Deep Reinforcement Learning

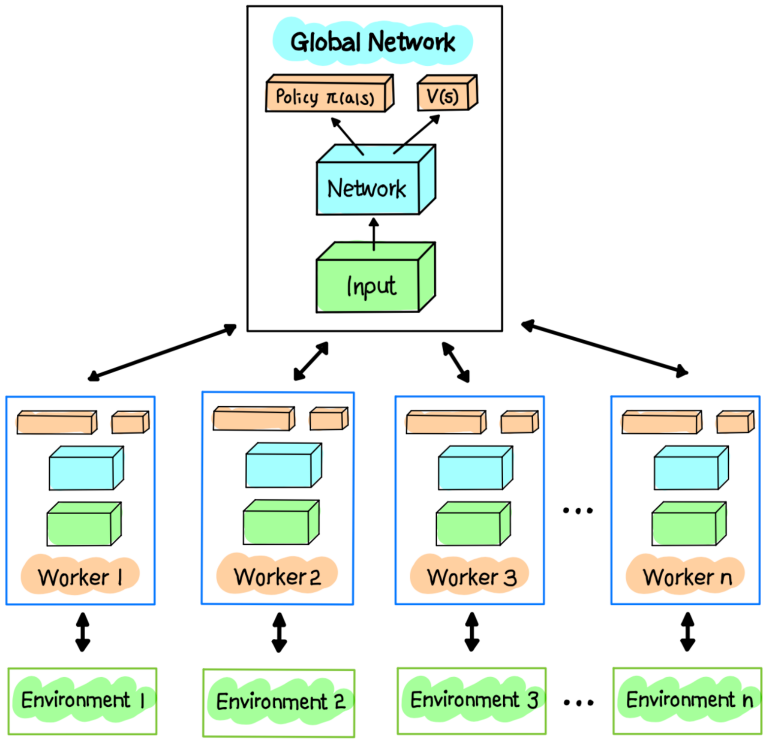
25 A3C

Yunseong Cho

목차

Review

A3C의 구조



여러 개의 worker agent network들을 asynchronous하게 global network를 업데이트 하는 방식

Review

Updating global network

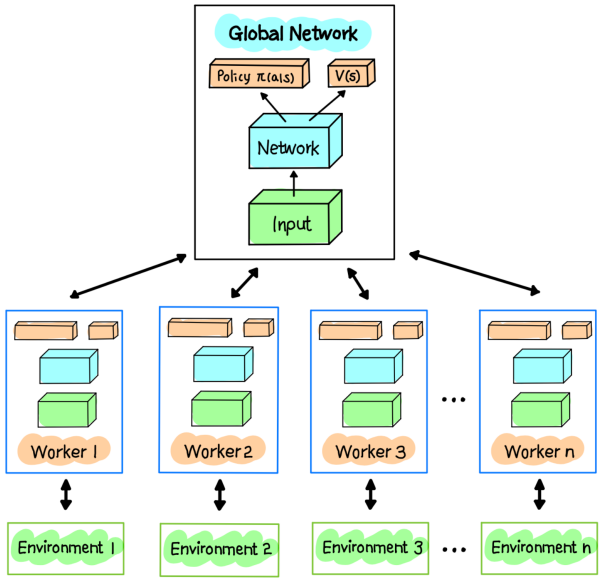
1. Global network parameter를 worker agent에게 copy.

2. Worker agent에 present state을 입력하여 일정 길이의 trajectory를 생성.

3. 얻은 trajectory의 segment들을 이용해서 gradient를 계산하고 그것을 누적.

4. 누적된 gradient값으로 global network parameter를 업데이트.

5. 업데이트 된 global network의 parameter를 다시 copy.



