Prioritized Experience Replay

Google Deep Mind, Tom Schaul et al, 2016 ICLR

- 강동구 -

목차

Introduction & Background

기존에 DQN에서 experience replay를 통해 transition을 uniform하게 뽑아서 correlation을 해결하였다.

model-based-planning 영역에서는 value iteration에서 state의 value의 차이를 이용한 우선순위화와 TD-error를 이용한 우선순위화도 있었다.

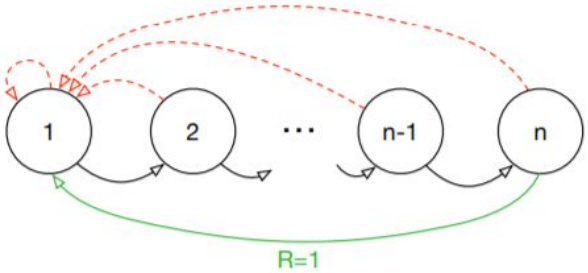
TD-error를 다양한 방도로 사용했었다. ex. 언제,어디로 explore할지, 어떤 feature를 선택해야 할지..

이 논문은 model-based -----> model-free

Main idea : transition이라고 다 같은 transition이 아니다!! |TD-error|가 클수록, 배울점이 많은 놈이다!

Introduction & Background

실험환경 : Blind Cliffwalk : n번째까지 우직하게 가야 보상얻음.



exploration이 얼마나 어려운지. 대부분 transition의 보상이 0.

즉, 진짜 어쩌다 보상받은 transition을 잘 살려내야하는 문제로 적합.

어떤 transition을 저장할 것이냐(X)

어떤 transition을 replay할 것이냐(O)

어떻게 우선순위화 할건데

먼저, transition들로부터 agent가 얼마나 배울 수 있는 '양'을 측정해야함.

이를 직접적으로 나타내는 수치가 없기에, 논리적으로 비슷한 TD-error δ를 이용.

TD-error의 크기가 agent입장에서는 How surprising or How unexpected를 나타냄.

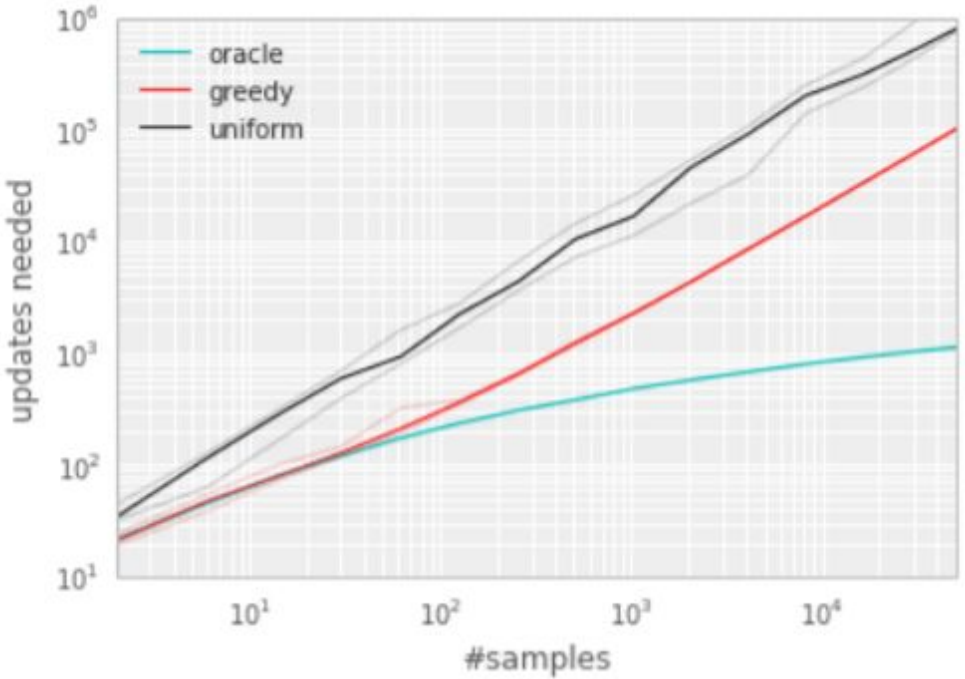
TD-error란…

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Big)}^{\text{temporal difference}}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

즉, 각 transition마다 |δ| 를 추가해 replay buffer에 쌓아 놓고, 제일 큰 |δ|를 가진 transition을 뽑아서 replay한다. ⇒ greedy TD-error prioritization

하면 뭐가 좋은데



X축은 replay buffer의 사이즈

Y축은 True value를 찾기위한 평균 update 수.

greedy는 단순 |δ|가 큰 transition으로 replay함

oracle은 global loss를 최소화하는 transition을 replay함.

