Reinforce Learning

Actor-Critic Algorithm

소프트웨어 꼰대 강의

노기섭 교수

(kafa46@hongik.ac.kr)

Recap: Policy Gradiant (REINFORCE)

Recap: Policy Gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)\ \nabla_\theta \log \pi_\theta(a_t \mid s_t)]$$

Remind this:

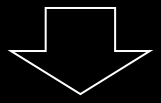
$$\approx \sum_{t \geq 0} r(\tau)\ \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

1. Sample trajectory τ^i from $\pi_\theta(a_t \mid s_t)$

2. Monte Carlo Estimation

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\, G_{i,t}$$

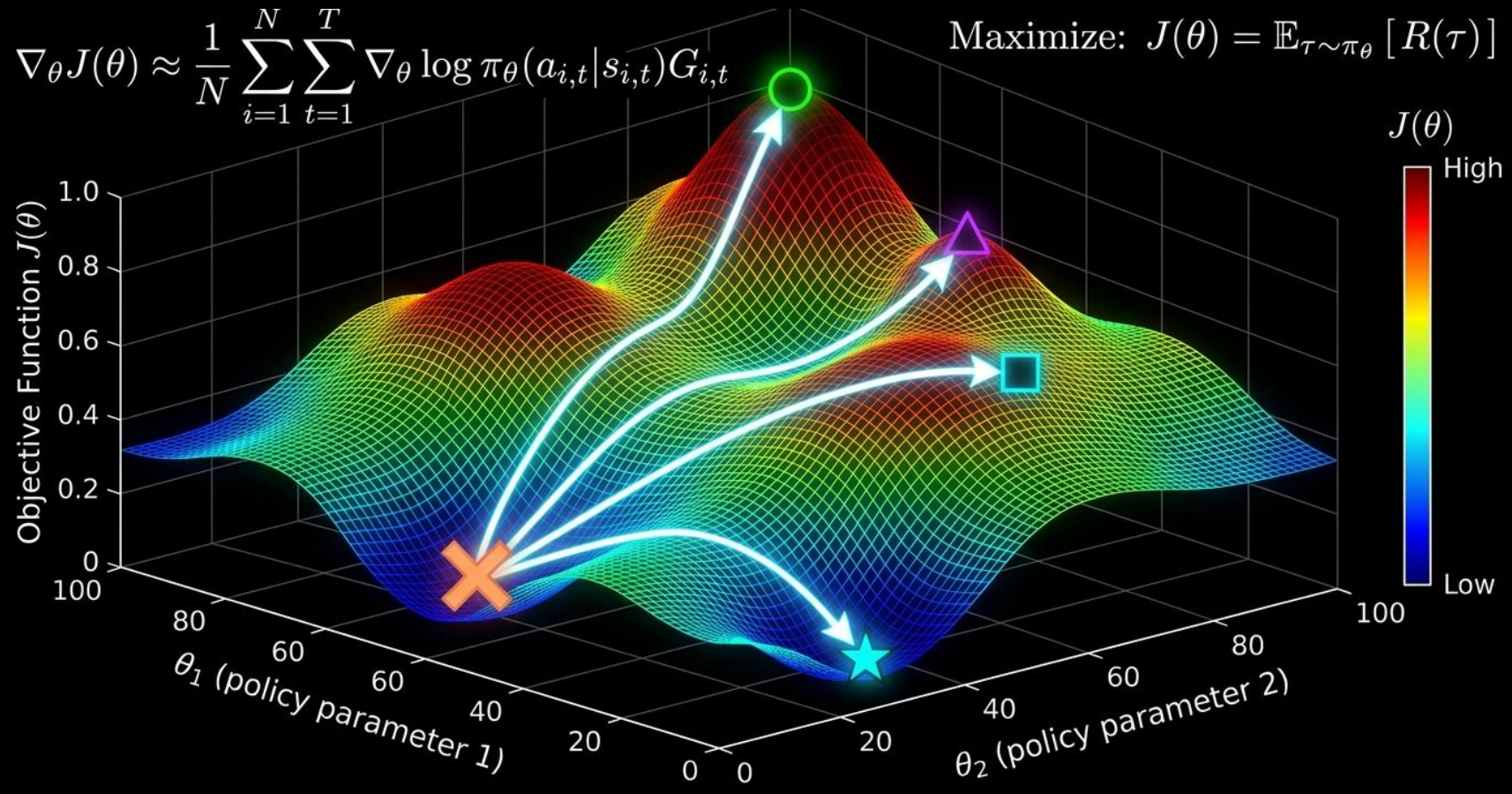
$$, where\ G_{i,t} = \sum_{k=t}^{T} r(s_{i,k}, a_{i,k})$$



3. Gradient Ascent: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Recap: REINFORCE

REINFORCE climbs the objective surface using stochastic gradient ascent.



Why is this Problematic

1. Sample trajectory τ^i from $\pi_\theta(a_t \mid s_t)$

2. Monte Carlo Estimation

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\, G_{i,t}$$

$$, where\ G_{i,t} = \sum_{k=t}^{T} r(s_{i,k}, a_{i,k})$$

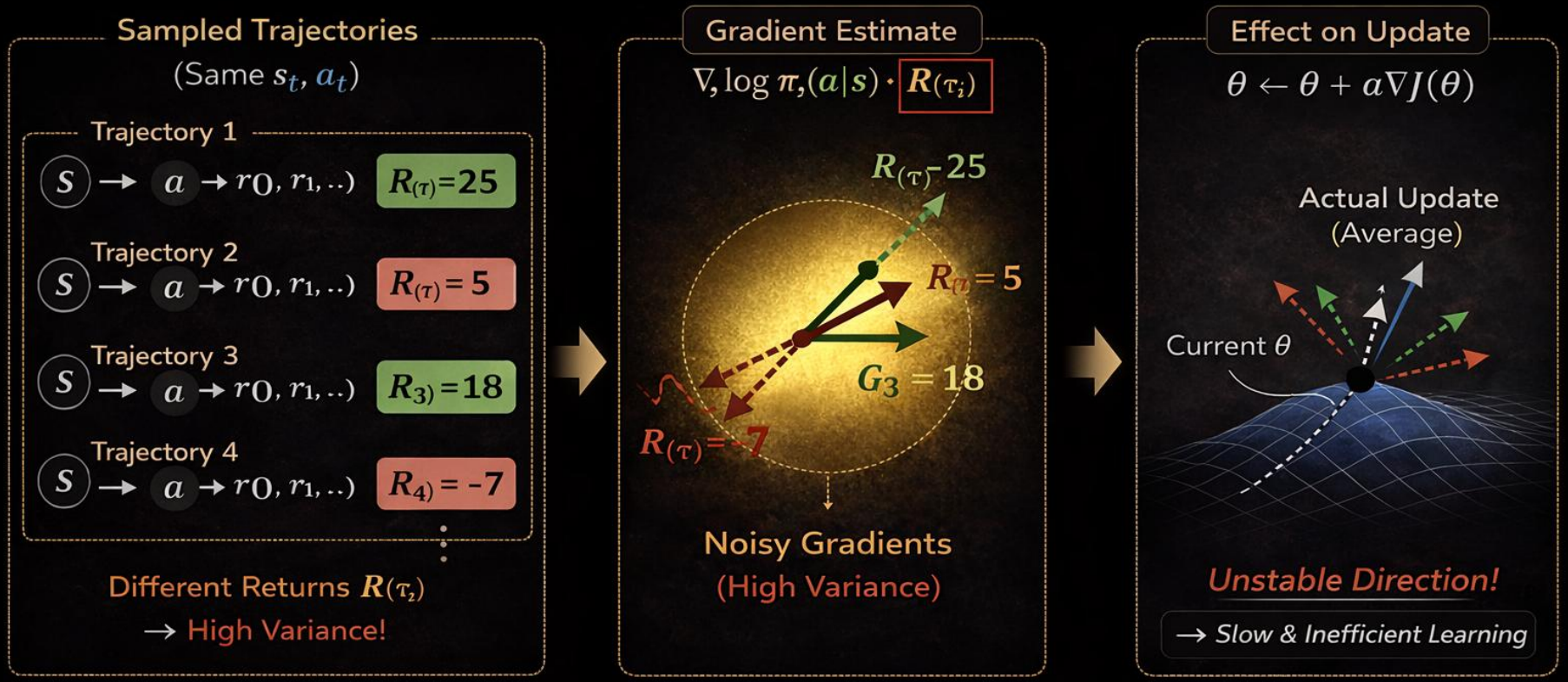
Entire Trajectory Return Required before Update! (All actions receive the same total return.)

