Proximal Policy Optimization (PPO)

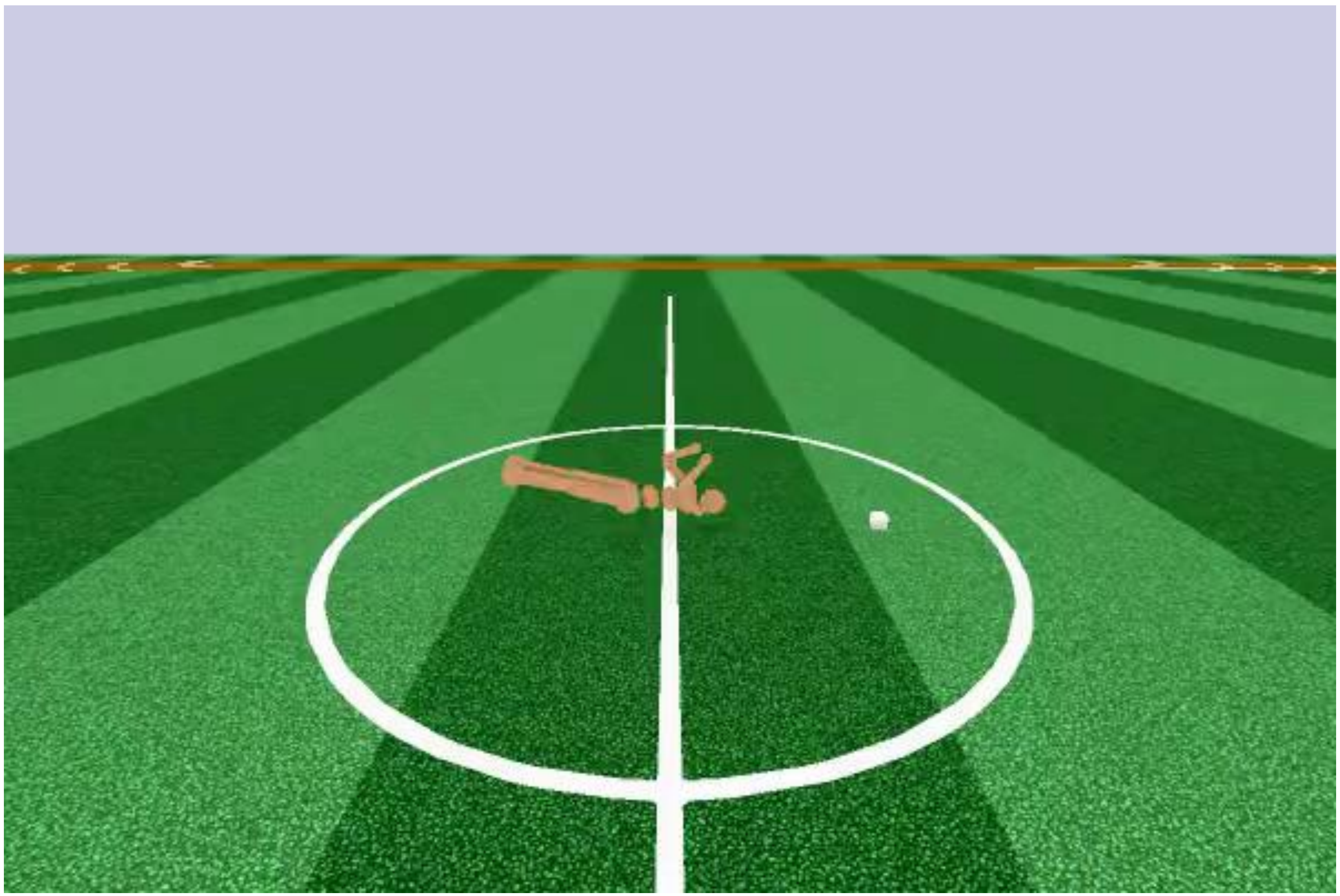
default reinforcement learning algorithm at OpenAI

DeepMind

https://youtu.be/gn4nRCC9TwQ

OpenAI

https://blog.openai.com/openai-baselines-ppo/



Policy Gradient (Review)

