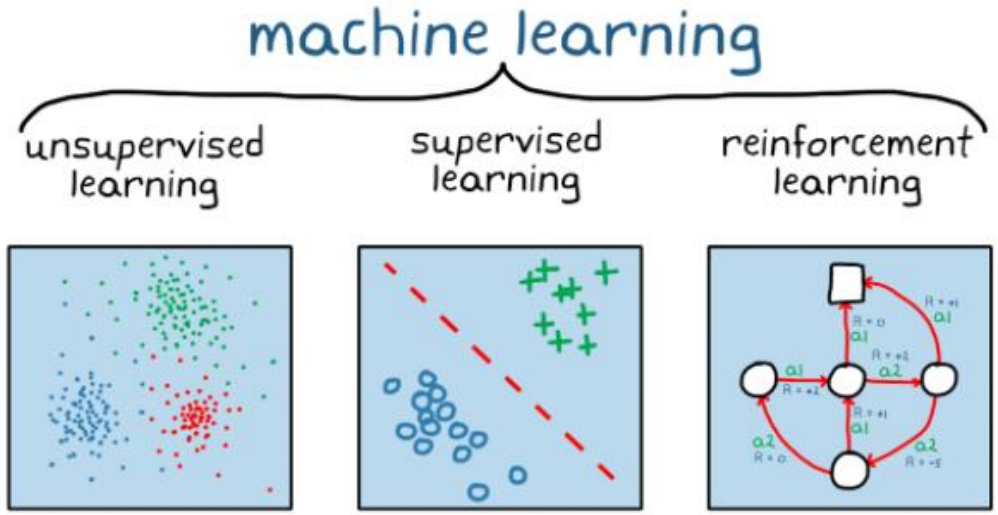
Playing Atari with Deep Reinforcement Learning

Seoyoung Lee
2021.09.23

01. Introduction

Reinforcement Learning



- high-dimensional sensory input(vision, speech)으로부터 agent를 제어하는 것을 학습하는 것은 RL의 오랜 목표.
- 딥러닝의 발전에 따라 Vision, Speech와 같은 고차원의 데이터들을 추출하는 것이 가능해짐. 이에 따라 딥러닝을 강화학습에 적용시키고자 함.

01. Introduction

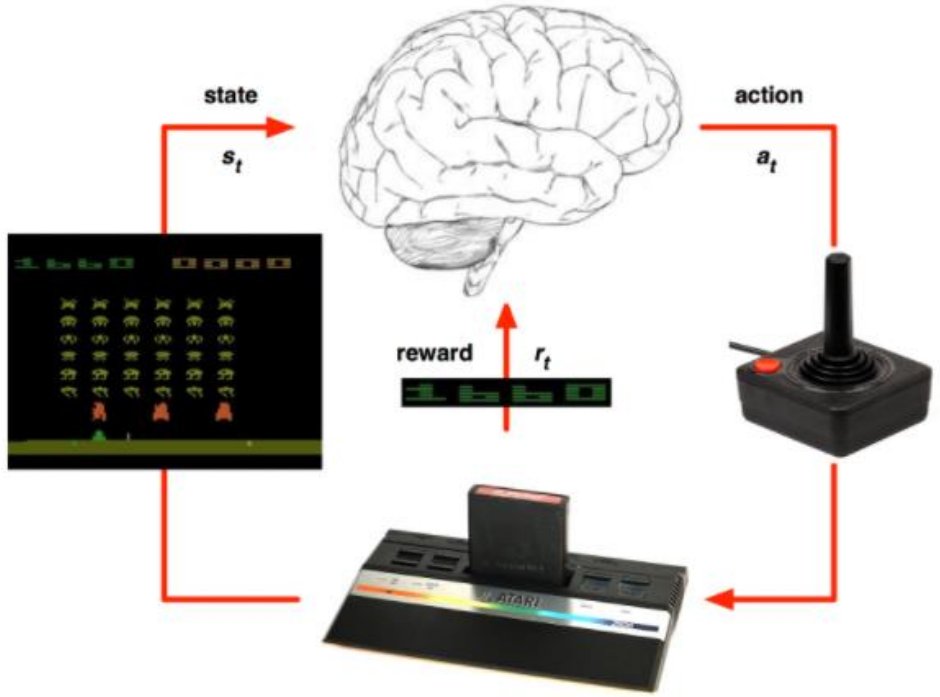
Reinforcement Learning

- 딥러닝을 강화학습에 적용하는 과정에서 문제점 발생
    1. 딥러닝 학습에는 대량의 레이블된 데이터가 필요
    2. 딥러닝 알고리즘의 데이터는 독립적이나, 강화학습은 데이터의 상호연관성이 높음