3. Gradient Ascent: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

High variance + Slow learning

The Direction to Tackle REINFORCE

Same State-Action (s, a) → Different Trajectory Returns $R(\tau)$ → Noisy Gradient Estimate



Solution: Reduce Variance using Baseline $V(s)$ → Advantage $A(s,a) = R(\tau) - V(s)$

Actor-Critic Algorithm

Can We Learn Step by Step?

1. Sample trajectory τ^i from $\pi_\theta(a_t \mid s_t)$

2. Monte Carlo Estimation

Can we learn step by step?

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\, G_{i,t}$$

$$, where\ G_{i,t} = \sum_{k=t}^{T} r(s_{i,k}, a_{i,k})$$

3. Gradient Ascent: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Can we replace Monte Carlo return with bootstrapped estimate?

$$G_{i,t} \approx r_t + \gamma V(s_{t+1})$$

Introducing Actor-Critic Algorithm

Please tell me how good is the policy is at an earlier stage?

He, my name is Ms. Critic.

I will give you score in every step!

Bridge to Actor-Critic: Policy Gradient Theorem (Sutton, 1999)

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\, G_{i,t}$$

$$, where\ G_{i,t} = \sum_{k=t}^{T} r(s_{i,k}, a_{i,k})$$

Replace

In REINFORCE, we approximate the action-value function using the full return G_t.

According to the Policy Gradient Theorem, the Monte Carlo return G_t can be replaced by the true action-value function $Q^\pi(s_t, a_t)$.

$\hat{Q}(s_{i,t}, a_{i,t})$

Approximation

$Q^\pi(s_{i,t}, a_{i,t})$

Policy Gradient Theorem: Sutton et al., 1999.

https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf

How to compute Q-value?

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\, G_{i,t}$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\, \hat{Q}(s_{i,t}, a_{i,t})$$

How to get action-value function $\hat{Q}$?

Baseline Structure of Actor-Critic

Policy Net

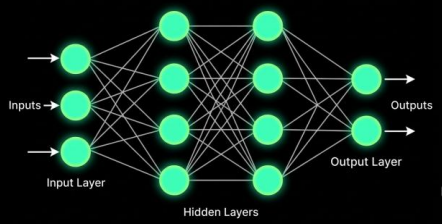
Critic Net

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\, \hat{Q}(s_{i,t}, a_{i,t})$$

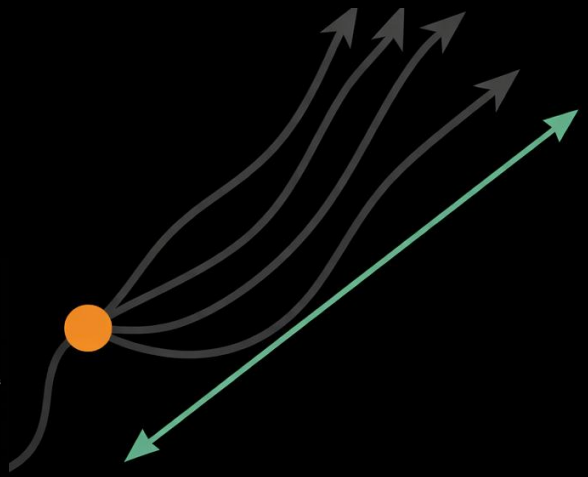
How to get action-value function $\hat{Q}$?

Using another neural network
to approximate value function called Critic.
(That's why we call this network as "Actor-Critic")

Using Critic network, we can update NN step by step.



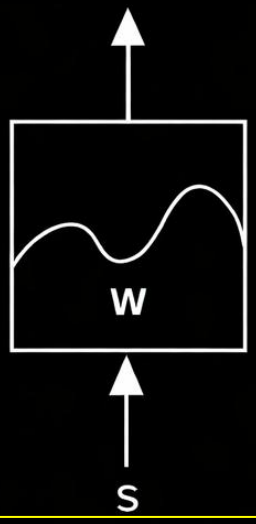
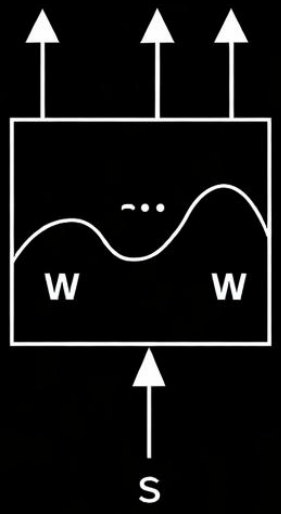
One more thing!
We can introduce "Bias".
REINFFORCE: low bias, variance: very high
Actor-Critic: increase little bias, but reduce variance a lot



$$Q_{i,t} \approx \sum_{t'=t}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})]s_{i,t}, a_t]$$

Possible Critic Architectures

$w: params\ of\ Critic$

State-Value Function Critic	Action-Value Function Critic	Action-Value Function Critic
$\hat{V}(s;w)$	$\hat{Q}(s,a;w)$	$\hat{Q}(s,a_1;w) \cdots \hat{Q}(s,a_n;w)$
		
✓ Simple ✓ Suitable for continuous actions	✓ Exact Action-Value ✓ No need advantage ✓ Difficult to optimize in continuous action spaces	✓ Exact Action-Value ✓ DQN style ✓ Proper to discrete action

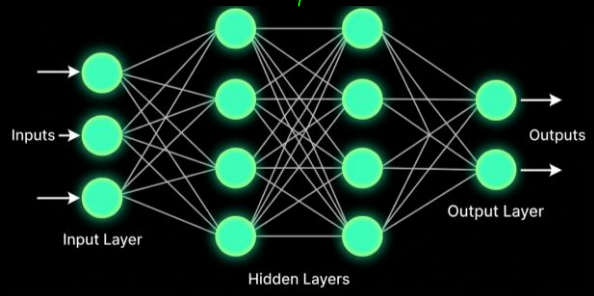
In most modern Actor–Critic methods such as A2C and PPO, we use a value-function critic $V(s)$.
The Q-function critic is mainly used in deterministic or continuous-action methods.

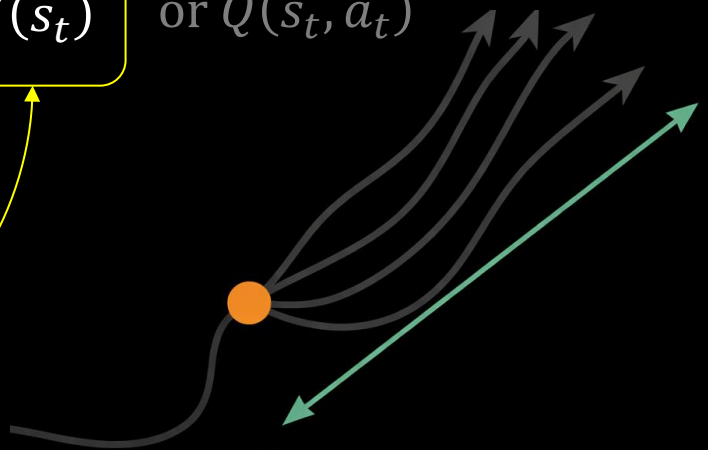
Basic Architecture of Actor–Critic

Policy Net

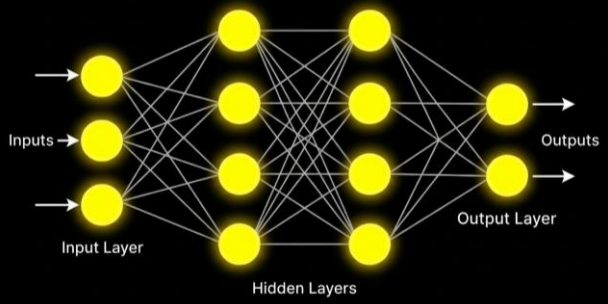
Critic Net

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\ \hat{V}(s_t) \quad \text{or } \hat{Q}(s_t, a_t)$$





$$Q_{i,t} \approx \sum_{t'=t}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})]s_{i,t}, a_t]$$



Approximate expected future return!

Advanced Actor-Critic (A2C)

Idea of Baseline

Policy Gradiant

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t]$$

Randomness 1: Random Sampling

Randomness 2: Return: Highly Noisy

This produces high variance!

Let's subtract "baseline" from G_t to reduce variance

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(G_t - b(s_t)\right)\right]$$

$$, where\ b(s_t) = V(s_t)$$

Advantage: $A(s_t, a_t) = G_t - V(s_t)$

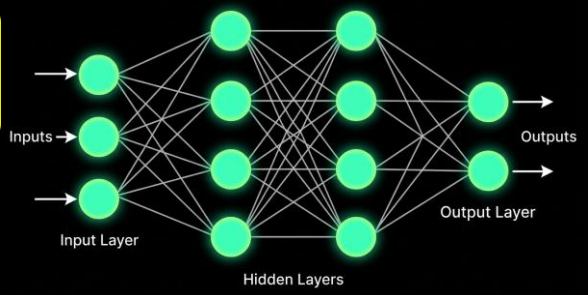
Bridge to Advantage Actor-Critic!