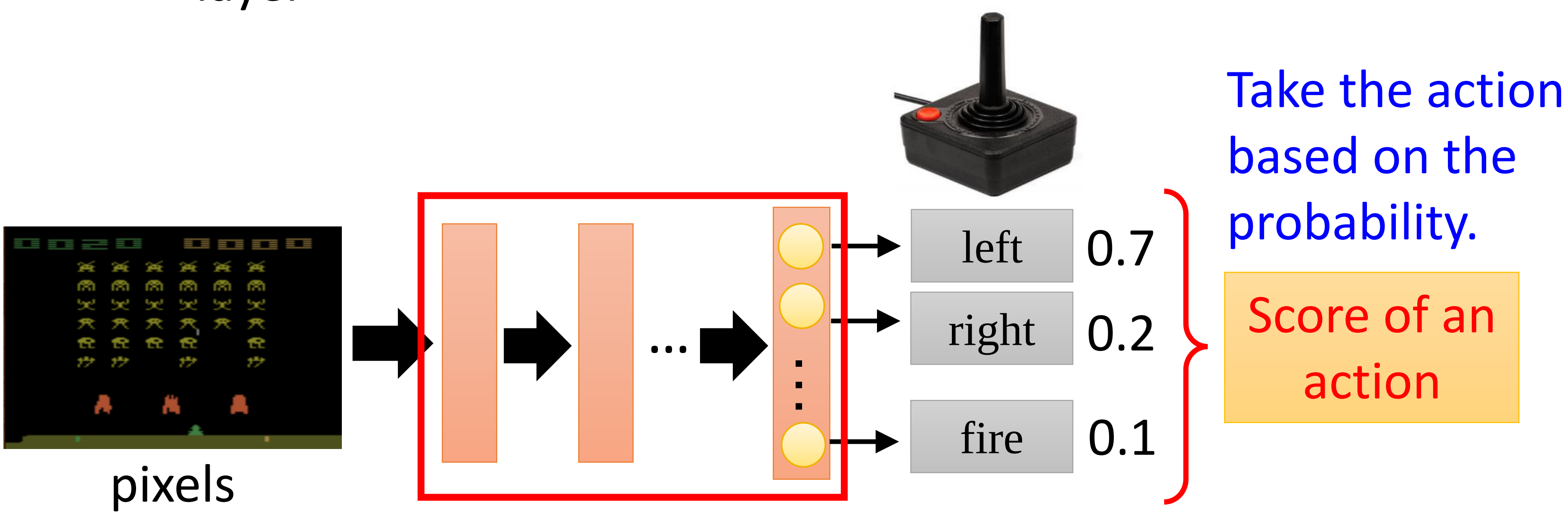
Basic Components

You cannot control (Env, Reward Function)

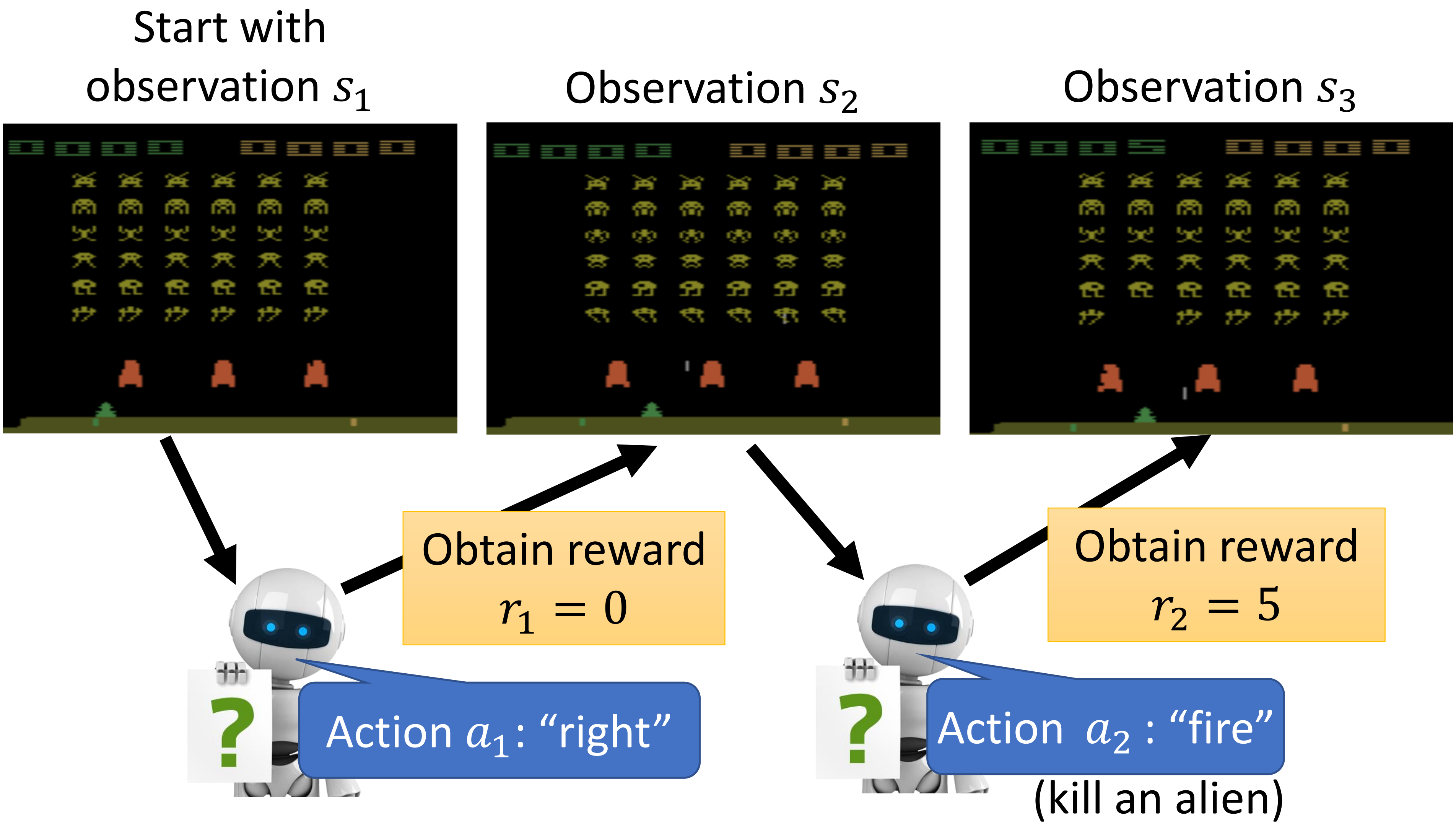
	Actor	Env	Reward Function
Video Game			Get 20 scores when killing a monster
Go	AlphaGo Google DeepMind		The rule of GO