- CNN, Q-learning 알고리즘과 experience replay 기법을 적용하여 raw video data로부터 policy를 control하는 법을 학습할 수 있음을 증명하고자 함

02. Background

Environment



- 본 논문에서 RL의 환경(ε)은 Atari 에뮬레이터
- 환경의 구성요소는 action의 sequence, observation하는 화면, reward(점수)
- agent는 각 time-step마다 가능한 action중 한가지(a_t)를 선택
- agent는 raw pixel(x_t)과 reward(r_t)만 전달받음

02. Background

MDP

- agent가 현재 화면만으로 모든 상황을 이해하는 것은 불가능
- action과 observation의 시퀀스를 관찰하고 학습을 진행

$$s_t = x_1,\ a_1,\ x_2,\ a_2,\ \dots,\ a_{t-1},\ x_t$$

- 모든 시퀀스가 time-step에 의존하므로 MDP가 유도되고 이를 강화학습에 적용 가능

02. Background

Reward

- agent의 목표는 미래의 reward를 극대화하는 action을 선택하는 것

- 일반적으로 시간이 지날수록 reward의 가치는 감소하므로 이를 반영하기 위해 discount factor γ가 정의됨

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

02. Background

Q-function (action-value function)

- Optimal Action-Value Function을 다음과 같이 표기 : Q* (s, a)
- Q*(s, a)는 해당 sequence s에 action a를 취했을 때 최고의 보상을 받을 수 있는 함수
- π 는 s_t에서 a_t를 맵핑해주는 policy function

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right]$$

02. Background

Q-function (action-value function)

- Optimal Action-Value Function은 벨만 방정식을 따름

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') \middle| s,a\right] \quad (1)$$

- Bellman 방정식을 활용한 action-value function의 update

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a') \middle| s,a\right]. \quad (2)$$

- 근사 함수

$$Q(s,a;\theta) \simeq Q^*(s,a). \quad (3)$$

02. Background

Q-Network

- 앞에서 정의한 Q-function을 모델링한 네트워크: Q-Network

$$Q(s,a;\theta) \simeq Q^*(s,a). \quad \textbf{(1)}$$

- Q-Network는 각 iteration마다 바뀌는 손실함수 $L_i(\theta_i)$의 최솟값을 찾아서 학습을 진행.

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

02. Background

Q-Network

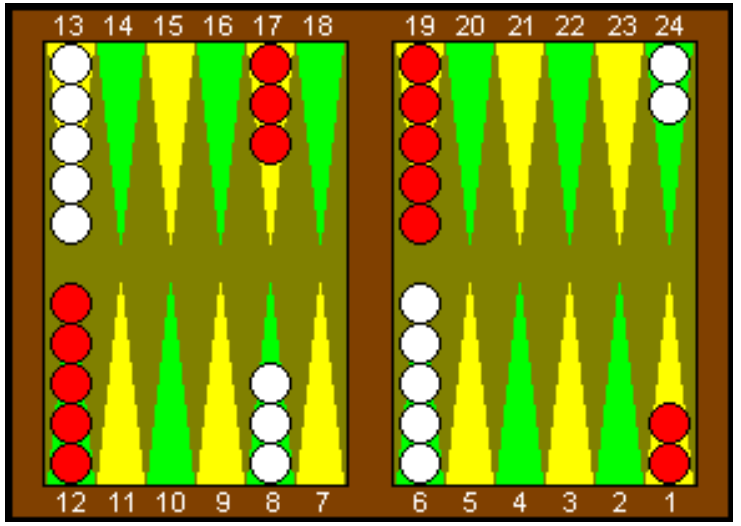
- Q-Network Loss function의 gradient는 다음과 같음