목적 : greedy 이놈을 oracle과 닮아가도록.

greedy TD-error prioritization의 문제

1. 전체 transition의 TD-error를 훑어서 max를 찾아야하는 sweep 계산비용 문제
2. transition의 TD-error가 업데이트 될때는 오직 해당 transition이 replay된 직후

이것들로 야기되는 구체적 문제

1. 처음부터 낮은 TD-error를 가진 transition은 다음에 replay되기까지 오래걸림
2. noise spike(stochastic reward)에 민감함. bootstrap을 통해 더욱 악화
3. 경험의 일부에서 greedy를 하는 것이기에, error가 천천히 줄어들음. 그래서 초기에 높은 TD-error를 가진 transition이 많이 뽑혀서, diversity의 부족해져서 over-fitting 가능성 있음. function approximator를 통해 더욱 악화

⇒ Stochastic sampling을써서, uniform과 greedy의 중간에 끼어들겠다.

Stochastic sampling

트랜지션 i가 뽑힐 확률 P(i)

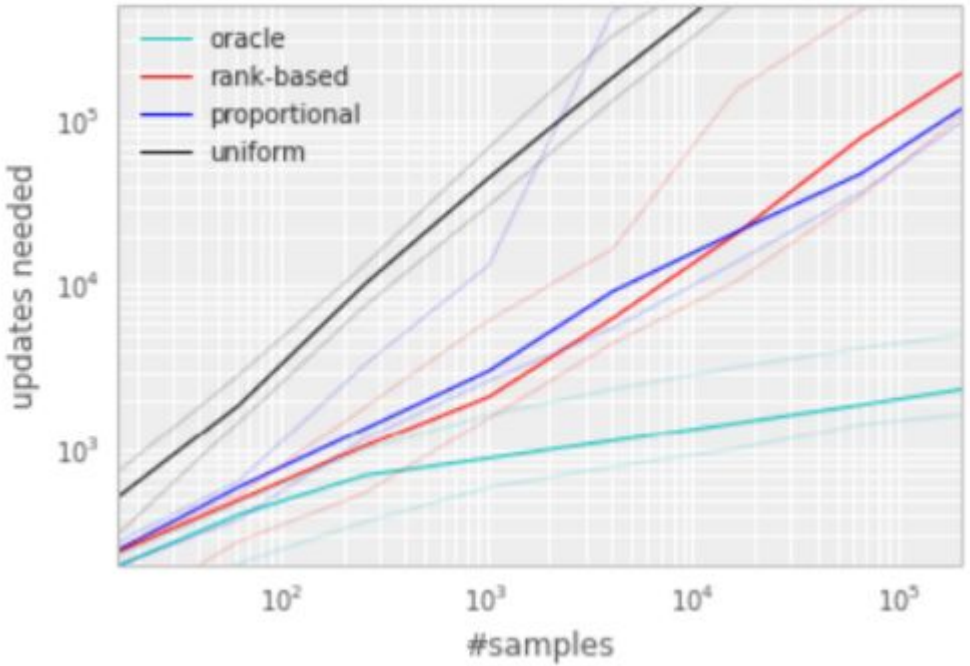
$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

$$p_i = |\delta_i| + \epsilon \quad \text{or} \quad p_i = \frac{1}{\text{rank}(i)}$$

알파 = 우선순위 영향정도(0~1). 0이면 uniform random과 같음.(hyper parameter)

direct, proportional한 방법과
indirect, rank-based한 방법이 있다.

rank-based한 것이 outlier에 덜 민감하고, 알파에 큰 영향을 받는다. 이게 robust하다고 할 수도 있다.