Policy of Actor

- Policy π is a network with parameter θ
  - Input: the observation of machine represented as a vector or a matrix
  - Output: each action corresponds to a neuron in output layer



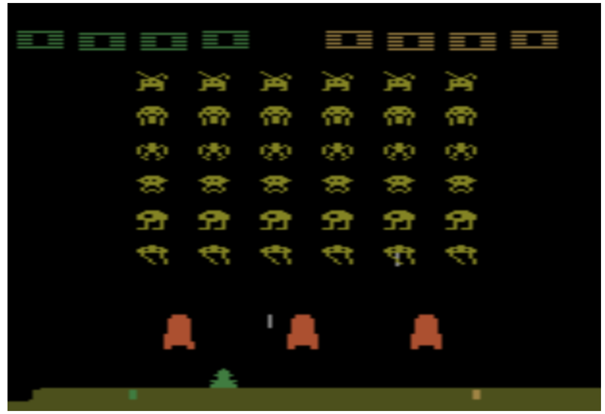
Example: Playing Video Game



Example: Playing Video Game

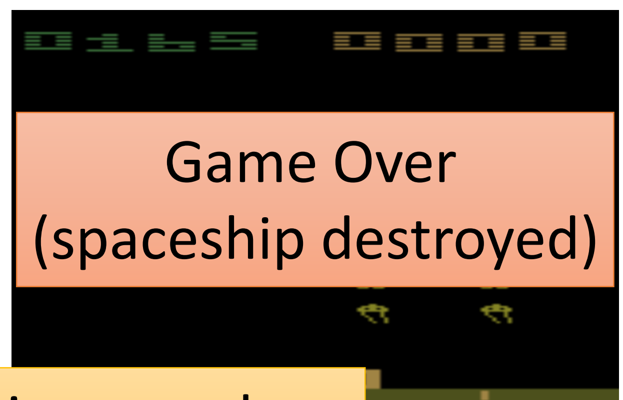
Start with observation s_1

Observation s_2



Observation s_3

After many turns



Game Over (spaceship destroyed)

Obtain reward r_T

Action a_T

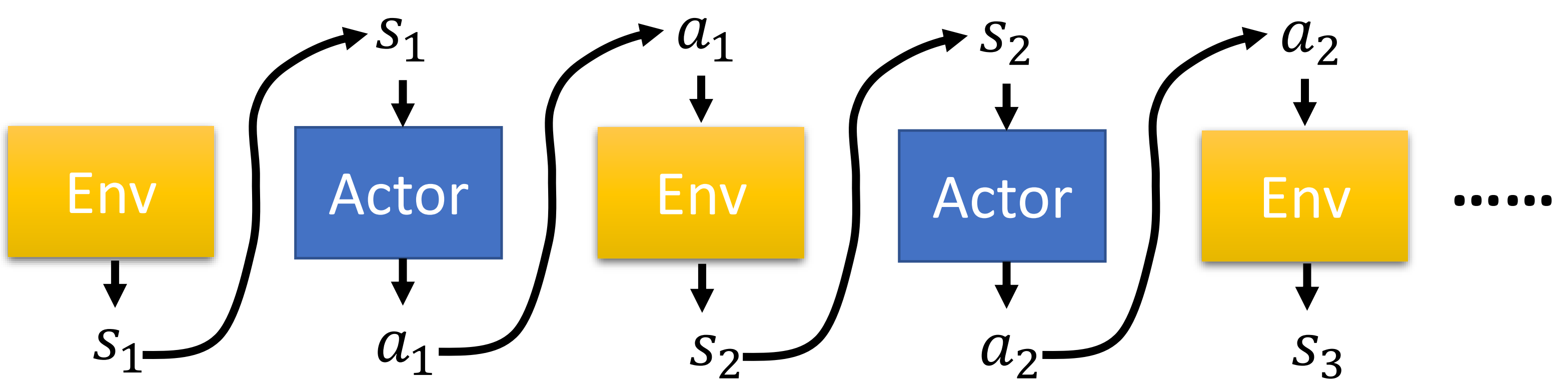
This is an ***episode***.

Total reward:

$$R = \sum_{t=1}^{T} r_t$$

We want the total reward be maximized.