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

03. Related Work

TD-Gammon

- 강화학습에서 가장 성능이 좋았던 기존 알고리즘은 TD-Gammon
- TD-Gammon은 Q-Learning과 비슷한 model-free구조
- 다만, Backgammon 게임에서만 높은 성능을 보이고, 다른 게임에서는 성능이 좋지 않음



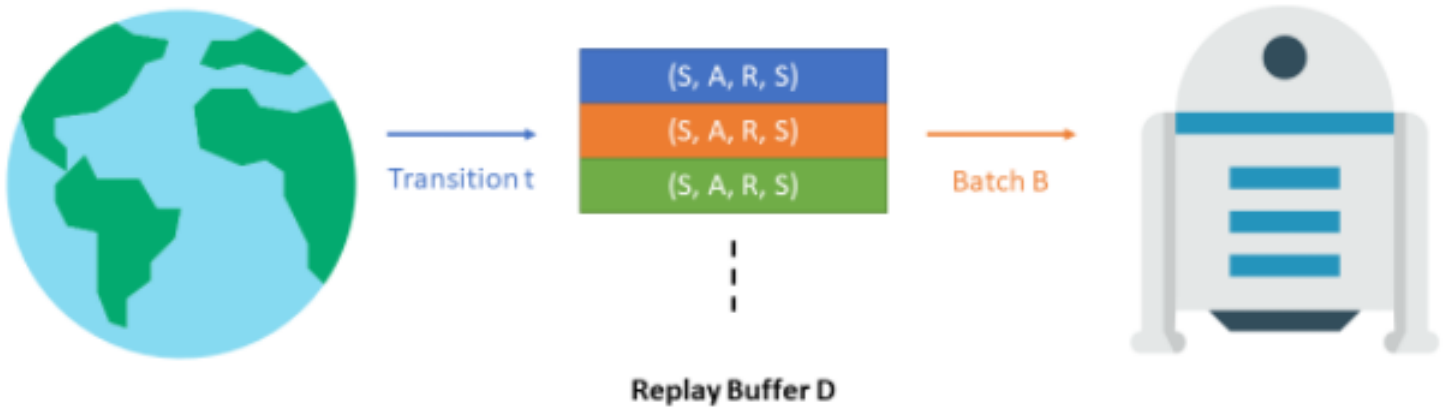
03. Related Work

NFQ

- 본 논문과 가장 유사한 접근법을 가진 이전 연구로는 Neural fitted Q-Learning(NFQ)가 존재
- NFQ에서는 Q-Network의 parameter 갱신을 위해 Rprop알고리즘 및 Batch Gradient Descent를 사용
- NFQ에서는 deep autoencoder를 사용하여 각 task로부터 low dimensional representation을 학습