Annealing the bias

애초에 uniform을 택하지 않음으로써,diversity를 어느정도 포기했기 때문에 bias가 생겨버렸다.

이걸 해결하기 위해 Importance sampling(IS) weight를 델타에 곱해줌으로써, uniform sampling의 효과를 얻는다.

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta}$$

How?

$$\begin{aligned}\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X)f(X) \\ &= \sum Q(X)\frac{P(X)}{Q(X)}f(X) \\ &= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]\end{aligned}$$

P(X) 가 uniform이라고 했을때 1/N

Q(X) 는 우선순위화한 확률 P(i)

그래서 실제론 Q로 뽑지만 P로 뽑는 효과를 지닌다.

또한 학습의 안정화를 위해 1/max(w)를 곱해서 표준화

Overall Algorithm

Algorithm 1 Double DQN with proportional prioritization

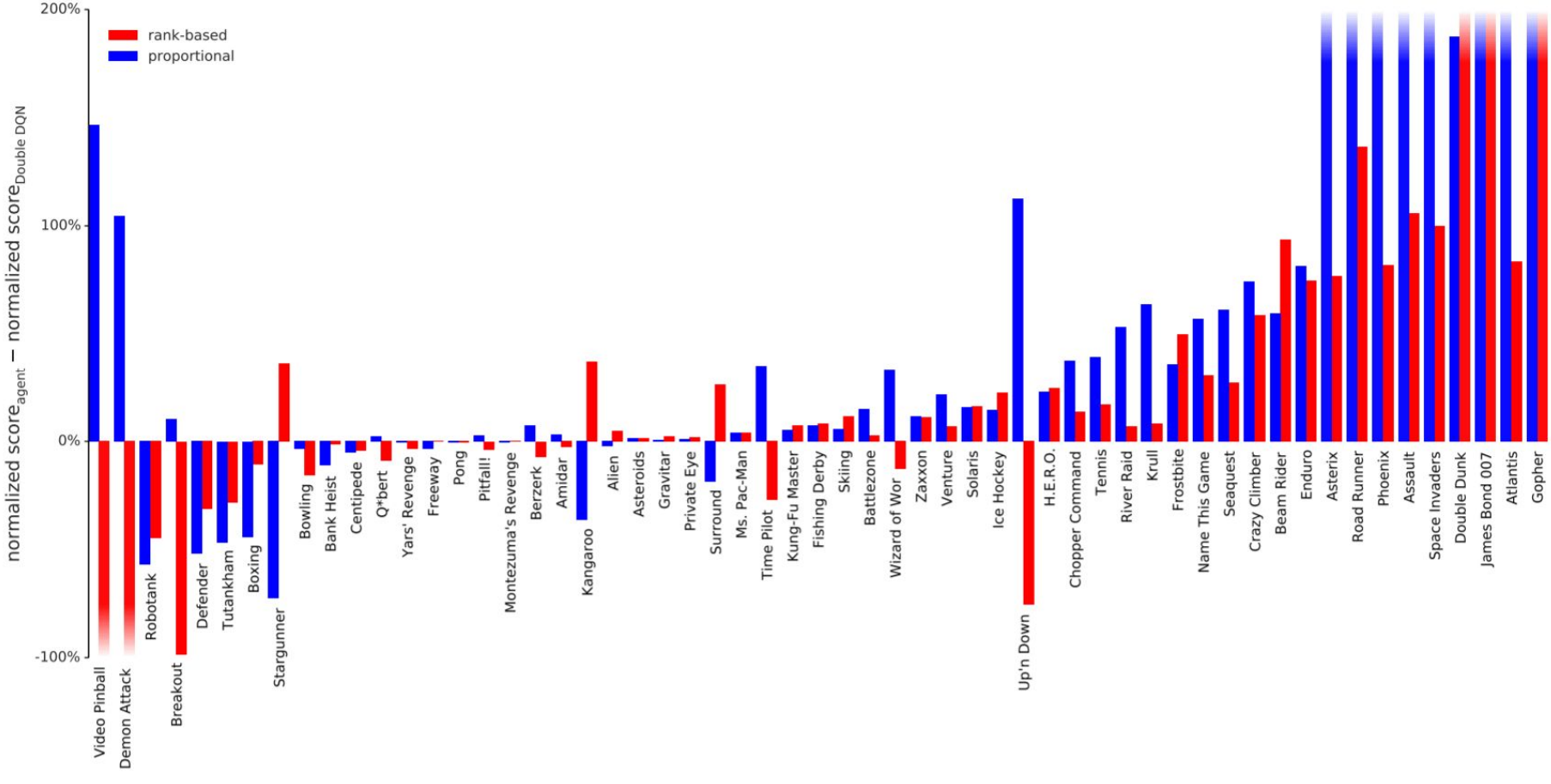
Input: minibatch k, step-size η, replay period K and size N, exponents α and β, budget T.
Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1$ **to** T **do**
  Observe S_t, R_t, γ_t
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$
  if $t \equiv 0 \mod K$ **then**
    for $j = 1$ **to** k **do**
      Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
      Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
      Compute TD-error $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
      Update transition priority $p_j \leftarrow |\delta_j|$
      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
    From time to time copy weights into target network $\theta_{\text{target}} \leftarrow \theta$
  end if
  Choose action $A_t \sim \pi_\theta(S_t)$
end for

