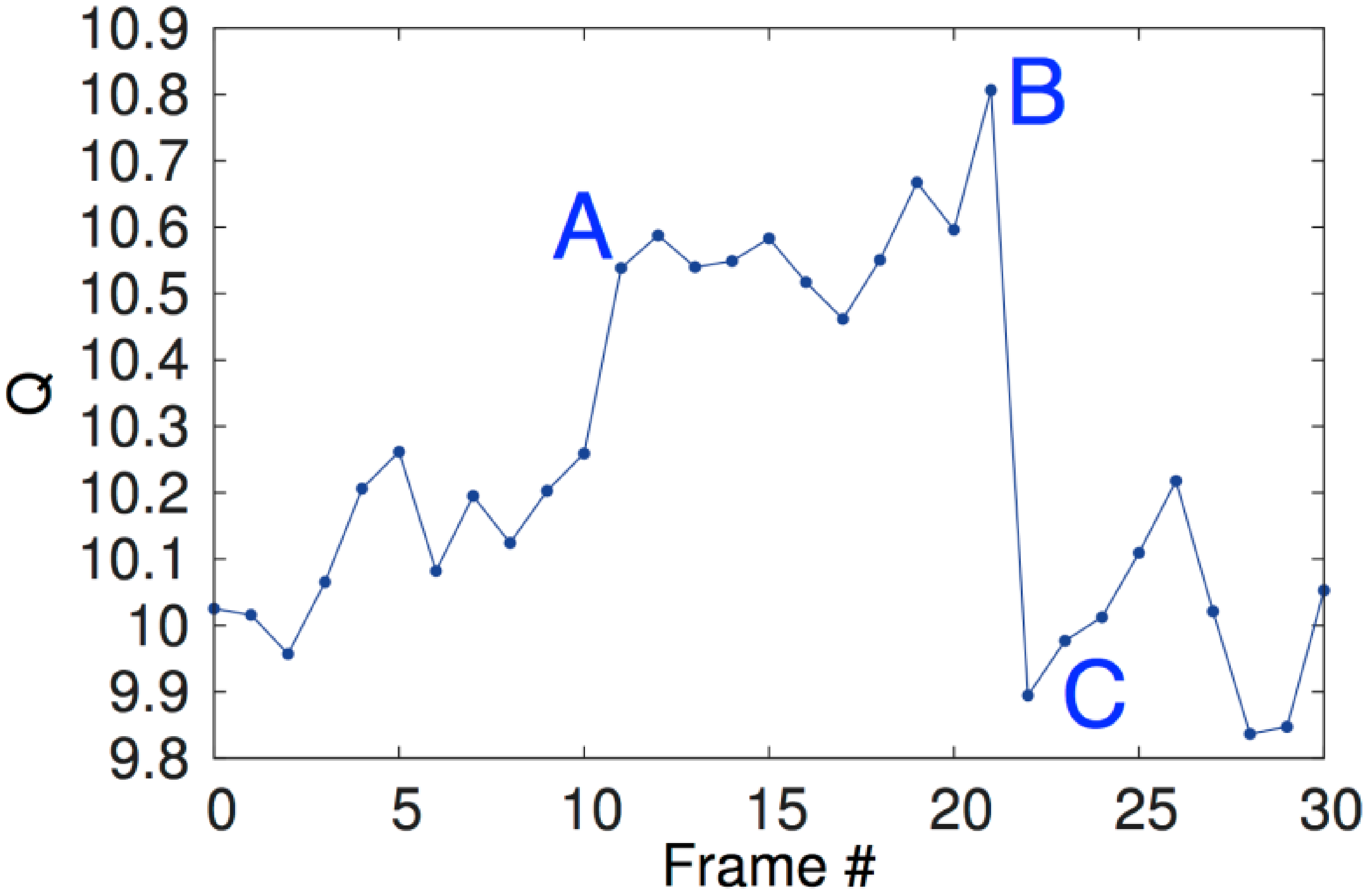
평균 maxQ 값

Q값의 시각화

5. Experiments



5. Experiments



5.1. Main Evaluation

5.1. Main Evaluation

	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

5.1. Main Evaluation

	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

6. Conclusion

- “강화학습을 위한 새로운 딥러닝 모델”
- RAW 픽셀만을 입력으로 사용하여 end-to-end 강화학습을 성공시킴.
- 별도의 하이퍼파라미터 조정없이 7개 게임 중 6개에서 기존 대비 최고성능을 기록
- Q-learning의 변형 알고리즘 제안

Q & A