04. Deep Reinforcement Learning

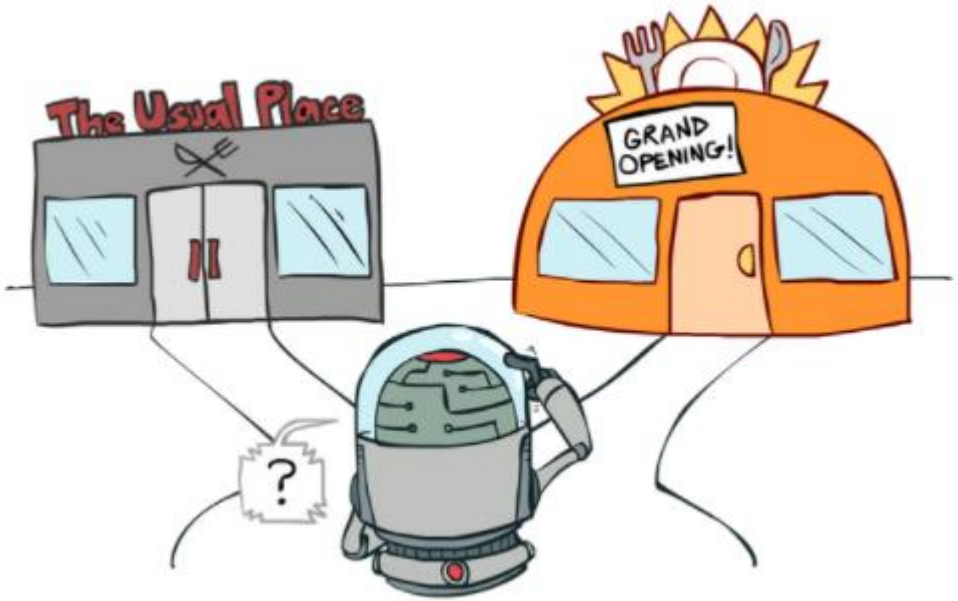
Experience Replay



- agent가 각 time-step마다 했던 Experience(e_t)을 Dataset(D)에 저장 $e_t = (s_t,\ a_t,\ r_t,\ s_{t+1})$
- Dataset에 저장된 풀에서 일부를 균등한 확률로 샘플링해 학습에 적용

04. Deep Reinforcement Learning

ϵ-greedy policy



- Experience replay가 끝난 후 agent는 ϵ-greedy policy를 사용해 action을 선택.
- ϵ-greedy policy는 ϵ의 확률로 random한 선택을 하거나 1- ϵ의 확률로 최선의 선택을 하는 알고리즘

04. Deep Reinforcement Learning

DQN의 장단점

- 각 time-step의 Experience가 잠재적으로 많은 가중치 업데이트에 재사용
- mini-batch를 만드는 sampling 과정을 통해 데이터들 간의 high correlation을 효율적으로 관리
- 기존의 on-policy는 현재의 parameter가 다음 데이터 샘플을 결정하므로 parameter가 극솟값에 빠질 수 있으나, experience replay를 사용하면, behavior distribution이 균형을 이룸

- 메모리 용량으로 인해 모든 Experience를 Dataset에 저장 불가
- 모든 Experience를 균등하게 sampling하므로 이를 보완할 최적화된 sampling 방법 필요

04. Deep Reinforcement Learning

Algorithm

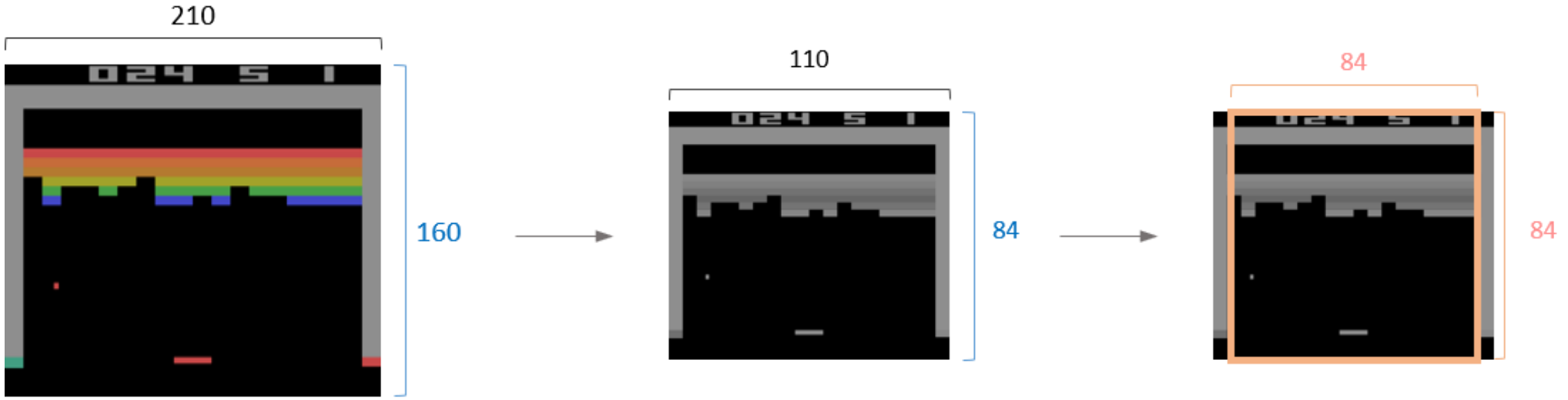
Algorithm 1 Deep Q-learning with Experience Replay