Again, Remind "Structure of Actor-Critic"

Policy Net

Critic Net

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\, A(s_{i,t}, a_{i,t})$$

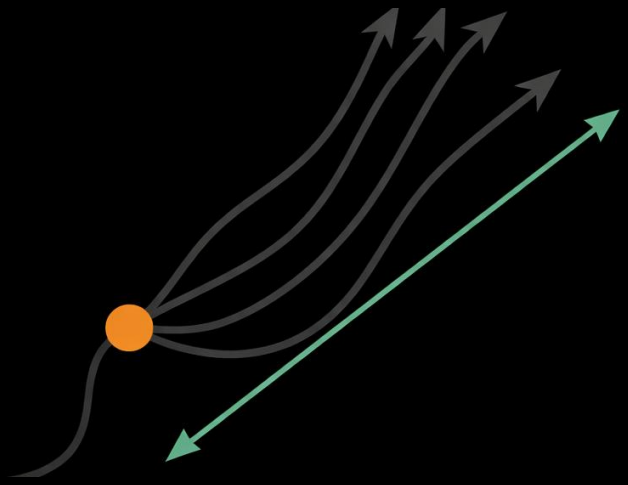


This objective has lower variance but introduces bias compared to REINFORCE due to TD bootstrapping.

Can we also subtract a baseline to reduce the variance?

✓ YES

✓ And we should!



Idea of Baseline

Policy Gradiant

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\nabla_\theta \log \underline{\pi_\theta(a_t \mid s_t)}\, \underline{G_t}\right]$$

Randomness 1: Random Sampling

Randomness 2: Return: Highly Noisy

This produces high variance since trajectory stochastic.

Let

$X = \nabla_\theta \log \pi_\theta(a_t \mid s_t)$

$Y = G_t$ then $Var(XY)$

If $Var(Y)$ is high, $Var(XY)$ become higher

Let's subtract "baseline $b(s_t) \approx V(s_t)$" from G_t to reduce variance

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\big(G_t - b(s_t)\big)\right]$$

The result: Expectation is unchanged!

But why does the expected value remain unchanged even if Baseline is subtracted?
(Go to next slide for more details.)

Why Expected Value remains unchanged?

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(G_t - b(s_t)\right)\right]$$

$$, where\ b\ is\ baseline$$

$$\mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) b(s_t)] = 0$$

Because

$$\mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)] = 0$$

Go to next slide for more details.

Why Expected Value remains unchanged? (cont.)

$\mathbb{E}_{\tau\sim\pi\_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)] = 0$

Recap: log derivation trick!

$$\nabla_\theta \log \pi_\theta(a_t \mid s_t) = \frac{\nabla_\theta \pi_\theta(a_t \mid s_t)}{\pi_\theta(a_t \mid s_t)}$$

Substitute

Just apply log derivation trick!

$$\mathbb{E}_{\tau\sim\pi\_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)] = \sum_a \pi_\theta(a_t \mid s_t) \boxed{\nabla_\theta \pi_\theta \log(a_t \mid s_t)}$$

$$= \sum_a \cancel{\pi_\theta(a_t \mid s_t)} \frac{\nabla_\theta \pi_\theta(a_t \mid s_t)}{\cancel{\pi_\theta(a_t \mid s_t)}}$$

$$= \sum_a \nabla_\theta \pi_\theta(a_t \mid s_t)$$

Continued the next slide

Why Expected Value remains unchanged? (cont.)

$$\mathbb{E}_{\tau\sim\pi\_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)] = 0$$

$$\mathbb{E}_{\tau\sim\pi\_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)] = \sum_a \nabla_\theta \pi_\theta(a_t \mid s_t)$$

Since Policy is a probability distribution,

$$\sum_a \pi_\theta(a_t \mid s_t) = 1$$

If both sides are differentiated by θ,

$$\nabla_\theta \sum_a \pi_\theta(a_t \mid s_t) = \nabla_\theta 1 = 0 \quad \Rightarrow \quad \sum_a \nabla_\theta \pi_\theta(a_t \mid s_t) = 0$$

Finally, we get $\mathbb{E}_{\tau\sim\pi\_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)] = 0$

The Role of Baseline?

Why is the variance reduced if we apply baseline?

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi\_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(G_t - b(s_t)\right)\right]$$

$, where\ b\ is\ baseline$

In Statistics, if subtract Mean from a random variable,
the variance is minimized!

The optimal baseline: $b^*(s) = \dfrac{\mathbb{E}[G_t \nabla(\log \pi)^2]}{\mathbb{E}[\nabla(\log \pi)^2]}$

Approximation of baseline: $V(s)$

No changes what we want know:
➔ How much better than average

But why?

Continued the next slide

Variance Analysis of Baseline

Baseline이 expectation을 바꾸지 않음
→ Variance Minimization 문제로 변환

Let simplify Policy Gradient Estimator as

$$g = XY$$
$$, where\ X = \nabla_\theta \log \pi_\theta(a|s)\ , Y = (G_t - b)$$

We want to minimize the variance of g, i.e., $Var(g)$

$$\mathbb{E}[g] = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s)(G_t - b)]$$

$$\mathbb{E}[g] = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) G_t] - \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) b]$$

Since we proved $\nabla_\theta \log \pi_\theta(a|s) = 0$

The expected value is independent of baseline!

The definition of variance

$$Var(g) = \mathbb{E}[g^2] - (\mathbb{E}[g])^2$$

$$\mathbb{E}[g^2] = \mathbb{E}[(X(G_t - b))^2]$$
$$= \mathbb{E}[X^2(G_t - b)^2]$$

Variance minimization is reduced to the problem of minimizing $\mathbb{E}[g^2]$.

Continued the next slide

Deriving the Optimal Baseline

We want to minimize $f(b)$

$$f(b) = \mathbb{E}[X^2(G_t - b)^2]$$

Differentiate $f(b)$ regarding b:

$$\frac{df}{db} = \mathbb{E}[X^2 \cdot 2(b - G_t)]$$

Since it has to be zero
from the optimum point:

$$\mathbb{E}[X^2(b - G_t)] = 0$$

$$b\mathbb{E}[X^2] = \mathbb{E}[X^2 G_t]$$

Therefore,

$$b^*(s) = \frac{\mathbb{E}[X^2 G_t]}{\mathbb{E}[X^2]}$$

$$= \frac{\mathbb{E}[G_t \nabla(\log \pi)^2]}{\mathbb{E}[\nabla(\log \pi)^2]}$$

Mathematically possible,

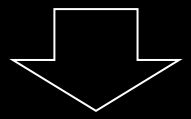
practically impossible.

Final approximation as:

$$b^*(s) \approx b(s)$$
$$= \mathbb{E}[G_t|s]$$
$$= V(s)$$

Why is the variance reduced?

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi\_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t\,]$$



$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi\_\theta}\big[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\big(G_t - b(s_t)\big)\big]$$

We're not just subtracting the average!

See previous slide:
$b\mathbb{E}[X^2] = \mathbb{E}[X^2 G_t]$

We're subtracting a weighted mean of returns, weighted by X^2!

The optimal baseline minimizes the variance of
the gradient estimator by subtracting a weighted mean of returns.

Variance에 더 크게 기여하는 샘플들을 더 중요하게 평균

Moving to Practical Implementation (Advantage A2C)

The role of components in A2C

The Objective can be modified for exploiting bias

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\, \hat{Q}(s_{i,t}, a_{i,t})$$

좋은 행동인가?

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\left(Q(s_{i,t}, a_{i,t}) - V(s_{i,t})\right)$$

평균보다 얼마나 더 좋은 행동인가?

$$Q^\pi(s_t, a_t) = \sum_{t=1}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t]$$

How good the action we take from current state.

$$V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$$

The average return when other agent face the same state.

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

Advantage Function: reflecting how good action we have taken compared to other candidates.

Decomposing the Action-Value Function

$$Q^{\pi}(s_t, a_t) = \sum_{t=1}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t]$$

Monte Carlo 방식
- 미래 전체 trajectory 필요
- episode 끝까지 rollout 필요
- variance 매우 큼

How good the action we take from current state.