Review

Objective function form

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T-1} G_t \nabla_\theta log\pi_\theta(a_t|s_t)\,]$$ REINFORCE

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T-1} (G_t - V_\phi(s_t))\nabla_\theta log\pi_\theta(a_t|s_t)\,]$$ REINFORCE with baseline

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[(r + \gamma V_\phi(s') - V_\phi(s))\nabla_\theta log\pi_\theta(a_t|s_t)]$$ TD Actor-Critic

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[A_{\phi_1,\phi_2}(s,a)\nabla_\theta log\pi_\theta(a_t|s_t)]$$ Advantage Actor-Critic

$$= Q_{\phi_2}(s,a) - V_{\phi_1}(s)$$

저 파트는 actor network를 업데이트할 때 policy가 고른 action이 얼마나 좋은지를 나타냄

A3C

A3C = Asynchronous Advantage Actor-Critic

목적함수에서 advantage function을 사용

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[A_{\phi_1,\phi_2}(s,a)\nabla_\theta log\pi_\theta(a_t|s_t)]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[(G_t^{(n)} - V_\phi(s))\nabla_\theta log\pi_\theta(a_t|s_t)]$$

n-step return 을 실제로 사용함

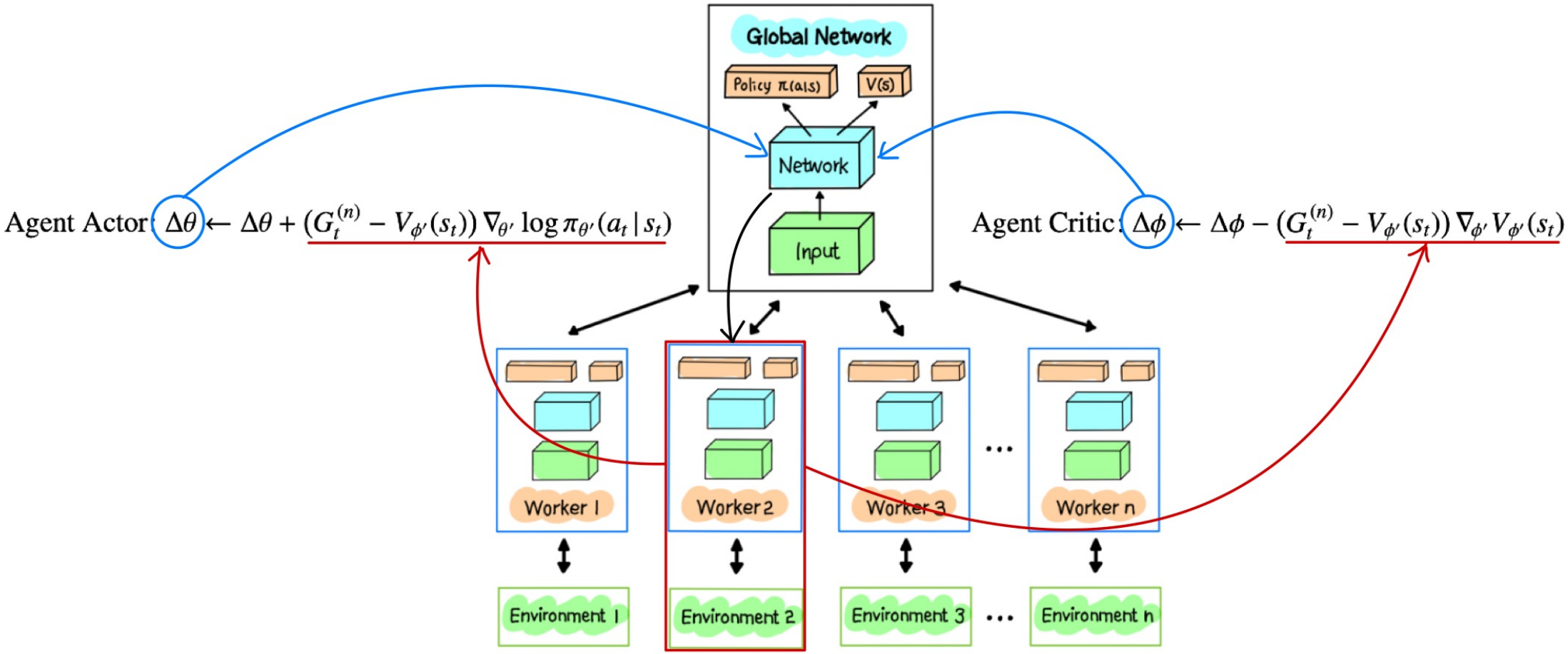
왜?