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

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

04. Deep Reinforcement Learning

Preprocessing



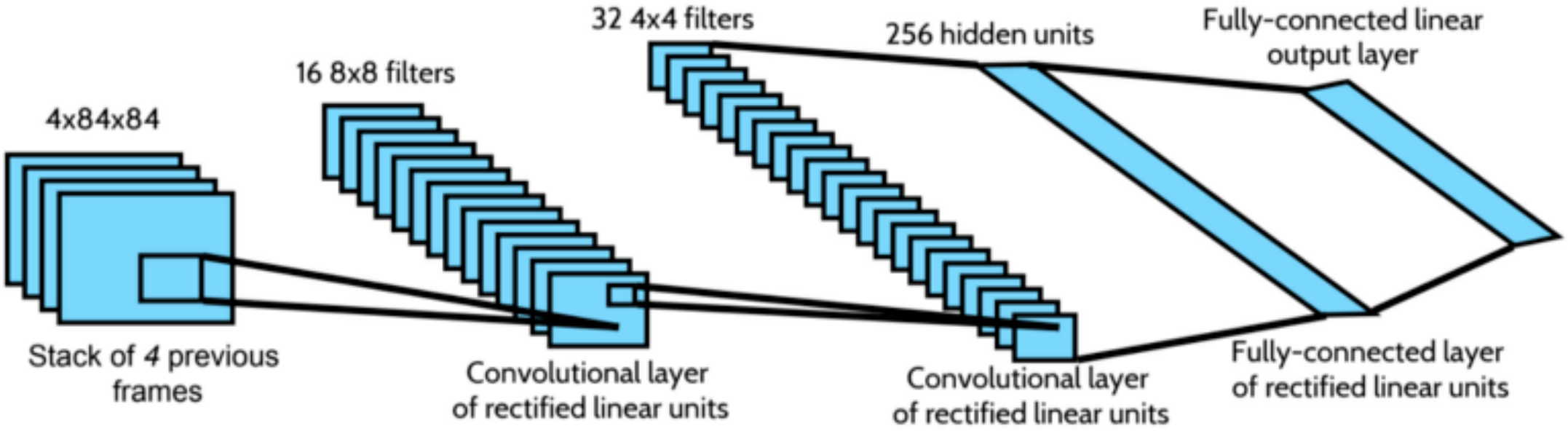
128 color,
210 X 160 pixels

gray scale,
110 X 84 pixels

cropping
84 X 84 pixels

04. Deep Reinforcement Learning

Network Architecture



05. Experiments

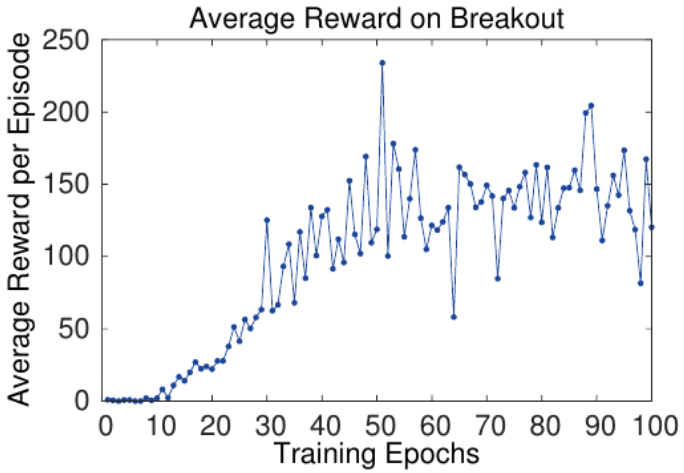
Environment & Settings

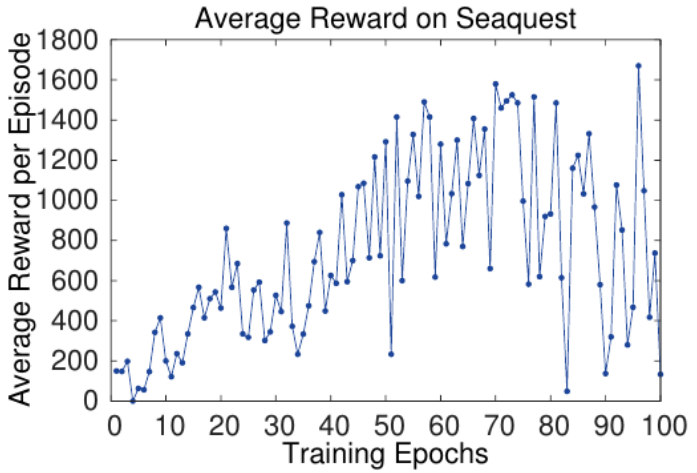
- 7개의 ATARI 게임에 대해 성능을 평가
- 모든 게임에 대해 동일한 조건으로 실험 진행