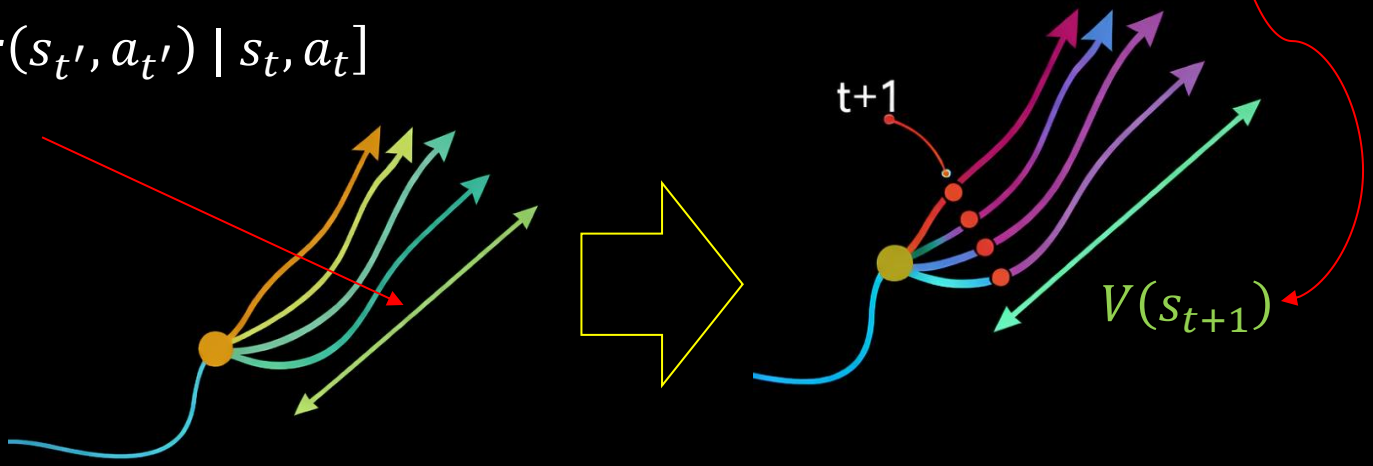
⇩

Q = immediate reward + future rewards

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \sum_{t'=t+1}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t]$$

$V(s_{t+1})$

$$Q_t \approx \sum_{t'=t}^{T} \mathbb{E}[r(s_{t'}, a_{t'}) \mid s_t, a_t]$$



Approximating the Action-Value Function

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \sum_{t'=t+1}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t]$$

Since $\mathbb{E}_{a\sim\pi}[Q^{\pi}(s_{t+1}, a)]$

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma V(s_{t+1})$$

} Bellman Equation

By approximation

$$Q^{\pi}(s_t, a_t) \approx r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})$$

Formulate Advantage

$$Q^{\pi}(s_t, a_t) \approx r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})$$

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$

By applying TD (Time Difference)

$$A^{\pi}(s_t, a_t) \approx r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

Derive TD error

$$\delta_t \approx r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t)$$

$$\delta_t \approx A(s_t, a_t)$$

Vanilla A2C does not explicitly learn a full Q-function

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \sum_{t'=t+1}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]$$

$\sum_{t'=t+1}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t] \rightarrow V(s_{t+1})$

Q = immediate reward + future rewards

$$A^{\pi}(s_t, a_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

Actor-Critic does NOT learn Q-function.

We can just fit value function using TD error!

Critic Net Training - Approach 1. Monte Carlo Evaluation

$$V^{\pi}(s_t) \approx \sum_{t'=t}^{T} \mathbb{E}[r(s_{t'}, a_{t'}) \mid s_t, a_t] = \sum_{t'=t}^{T} r(s_{t'}, a_{t'})$$

Generate multiple trajectories
(Train Dataset)

$$\left(s_{i,t} , \underbrace{\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})}_{y_{i,t}} \right)$$

실제 trajectory 끝까지 rollout 해서
return을 label로 생성

Compute the loss
by supervised regression

$$L(\phi) = \frac{1}{2} \sum_{i} \left\| \hat{V}_{\phi}^{\pi}(s_i) - y_i \right\|^2$$

Critic도 결국
supervised learning처럼 학습

Critic Net Training - Approach 2. TD Evaluation

$$y_{i,t} = \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'}) \mid s_t, a_t]$$

$$\approx r(s_{i,t}, a_{i,t}) + V^{\pi}(s_{i,t+1}) \approx r(s_{i,t}, a_{i,t}) + \hat{V}_{\phi}^{\pi}(s_{i,t+1})$$

critic 자신의 추정값으로 bootstrap

Generate multiple trajectories
(Train Dataset)

$$\left(s_{i,t}, \underbrace{r(s_{i,t}, a_{i,t}) + \hat{V}_{\phi}^{\pi}(s_{i,t+1})}_{y_{i,t}}\right)$$

미래를 끝까지 보지 말고
현재 critic이 미래를 대신 예측

Compute the loss
by supervised regression

$$L(\phi) = \frac{1}{2}\sum_{i} \left\| \hat{V}_{\phi}^{\pi}(s_i) - y_i \right\|^2$$

A2C Algorithm

1. Take an action, then get one-step experience (s, a, s', r)

2. Fit Value Function (Critic TD loss → Train Critic Net)

$$L(\phi) = \frac{1}{2}\sum_i \left\| \hat{V}_\phi^\pi(s_i) - y_i \right\|^2 \text{ , where } y_i = r_i + \gamma \hat{V}_\phi^\pi(s_{t+1})$$

3. Evaluate Advantage Function

$$A^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma \hat{V}_\phi^\pi(s_{t+1}) - \hat{V}_\phi^\pi(s_t)$$

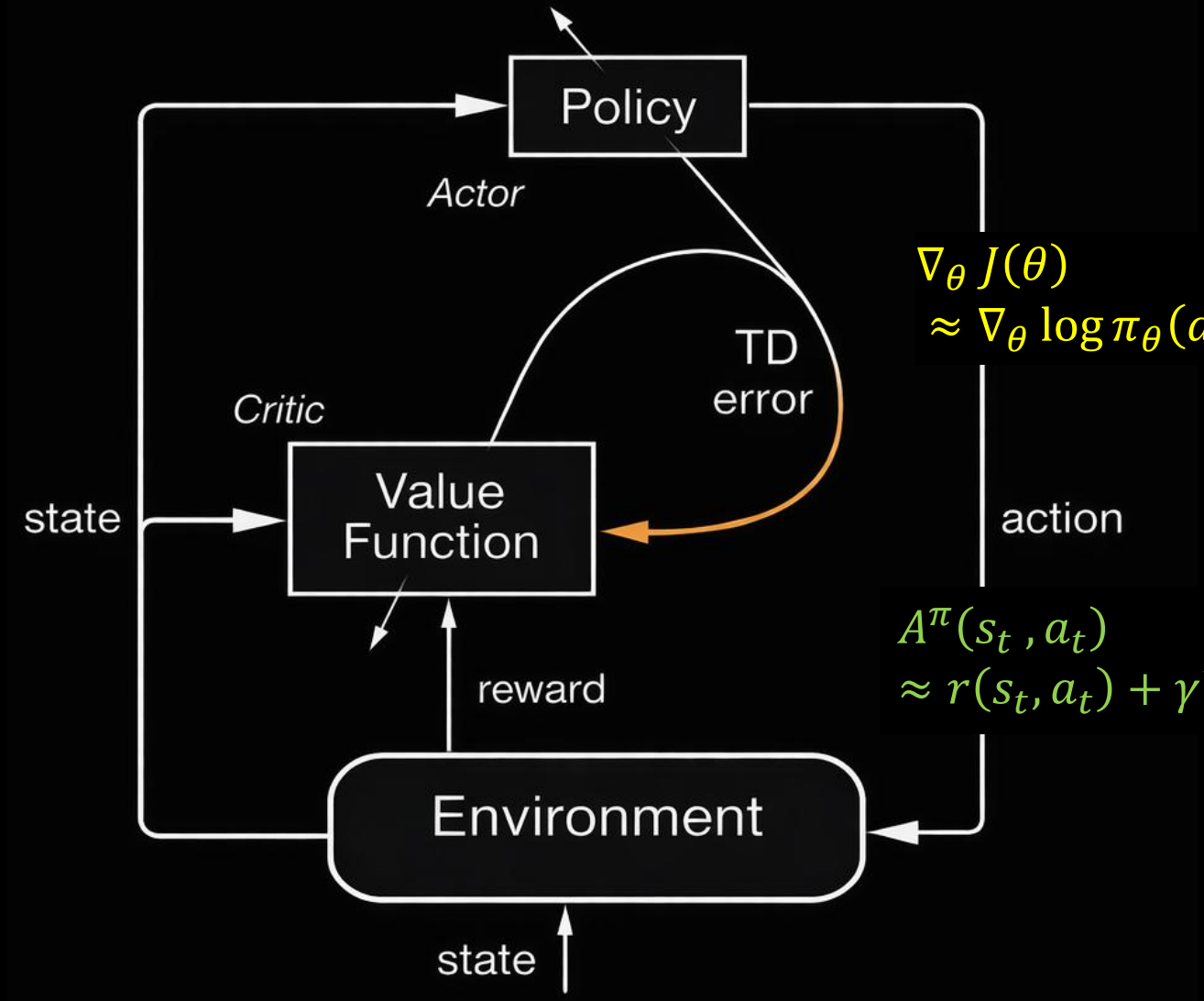
4. Compute Policy Gradient (Actor Update)