Actor, Environment, Reward



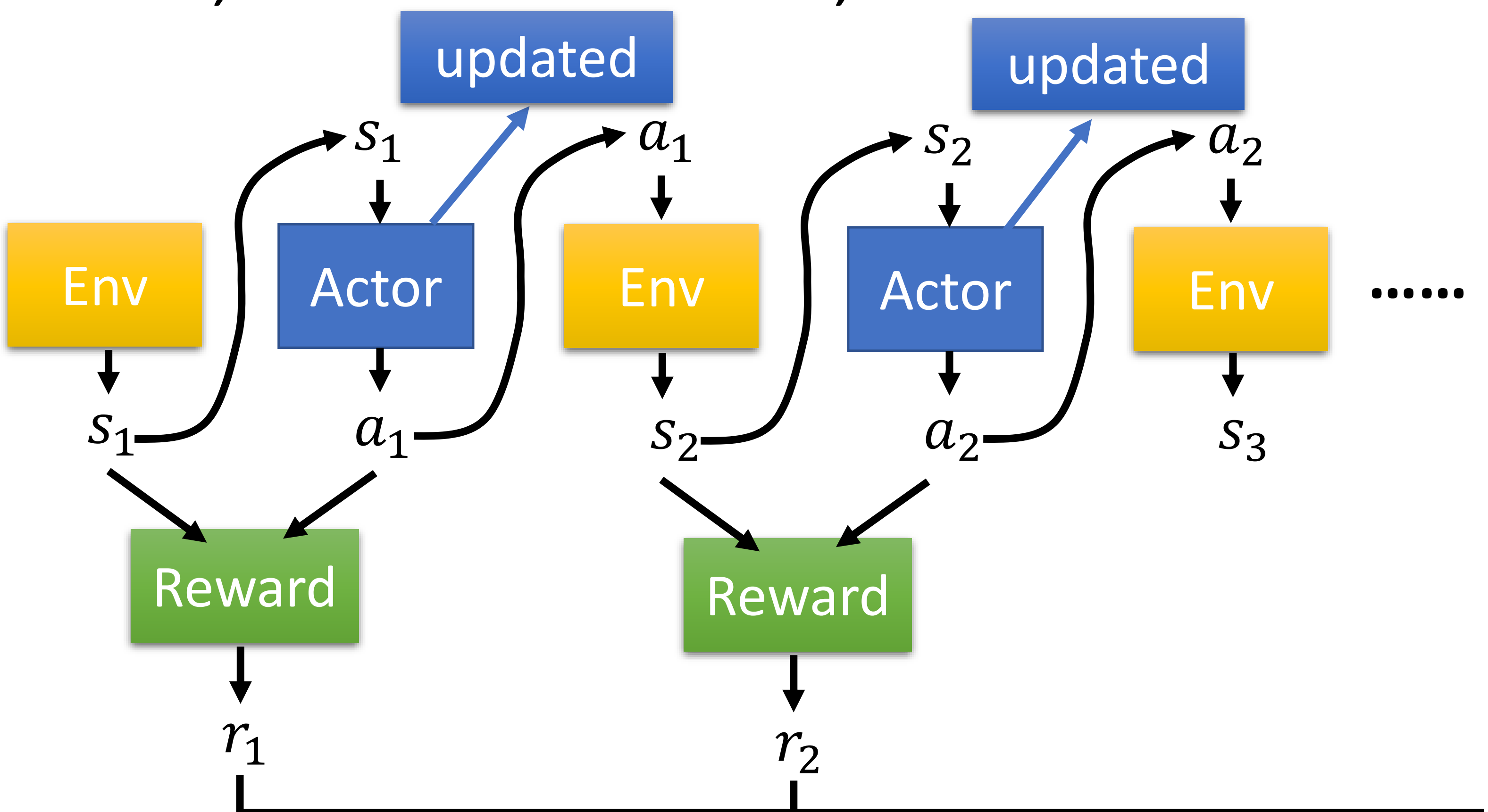
Trajectory $\tau = \{s_1, a_1, s_2, a_2, \cdots, s_T, a_T\}$

$$p_\theta(\tau)$$

$$= p(s_1)p_\theta(a_1|s_1)p(s_2|s_1, a_1)p_\theta(a_2|s_2)p(s_3|s_2, a_2)\cdots$$

$$= p(s_1)\prod_{t=1}^{T} p_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)$$

Actor, Environment, Reward



Expected Reward

$$\bar{R}_\theta = \sum_{\tau} R(\tau) p_\theta(\tau) = E_{\tau \sim p_\theta(\tau)}[R(\tau)]$$

$$R(\tau) = \sum_{t=1}^{T} r_t$$

Policy Gradient

$$\bar{R}_\theta = \sum_\tau R(\tau) p_\theta(\tau) \qquad \nabla \bar{R}_\theta = ?$$

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau) \nabla p_\theta(\tau) = \sum_\tau R(\tau) p_\theta(\tau) \frac{\nabla p_\theta(\tau)}{p_\theta(\tau)}$$

$R(\tau)$ do not have to be differentiable

It can even be a black box.

$$\nabla f(x) = f(x) \nabla log f(x)$$

$$= \sum_\tau R(\tau) p_\theta(\tau) \nabla log p_\theta(\tau)$$

$$= E_{\tau \sim p_\theta(\tau)}[R(\tau) \nabla log p_\theta(\tau)] \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \nabla log p_\theta(\tau^n)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p_\theta(a_t^n | s_t^n)$$

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau) \nabla log p_\theta(\tau)]$$