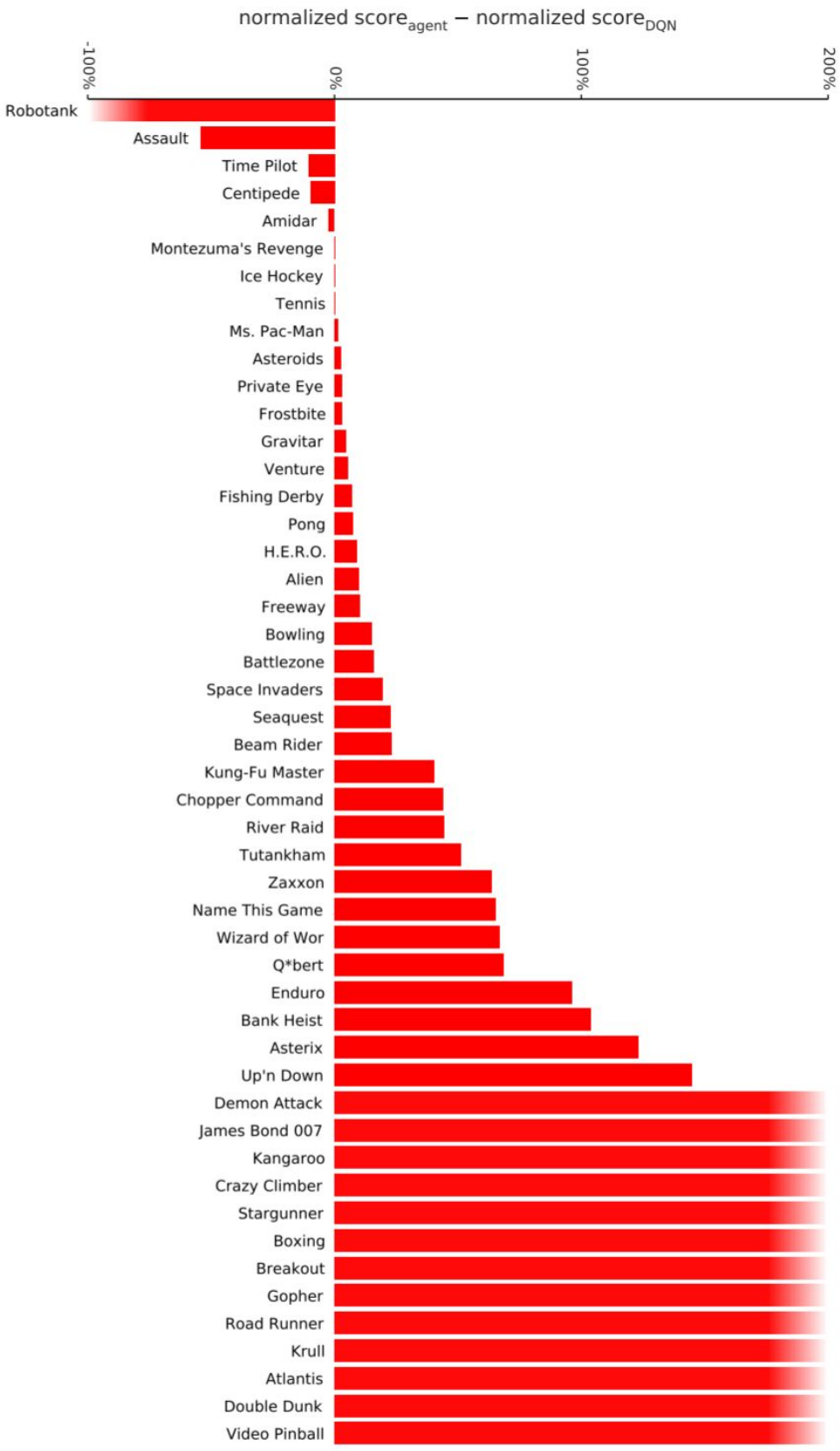
성능

아타리 게임중 랜덤 action 0%, human이 100%라고 했을때, 각 알고리즘 성능

	DQN		Double DQN (tuned)		
	baseline	rank-based	baseline	rank-based	proportional
Median	48%	106%	111%	113%	128%
Mean	122%	355%	418%	454%	551%
> baseline	–	41	–	38	42
> human	15	25	30	33	33
# games	49	49	57	57	57

Table 1: Summary of normalized scores. See Table 6 in the appendix for full results.


rank-based
proportional
200%
100%
0%
-100%
normalized score$_{agent}$ − normalized score$_{Double\ DQN}$
Video Pinball
Demon Attack
Robotank
Breakout
Defender
Tutankham
Boxing
Stargunner
Bowling
Bank Heist
Centipede
Q*bert
Yars' Revenge
Freeway
Pong
Pitfall!
Montezuma's Revenge
Berzerk
Amidar
Kangaroo
Alien
Asteroids
Gravitar
Private Eye
Surround
Ms. Pac-Man
Time Pilot
Kung-Fu Master
Fishing Derby
Skiing
Battlezone
Wizard of Wor
Zaxxon
Venture
Solaris
Ice Hockey
Up'n Down
H.E.R.O.
Chopper Command
Tennis
River Raid
Krull
Frostbite
Name This Game
Seaquest
Crazy Climber
Beam Rider
Enduro
Asterix
Road Runner
Phoenix
Assault
Space Invaders
Double Dunk
James Bond 007
Atlantis
Gopher


normalized score$_{agent}$ − normalized score$_{DQN}$