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s) \hat{A}^\pi(s, a)$$

5. Update Policy Parameters

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Overall A2C Architecture

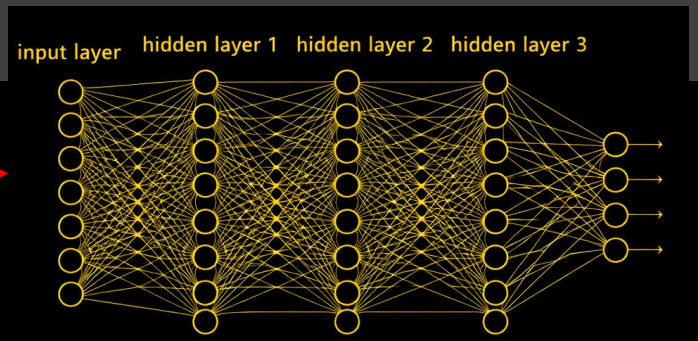


$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s) \hat{A}^\pi(s,a)$$

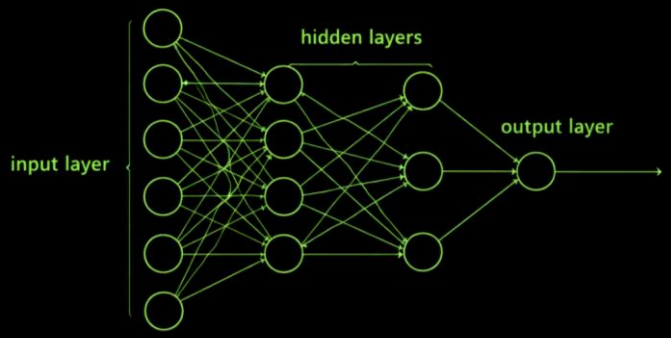
$$A^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma \hat{V}_\phi^\pi(s_{t+1}) - \hat{V}_\phi^\pi(s_t)$$

Network Architectures

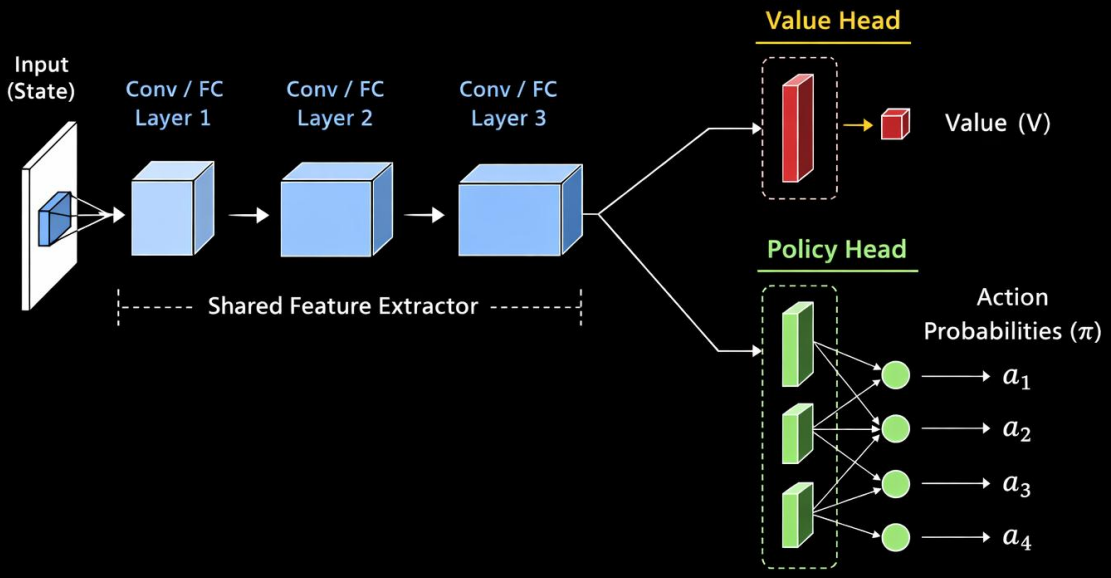
■ **Two possible NN Architecures:**



- Seperate Policy & Critic NN
  - More Params
  - More Stable in Training

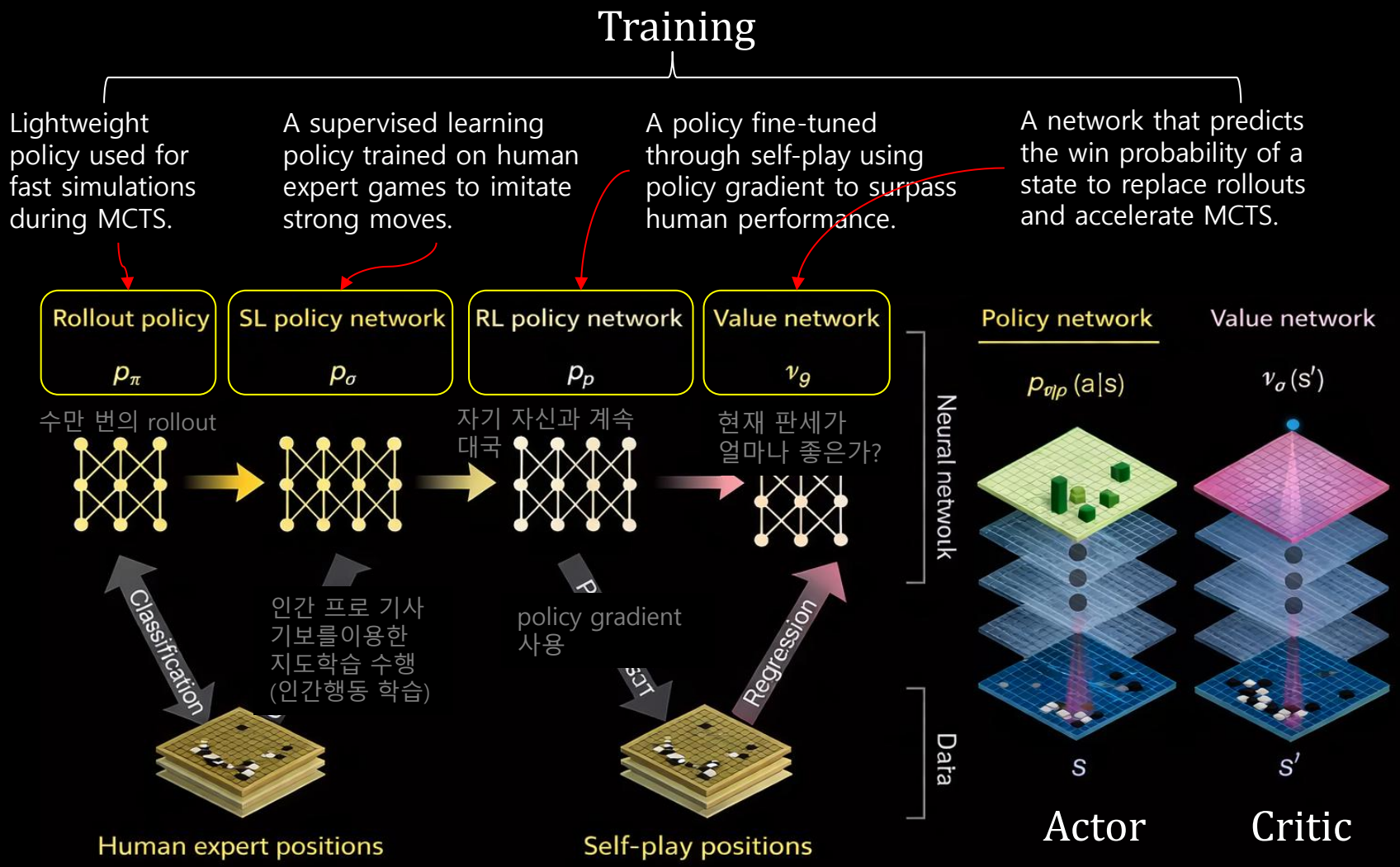


- Two-head Network
  - Share Features
  - Less Params
  - Hard to find good coefficient to balance Actor & Critic loss