Policy Gradient

Given policy π_θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

(s_2^1, a_2^1) $R(\tau^1)$

⋮ ⋮

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

⋮ ⋮

only used once

Update Model

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p_\theta(a_t^n | s_t^n)$$

Data Collection

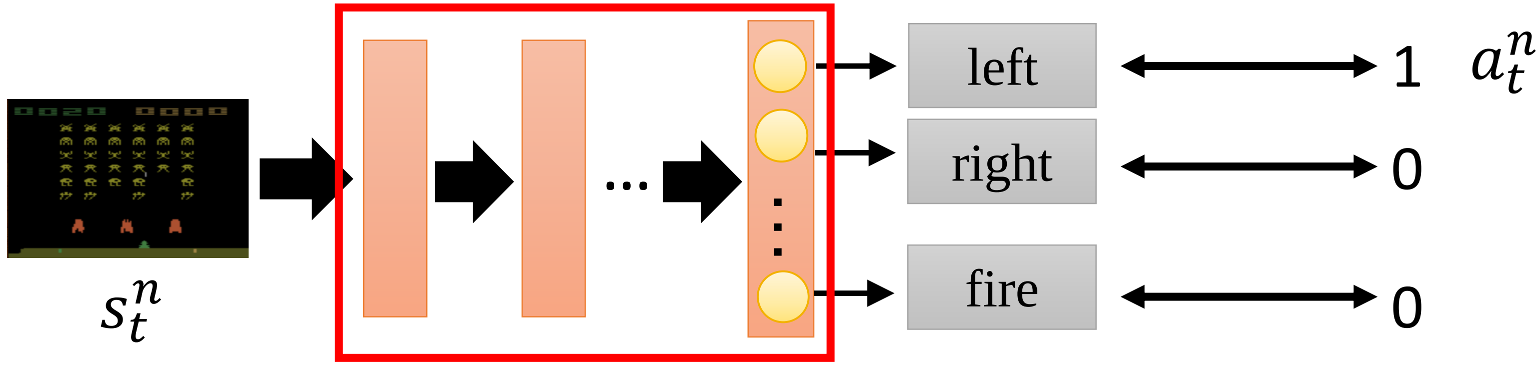
$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

Implementation

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p_\theta(a_t^n | s_t^n)$$

$s_t^n \quad a_t^n \quad R(\tau^n)$

Consider as classification problem



$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} log p_\theta(a_t^n | s_t^n) \rightarrow \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \nabla log p_\theta(a_t^n | s_t^n)$$

TF, pyTorch …

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \underline{R(\tau^n)} log p_\theta(a_t^n | s_t^n) \rightarrow \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \underline{R(\tau^n)} \nabla log p_\theta(a_t^n | s_t^n)$$

Tip 1: Add a Baseline

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

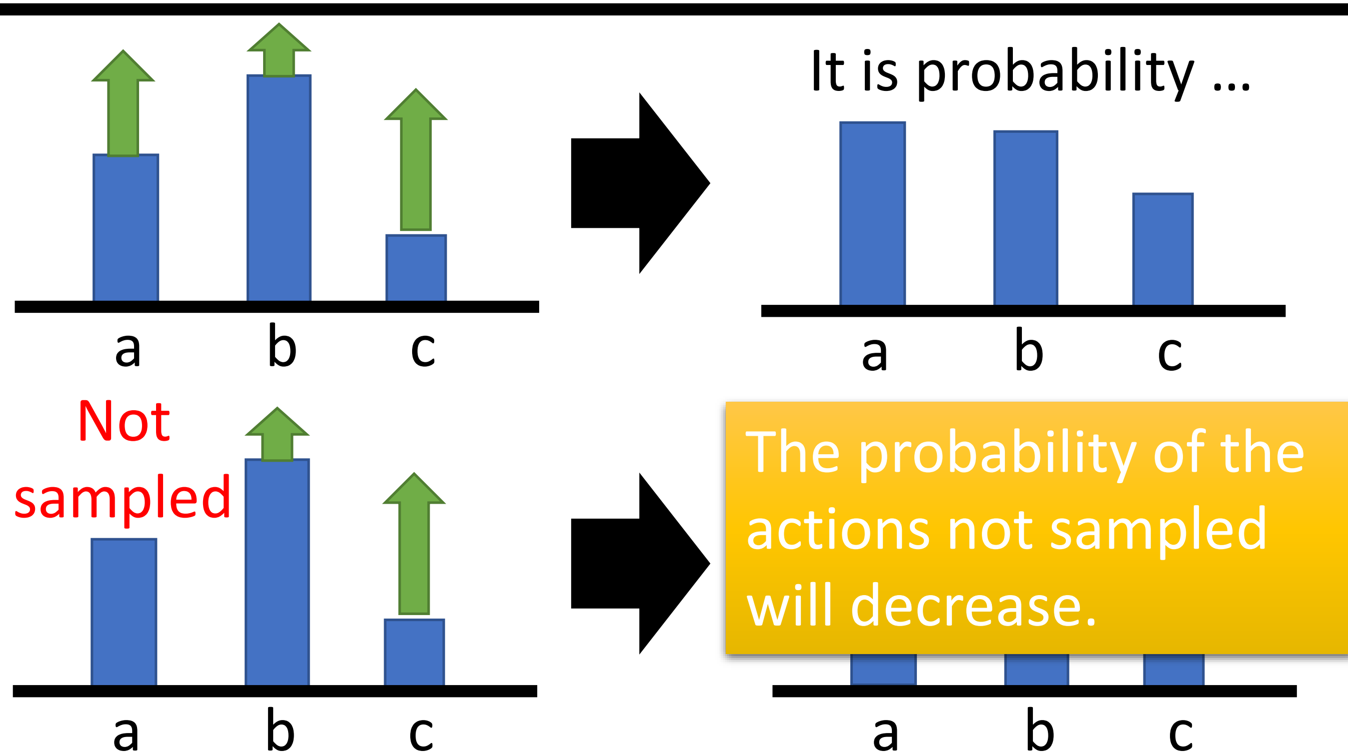
It is possible that $R(\tau^n)$ is always positive.

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) - b) \nabla log p_\theta(a_t^n | s_t^n) \qquad b \approx E[R(\tau)]$$

Ideal case

Sampling

......