-100%
0%
100%
200%
Robotank
Assault
Time Pilot
Centipede
Amidar
Montezuma's Revenge
Ice Hockey
Tennis
Ms. Pac-Man
Asteroids
Private Eye
Frostbite
Gravitar
Venture
Fishing Derby
Pong
H.E.R.O.
Alien
Freeway
Bowling
Battlezone
Space Invaders
Seaquest
Beam Rider
Kung-Fu Master
Chopper Command
River Raid
Tutankham
Zaxxon
Name This Game
Wizard of Wor
Q*bert
Enduro
Bank Heist
Asterix
Up'n Down
Demon Attack
James Bond 007
Kangaroo
Crazy Climber
Stargunner
Boxing
Breakout
Gopher
Road Runner
Krull
Atlantis
Double Dunk
Video Pinball

Extension

지도학습 : dataset sample의 last-seen error를 기반. rare class에 있는 sample 더 많이 뽑기

Off-policy replay : rejection sampling과 importance sampling ratio의 개념과 비슷(p와 w)

Feedback for Exploration : 해당 transition i가 총 몇번 replay 된 수 M을 통해 exploration 하이퍼파라미터 업데이트

Prioritized Memories : 어떤 transition이 쓸모 없는지 알면, memory 사이즈를 줄일수 있음. DQN이 더이상 network size가 중요하지 않고, memory의 사이즈가 중요하므로 이건 하찮은 문제가 아님. e.g 해당 transtion의 age를 추적하여 오래된 쓸모있는 transition 계속 저장해두기(‘hall of fame’ in multi-agent 문서), planner 혹은 human expert의 trajectory는 계속 남겨두기 등..

결론

효율적인 experience replay를 위한 방법을 소개하고, 2가지 버전을 연구했고, 2배이상 학습속도를 높였다.