AlphaGo: Seperate Network Architecture

■ **Monte Carlo Tree Search (MCTS) with Actor-Critic Algorithm**

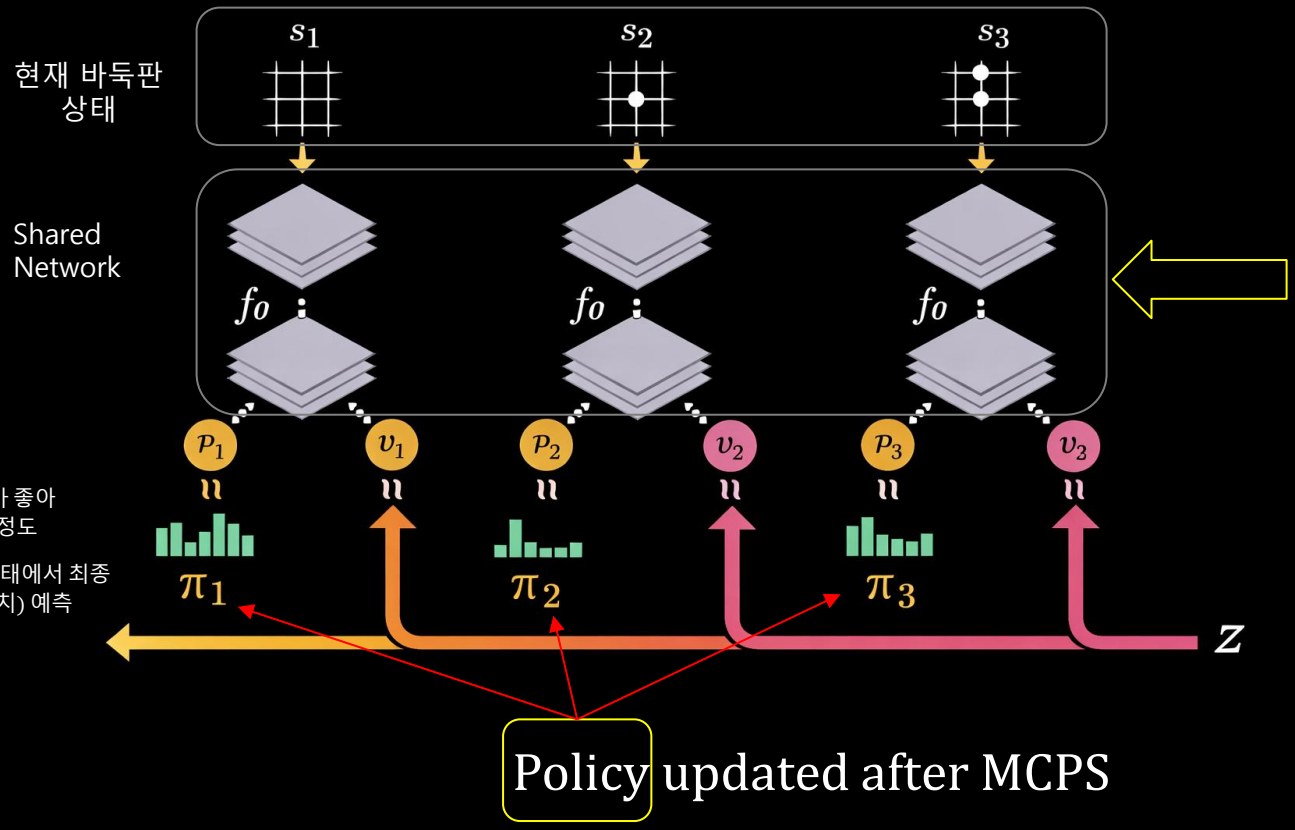


AlphaGo Zero: Architecture

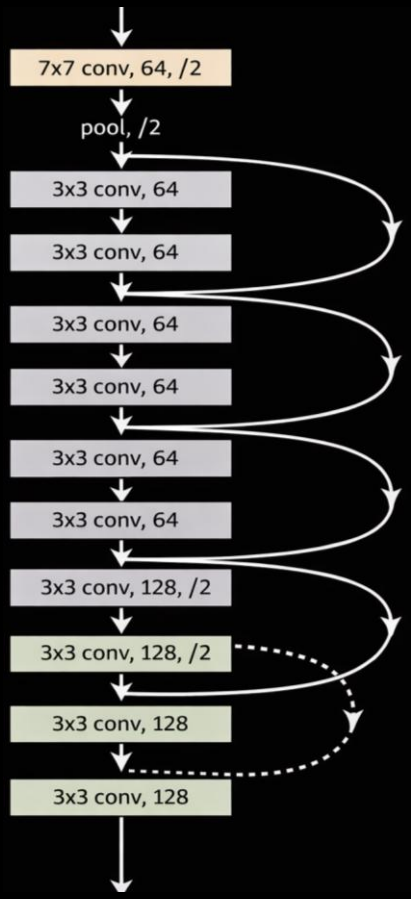
현 상황에서 n번 째까지 (끝까지) 둬보면서
가장 좋은 수를 고르는 방법론

- **Monte Carlo Tree Search (MCTS) with Actor-Critic Algorithm**
- **Shared ResNet Architecture**
  - ✓ Remove human data, only self-play

모든 수를 탐색 X
유망해 보이는 수를 트리 탐색 수행



Shared ResNet



Policy updated after MCPS

AlphaGo Zero: MCPS

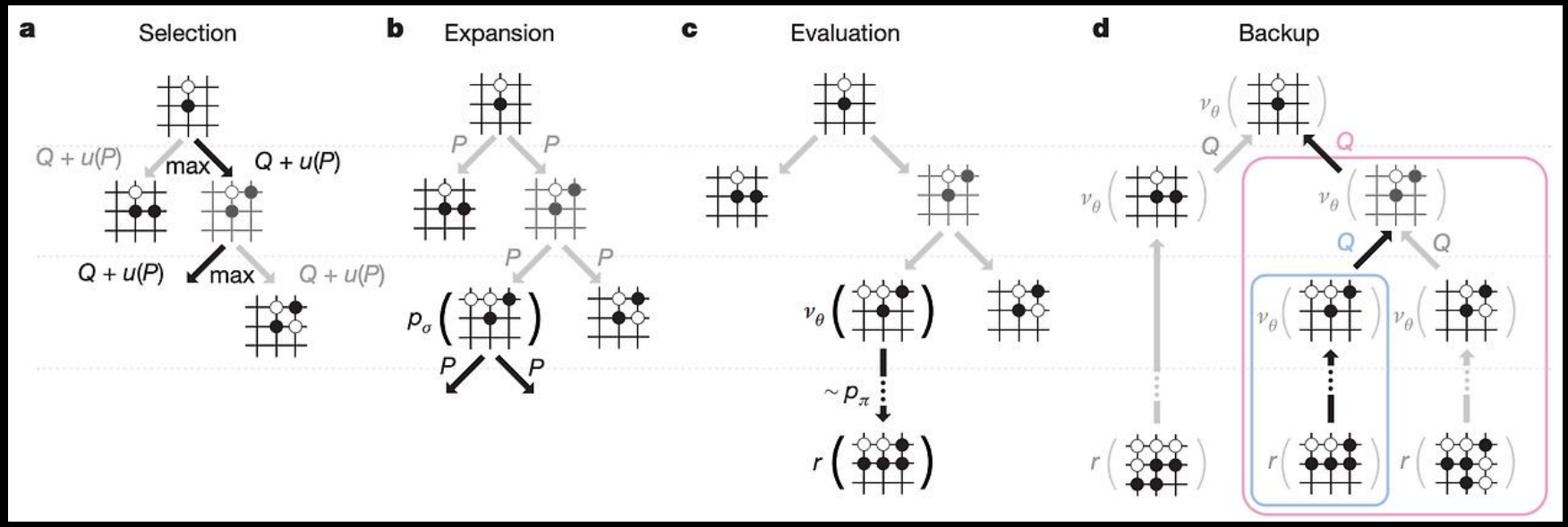
MCPS: 현 상황에서 n번 째까지 (끝까지) 둬보면서
가장 좋은 수를 고르는 방법론

1. Selection (탐색 선택)
현재 노드에서 다음 행동을 고를 때,
- prior $p(a \mid s)$로 "유망한 수"를 우선 탐색하고
- 방문횟수/가치 정보를 함께 고려해서 균형 탐색

2. Expansion (확장)
아직 충분히 확장되지 않은 새 노드에 도달하면
- 신경망으로 그 노드의 p, v를 예측해서 트리에 붙임

3. Backup (역전파)
leaf에서 얻은 value v를 루트까지 되돌려 보내면서
- 각 행동의 평균 가치 Q와 방문수 N를 업데이트

4. Improved policy 생성
루트에서 행동별 방문수 $N(s,a)$를 기반으로
탐색을 반영한 π_i 정책 생성



Problem in A2C: Correlation Issue

Structural Limitation

1. Take action, then get one-step experience (s, a, s', r)
2. Fit Value Function

$$L(\phi) = \frac{1}{2}\sum_i \left\| \hat{V}_\phi^\pi(s_i) - y_i \right\|^2$$

3. Evaluate Advantage Function

$$A^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma \hat{V}_\phi^\pi(s_{t+1}) - \hat{V}_\phi^\pi(s_t)$$

4. Compute Policy Gradient (Actor Update)

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s) \hat{A}^\pi(s, a)$$

5. Update Policy Parameters

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Policy Gradient algorithm is **on-policy algorithm so that we cannot use replay buffer** to solve correlation issue.