Tip 2: Assign Suitable Credit

$\times 3$	$\times -2$	$\times -2$
(s_a, a_1)	(s_b, a_2)	(s_c, a_3)
+5	+0	-2

$$R = +3$$

$\times -7$	$\times -2$	$\times -2$
(s_a, a_2)	(s_b, a_2)	(s_c, a_3)
-5	+0	-2

$$R = -7$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (\sout{R(\tau^n)} - b) \nabla log p_\theta(a_t^n | s_t^n)$$

$$\rightarrow \sum_{t'=t}^{T_n} r_{t'}^n$$

Tip 2: Assign Suitable Credit

Advantage Function $A^{\theta}(s_t, a_t)$

How good it is if we take a_t other than other actions at s_t.

Estimated by "*critic*" (later)

Can be state-dependent

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left(R(\tau^n) - b \right) \nabla \log p_{\theta}(a_t^n | s_t^n)$$

$$\sum_{t'=t}^{T_n} r_{t'}^n \rightarrow \sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n$$

Add discount factor $\gamma < 1$

From on-policy to off-policy

Using the experience more than once

On-policy v.s. Off-policy

- On-policy: The agent learned and the agent interacting with the environment is the same.
- Off-policy: The agent learned and the agent interacting with the environment is different.

阿光下棋

佐為下棋、阿光在旁邊看

On-policy → Off-policy

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)]$$

- Use π_θ to collect data. When θ is updated, we have to sample training data again.
- Goal: Using the sample from $\pi_{\theta'}$ to train θ. θ' is fixed, so we can re-use the sample data.

Importance Sampling

$$E_{x\sim p}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x^i)$$

x^i is sampled from $p(x)$

We only have x^i sampled from $q(x)$

$$= \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

Importance weight

Issue of Importance Sampling

$$E_{x\sim p}[f(x)] = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

$$\mathrm{V}ar_{x\sim p}[f(x)] \qquad \mathrm{V}ar_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

$$VAR[X]$$
$$= E[X^2] - (E[X])^2$$

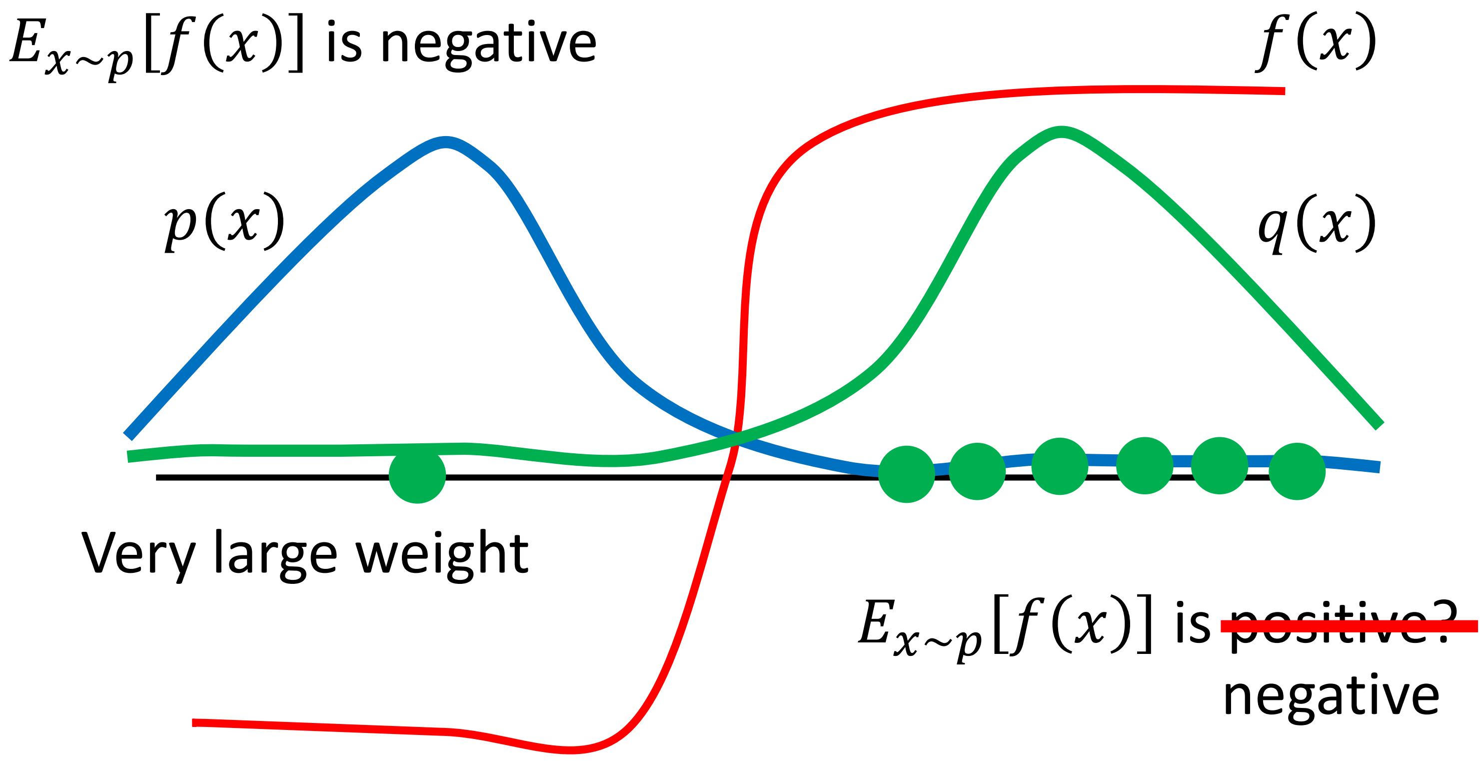
$$\mathrm{V}ar_{x\sim p}[f(x)] = E_{x\sim p}[f(x)^2] - \left(E_{x\sim p}[f(x)]\right)^2$$

$$\mathrm{V}ar_{x\sim q}[f(x)\frac{p(x)}{q(x)}] = E_{x\sim q}\left[\left(f(x)\frac{p(x)}{q(x)}\right)^2\right] - \left(E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]\right)^2$$

$$= E_{x\sim p}\left[f(x)^2\boxed{\frac{p(x)}{q(x)}}\right] - \left(E_{x\sim p}[f(x)]\right)^2$$