n-step까지는 실제 reward를 사용하고 그 이후를 approximate하기 때문에 variance가 작고 학습이 빠름

또한, advantage function에 비해서 네트워크가 하나 적음

$$G_t^{(n)} = R_{t+1} + \cdots + \gamma^{n-1}R_{n+t} + \gamma^n V(s_{n+t})$$

A3C


Agent Actor: $\Delta\theta \leftarrow \Delta\theta + (G_t^{(n)} - V_{\phi'}(s_t)) \nabla_{\theta'} \log \pi_{\theta'}(a_t \mid s_t)$
Agent Critic: $\Delta\phi \leftarrow \Delta\phi - (G_t^{(n)} - V_{\phi'}(s_t)) \nabla_{\phi'} V_{\phi'}(s_t)$
Global Network
Policy π(a|s)
V(s)
Network
Input
Worker 1
Worker 2
Worker 3
Worker n
Environment 1
Environment 2
Environment 3
Environment n

A3C (each worker agent)

Global network's shared parameters θ, ϕ, and one worker agent's parameters θ', ϕ'

global time step — Initialize step counter $t \leftarrow 1$

for *true* **do**

  Reset gradients $d\theta \leftarrow 0$, $d\phi \leftarrow 0$ — initialize gradients

  Synchronize parameters $\theta' = \theta$, $\phi' = \phi$

initial state — Set $t_{\text{start}} = t$, and get state s_t

  for s_t *not terminal and* $t \leq t_{\text{start}} + t_{\max}$ **do** — # of segments

generate trajectory — Select action a_t according to policy $\pi(a_t \mid s_t; \theta')$

generate trajectory — Execute a_t and observe r_{t+1}, s_{t+1}, and $t \leftarrow t+1$

  end

calculate return — $R = V(s_t; \phi')$ (or 0 for terminal s_t)

calculate return — **for** $i \in \{t-1, \ldots, t_{\text{start}}\}$ **do**

calculate return — $R \leftarrow r_{i+1} + \gamma R$

  Accumulate gradients w.r.t. θ': $d\theta \leftarrow d\theta + (R - V(s_i; \phi')) \nabla_{\theta'} \log \pi(a_i \mid s_i; \theta')$

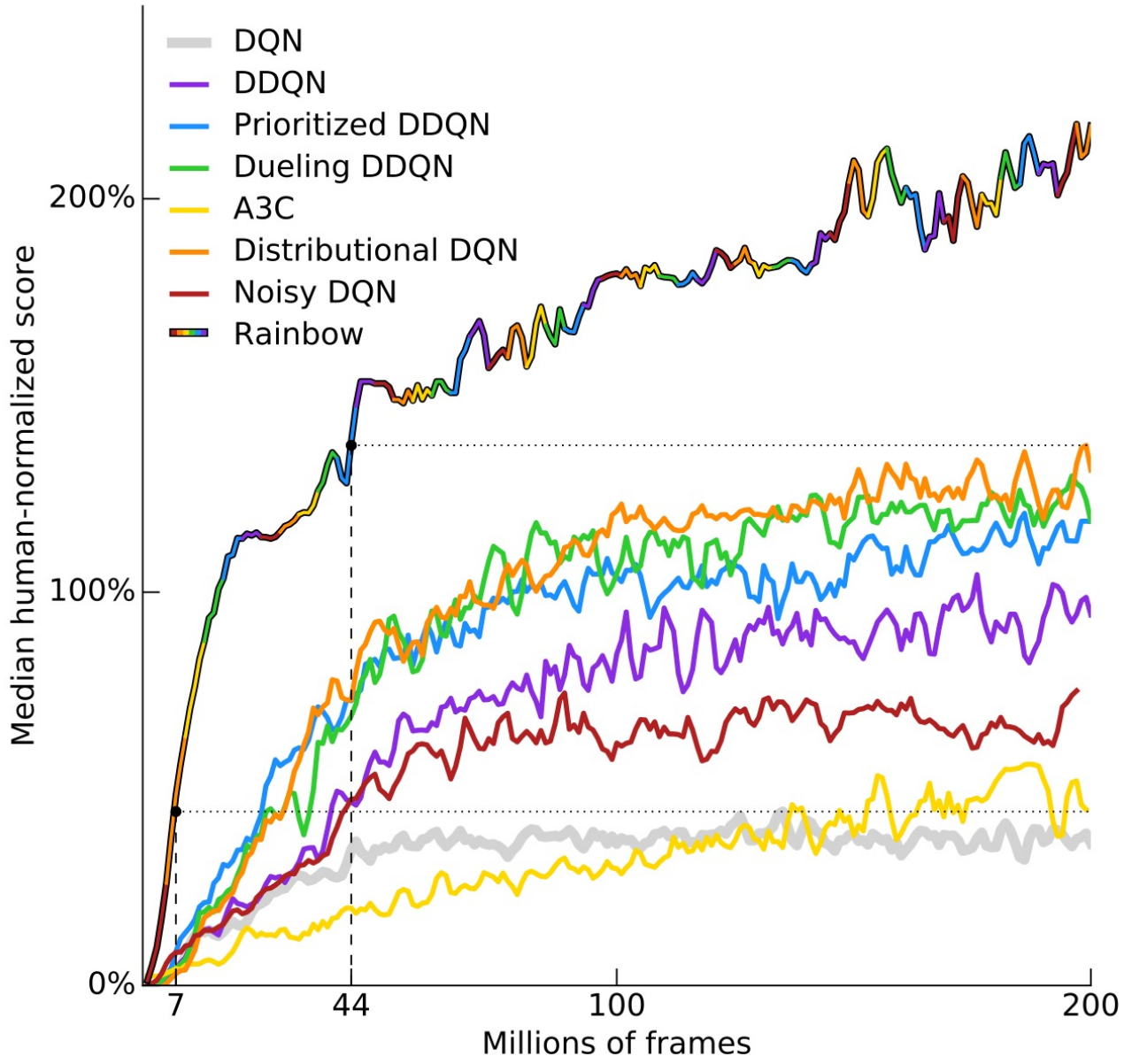
  Accumulate gradients w.r.t. ϕ': $d\phi \leftarrow d\phi - (R - V(s_i; \phi')) \nabla_{\phi'} V(s_i; \phi')$