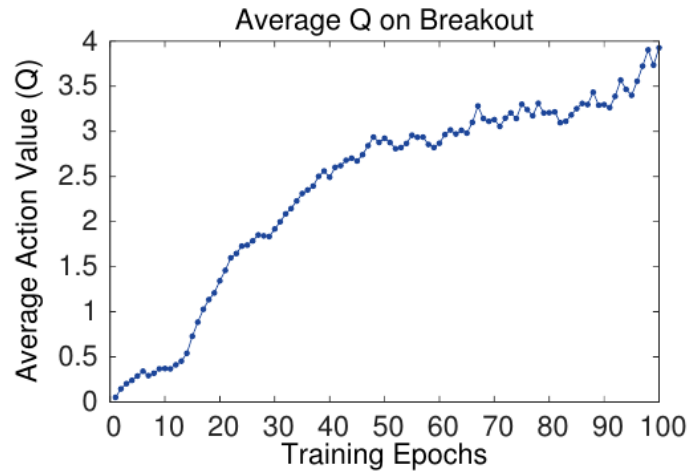
- 최적화 알고리즘으로 RMSProp사용 (minibatch size : 32)
- 모든 reward값은 -1, 0 ,+1로 제한하여 error Scale 제한
- agent에서 모든 프레임을 보고 action을 취하는 것이 아닌 k번째 프레임을 보고 action 선택

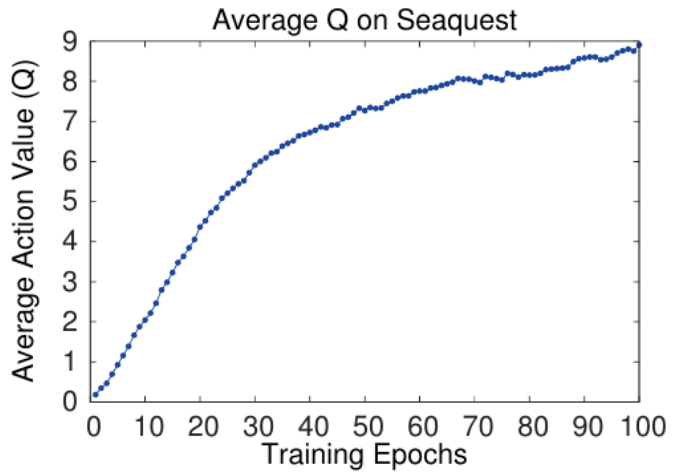
05. Experiments

Training and Stability



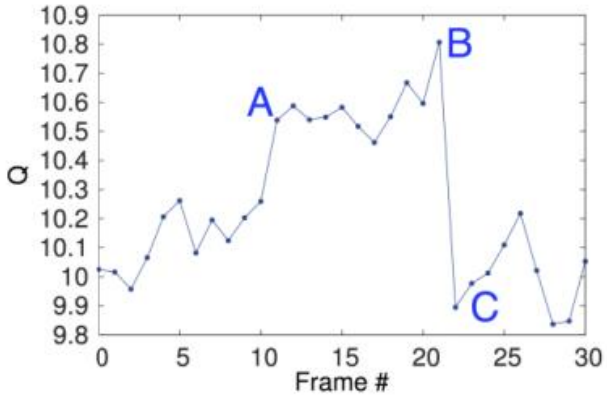






05. Experiments

Visualizing the Value Function



A

B

C

05. Experiments

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

06. Conclusion

- 강화 학습을 위한 새로운 딥러닝 모델을 소개
- Raw pixel을 입력으로 사용하며 ATATRI 2600의 control policy를 학습
- experience replay memory로 mini-batch 방식을 사용하는 변형된 online Q-learning을 제안함
- Architecture와 Hyper-parameter의 변화 없이 7개중 6개의 게임에서 높은 성능을 발휘

Thank you