Issue of Importance Sampling

$$E_{x\sim p}[f(x)] = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$



On-policy → Off-policy

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)]$$

- Use π_θ to collect data. When θ is updated, we have to sample training data again.
- Goal: Using the sample from $\pi_{\theta'}$ to train θ. θ' is fixed, so we can re-use the sample data.

$$\nabla \bar{R}_\theta = E_{\tau \sim p_{\theta'}(\tau)}\left[\frac{p_\theta(\tau)}{p_{\theta'}(\tau)} R(\tau)\nabla log p_\theta(\tau)\right]$$

- Sample the data from θ'.
- Use the data to train θ many times.

Importance Sampling

$$E_{x \sim p}[f(x)] = E_{x \sim q}[f(x)\frac{p(x)}{q(x)}]$$

On-policy → Off-policy

Gradient for update

$\nabla f(x) = f(x) \nabla log f(x)$

$$= E_{(s_t,a_t)\sim\pi_\theta}[A^\theta(s_t,a_t)\nabla log p_\theta(a_t^n|s_t^n)]$$

$A^{\theta'}(s_t,a_t)$ This term is from sampled data.

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{P_\theta(s_t,a_t)}{P_{\theta'}(s_t,a_t)} A^\theta(s_t,a_t)\nabla log p_\theta(a_t^n|s_t^n)]$$

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} \frac{p_\theta(s_t)}{p_{\theta'}(s_t)} A^\theta(s_t,a_t)\nabla log p_\theta(a_t^n|s_t^n)]$$

$$J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t,a_t)\right]$$

When to stop?

Add Constraint

穩紮穩打，步步為營

PPO / TRPO

θ cannot be very different from θ'

Constraint on behavior not parameters

Proximal Policy Optimization (PPO)

$$\nabla f(x) = f(x) \nabla log f(x)$$

$$J_{PPO}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta KL(\theta, \theta')$$

$$J^{\theta'}(\theta) = E_{(s_t, a_t) \sim \pi_{\theta'}} \left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t) \right]$$

TRPO (Trust Region Policy Optimization)

$$J_{TRPO}^{\theta'}(\theta) = E_{(s_t, a_t) \sim \pi_{\theta'}} \left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t) \right]$$

$$KL(\theta, \theta') < \delta$$

PPO algorithm

$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t,a_t)$$

- Initial policy parameters θ^0
- In each iteration
  - Using θ^k to interact with the environment to collect $\{s_t, a_t\}$ and compute advantage $A^{\theta^k}(s_t, a_t)$
  - Find θ optimizing $J_{PPO}(\theta)$

$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL\left(\theta, \theta^k\right)$$

Update parameters several times

- If $KL\left(\theta, \theta^k\right) > KL_{max}$, increase β
- If $KL\left(\theta, \theta^k\right) < KL_{min}$, decrease β

Adaptive KL Penalty

PPO algorithm

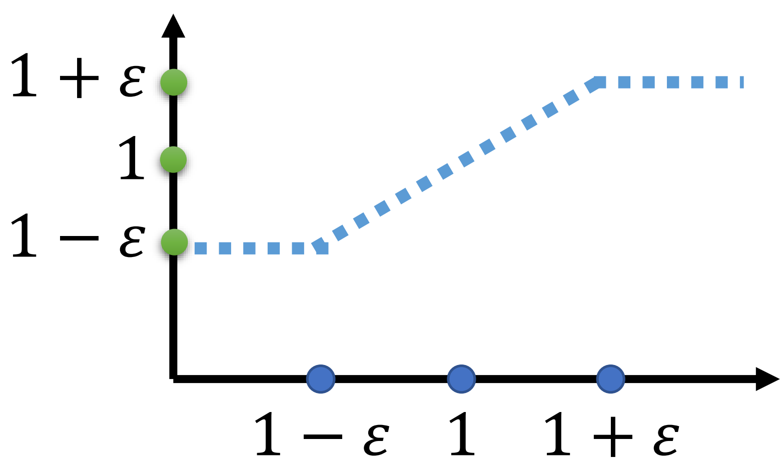
$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t)$$