  end

  Update asynchronously $\theta \leftarrow \theta + \alpha d\theta$ and $\phi \leftarrow \phi - \beta d\phi$

end

* Using CNN with one softmax output for $\pi(a_t \mid s_t; \theta)$ and one linear output for $V(s_i; \phi)$, we practically share all parameters except the output layers.

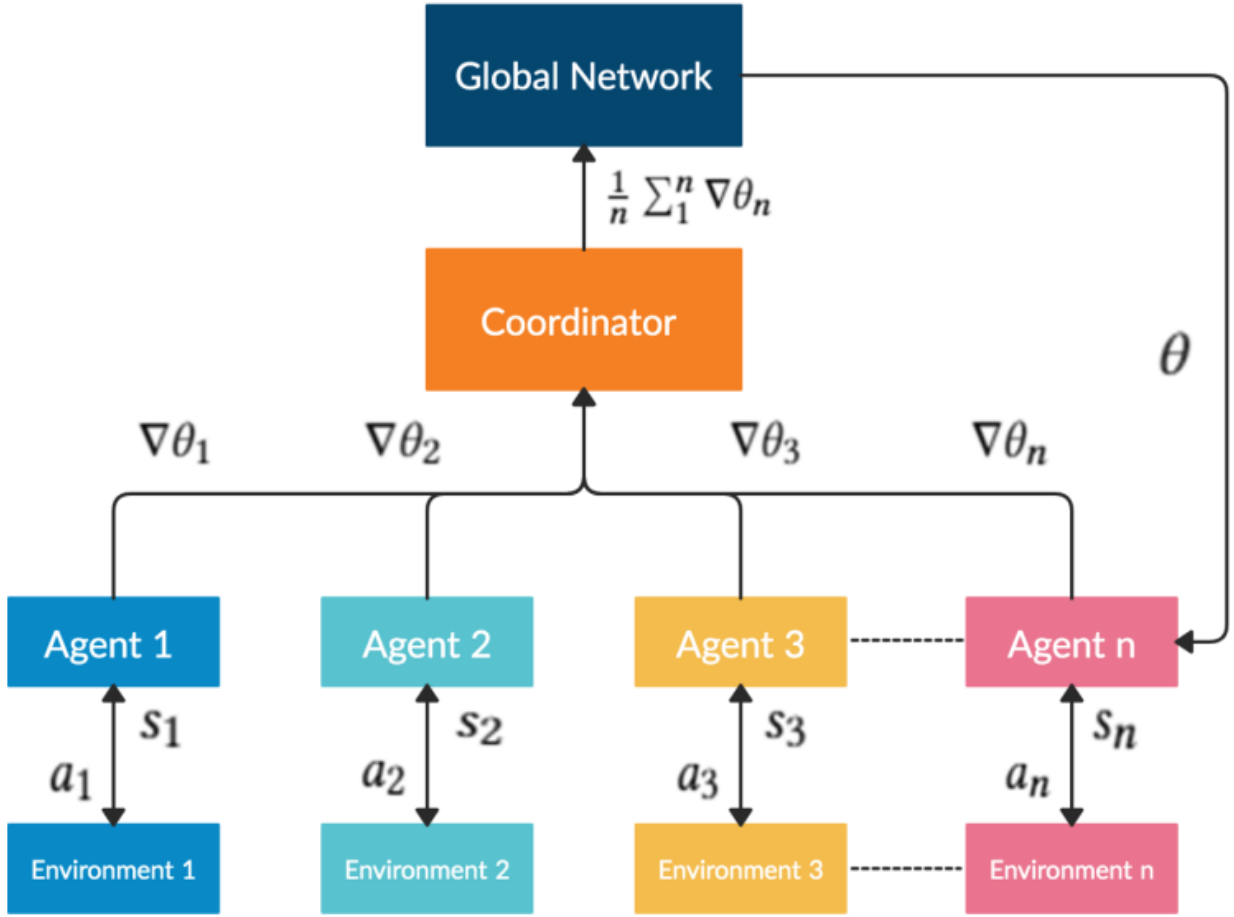


출처 : https://arxiv.org/abs/1710.02298

A2C

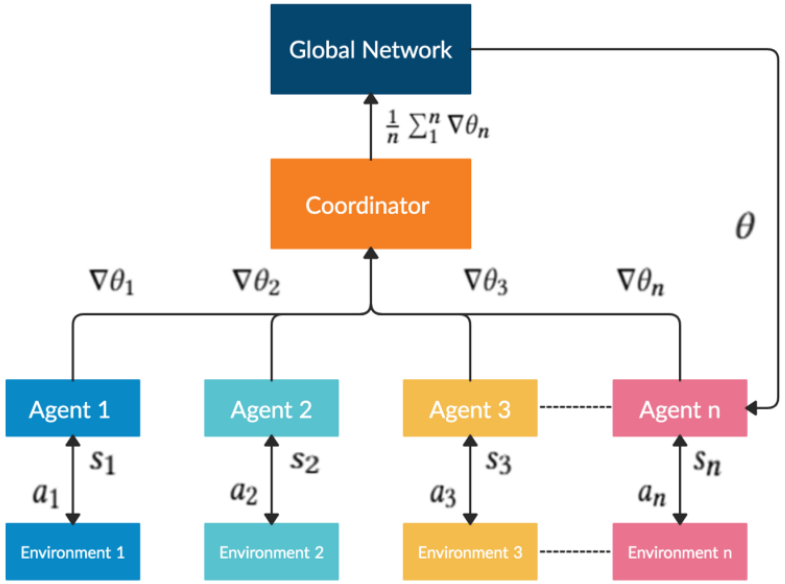
A2C = (Synchronous) Advantage Actor-Critic

= A3C와 다르게 각 agent가 동시에 global network의 parameter를 업데이트



A2C

- A2C는 각 agent가 전달한 gradient들을 coordinator에서 받아서 global network에게 average gradient를 전달하여 업데이트한다.
- 이러한 구조 덕분에 좀 더 응집력 있는(cohesive) 업데이트가 가능하여 수렴 속도가 더 빠르다.



- 또한 A3C는 각 agent가 서로 다른 policy로 서로 다른 샘플을 학습하기 때문에 업데이트를 해도 optimal하지 않을 수 있다.
- 하지만 A2C는 global network 업데이트 후에 모두 같은 policy를 copy하기 때문에 A3C에서 발생하는 inconsistency를 줄일 수 있다.

감사합니다