(s, a, s', r)

It is generated continuously.

$s_t \rightarrow s_{t+1} \rightarrow s_{t+2} \rightarrow \cdots$

Strong Correlation between previous, current, and next state.

It will break i.i.d. assumption of SGD.

All training results will be collapsed.

Solutions?
Go to next slide

Solutions over Correlation Issue in Actor-Critic

Mini-batch Trajectory Update (A2C)

- Collect multiple rollouts
- Perform minibatch-like updates instead of single-step updates

Parallel Environments (A3C)

- Generate experience from multiple environments simultaneously
- Reduce temporal correlation between samples

PPO / TRPO

- Stabilize policy updates
- Prevent large, destructive policy shifts

GAE (Generalized Advantage Estimation)

- Reduce variance of advantage estimates
- multi-step TD
- exponentially weighted estimation

A3C (Google, 2016)

A3C (Asynchronous Advantage Actor-Critic Algorithm)

Why A3C introduced?

- Sequential Data → Strong Correlation

  $s_t \rightarrow s_{t+1} \rightarrow s_{t+2} \rightarrow \cdots$

- On-policy → Difficulty of Replay Buffer (unlike DQN)

- Slow & Unstable Learning

- Solution:

  "Asynchronous Methods for Deep Reinforcement Learning"

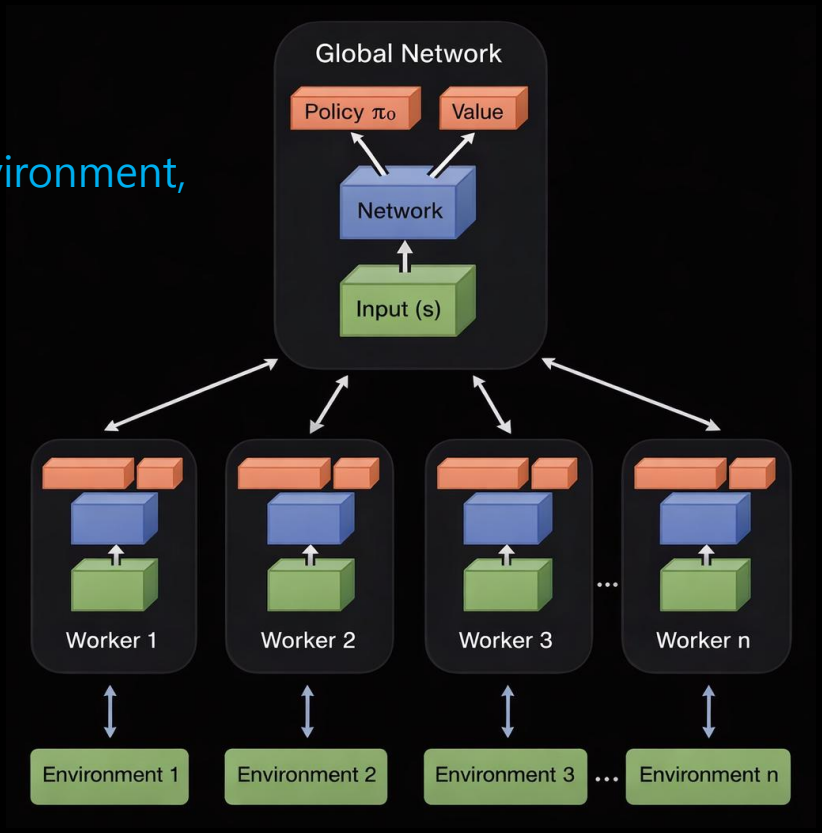
  Mnih et al., Google DeepMind, ICML 2016

  Paper link: https://arxiv.org/abs/1602.01783

A3C – Core Idea

Core Idea:

- Instead of learning sequentially in a single environment, multiple environments are run in parallel.



Each worker:

- Collects rollouts from its own environment
- Computes local gradients
- Asynchronously updates a shared global network

How to Solve the Strong Correlation:

- Experiences are collected from different environments
- Each environment has a different state distribution
- Each worker generates independent trajectories

✓ Closer to the i.i.d. Assumption

✓ No Need of Replay Buffer

A3C – Characteristics

Applying $n - step$ bootstrapping instead of 1-step

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \cdots + \gamma^n V(s_{t+n})$$

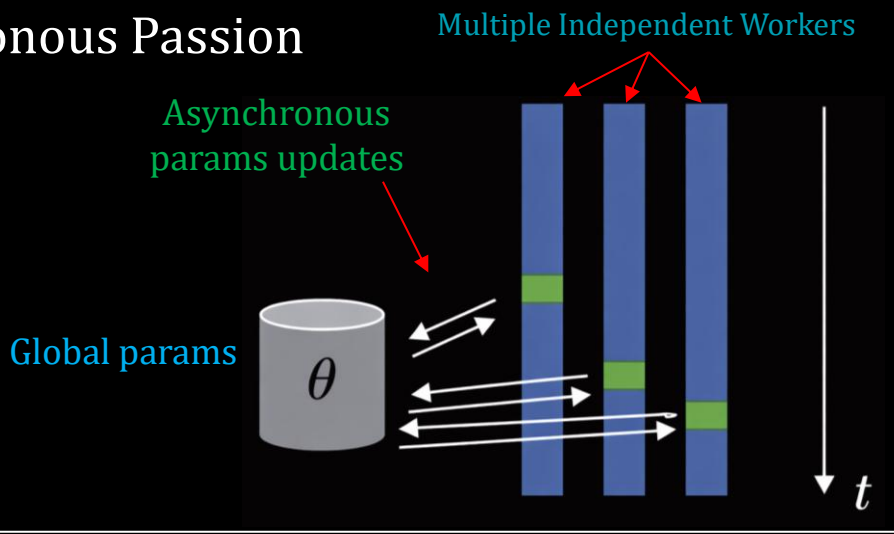
각자 gradient 계산

global params 직접 업데이트

→ Lock-free async optimization!

당시에는 혁신적, 그러나 GPU 시대에서 배치 업데이트에서 결국 문제 발생

Asynchronous Passion



Background for the Emergence of Synchronous A2C and PPO

Pros	Cons
✓ More stable ✓ Low Variance than Monte Carlo ✓ Simple Implementation ✓ Ease of Parallelization ✓ Able to Control Bias-variance Trade-off	✓ Multi-core CPU Architecture ✓ Inefficient GPU Batch Process due to Asynchronous Passion

Synchronous A2C (OpenAI, 2017)

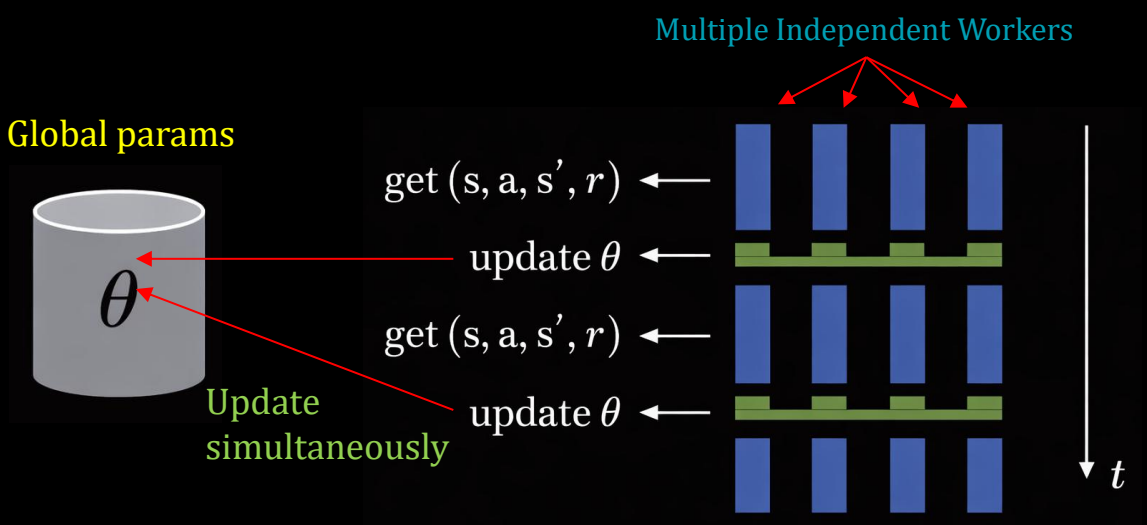
OpenAI, Aug. 18th, 2017.