PPO2 algorithm

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)}$$

$$clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}(s_t, a_t)$$



$$\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}$$

PPO algorithm

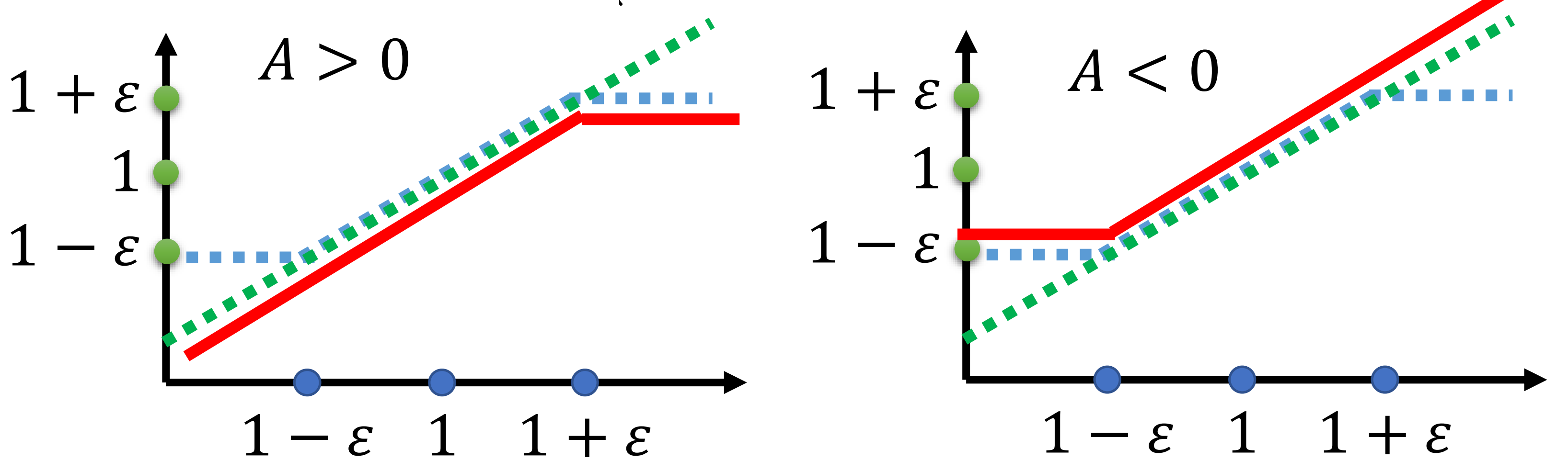
$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t)$$

PPO2 algorithm

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t), \right.$$

$$\left. clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon \right) A^{\theta^k}(s_t, a_t) \right)$$



https://arxiv.org/abs/1707.06347

Experimental Results

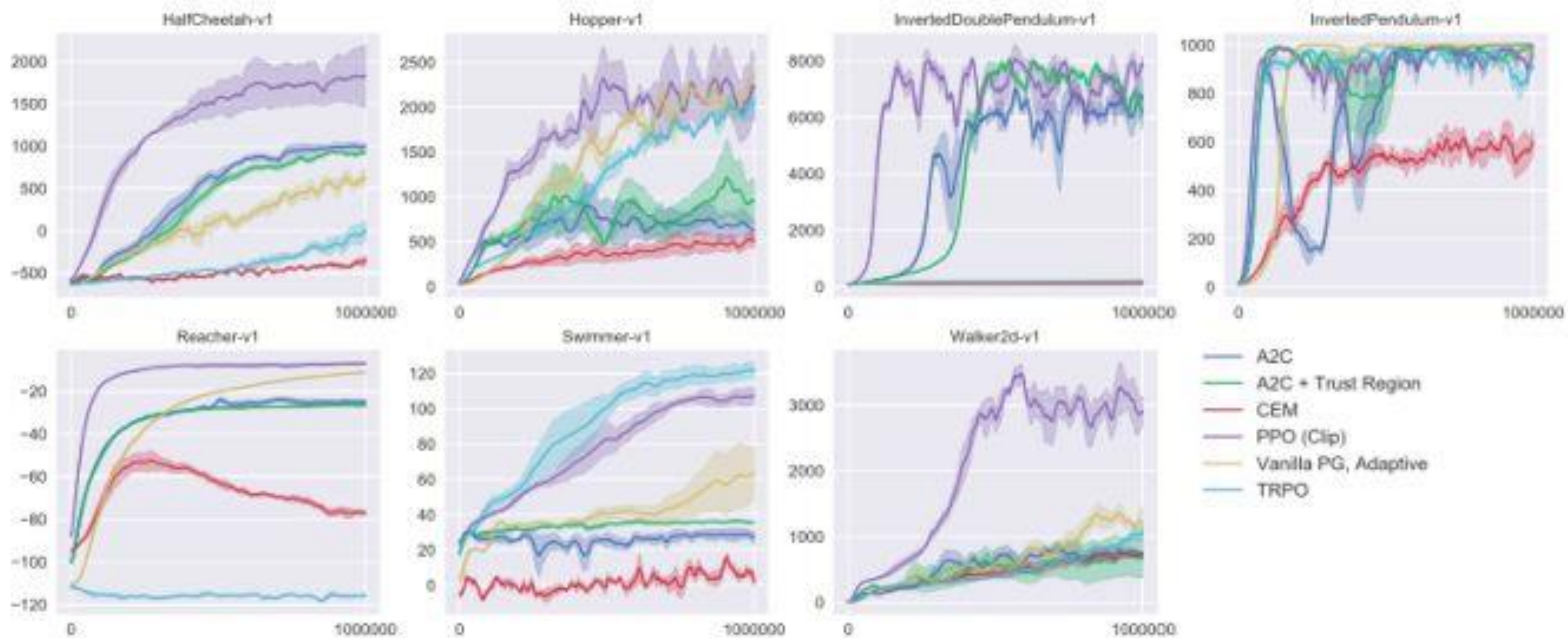


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.