Blog link: https://openai.com/index/openai-baselines-acktr-a2c

Core Idea:

- ✓ All workers first collect rollouts.
- ✓ Then they perform a synchronized update together at once.

collect batch → update → repeat



Advantages:

- ✓ Batch Update
- ✓ Efficient Matrix Ops
- ✓ GPU-friendly

Much more popular in practical fields!

A3C vs. A2C

A2C keeps the parallel exploration of A3C

but replaces asynchronous updates

with synchronized batch updates

for improved stability and GPU efficiency.

Asynchronous Advantage Actor-Critic 등장

➔ GPU 활용과 안정성 문제 발생

➔ **Synchronous version** 등장

➔ 이걸 줄여서 **A2C (Advantage Actor-Critic)** 라고 부르게 됨

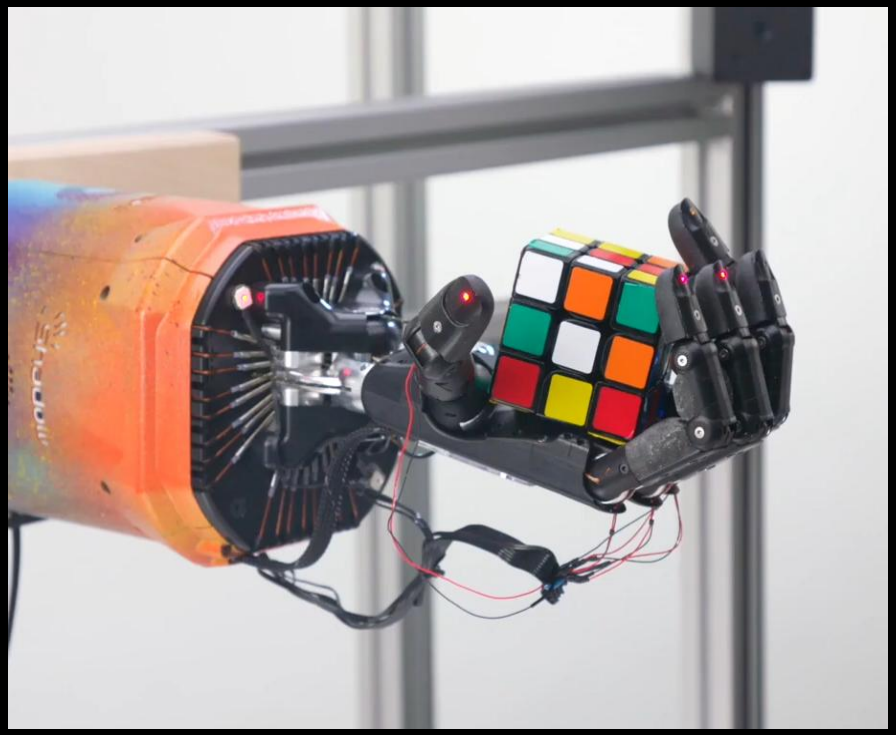
➔ 일반적으로 A2C라고 하면 "Synchronous A2C"를 의미함

	A3C	A2C
Update	Asynchronous	Synchronous
Stability	상대적으로 noisy	더 안정적
GPU usage	낮음	높음
구현	간단	더 체계적

Applications

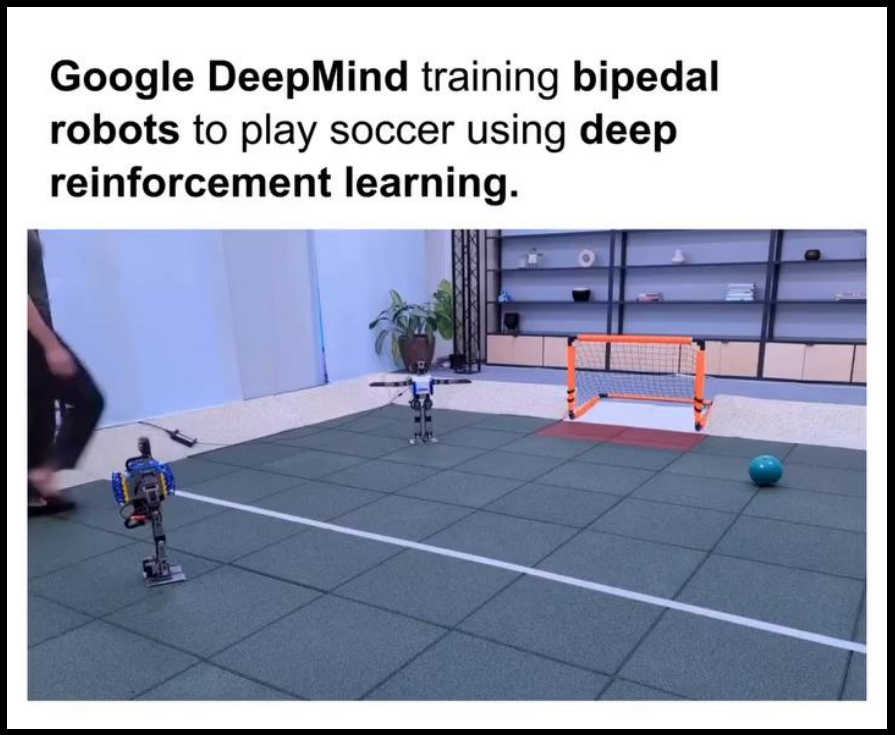
Solving Rubik's Cube with a Robot Hand: Uncut (OpenAI)

https://youtu.be/kVmp0uGtShk



Training bipedal robots to play soccer using deep reinforce learning (Google DeepMind)

https://www.youtube.com/shorts/xfs0qtcaNFM



Practical Concerns in A2C

"The devil is hidden in the details"

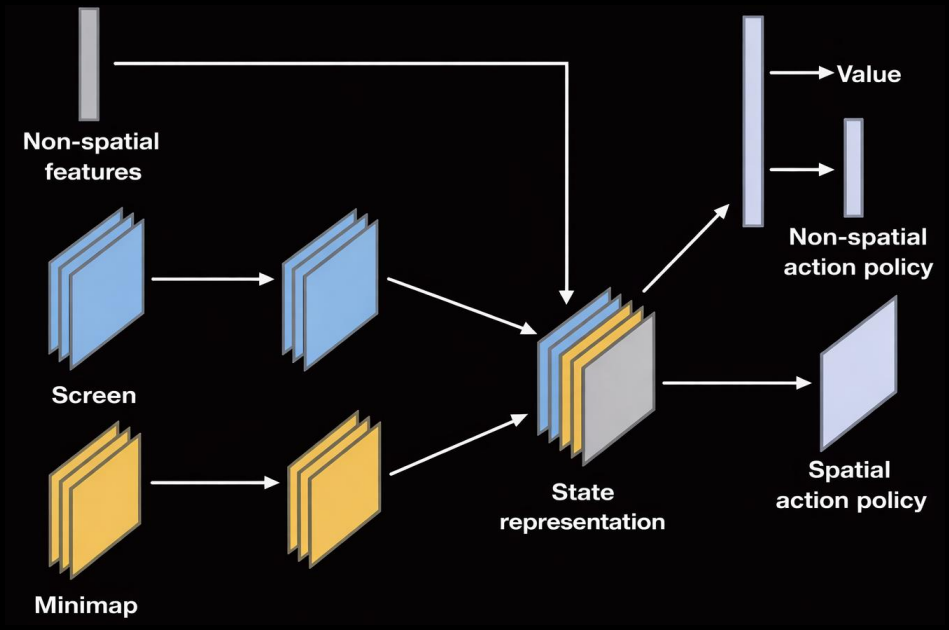
알고리즘 자체보다
실제 구현과 계산 자원 문제가 훨씬 어렵다.

A3C / A2C:

- ✓ Each worker must save n-step trajectory (history mgnt.)
- ✓ During forward step, intermediate state cache required (cache impl.)
- ✓ GPU memory ↑ + RAM usage ↑ (memory mgnt.)

⋮

⋮



수고하셨습니다 ..^^..