Policy Gradient (continue)

Recap: the REINFORCE Algorithm

$$\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$$

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\ldots$$

Recap: the REINFORCE Algorithm

$$\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$$

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\ldots$$

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \underbrace{\left[\sum_{h=0}^{H-1} r(s_h, a_h) \right]}_{R(\tau)}$$

Recap: the REINFORCE Algorithm

$$\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$$

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\ldots$$

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \underbrace{\left[\sum_{h=0}^{H-1} r(s_h, a_h) \right]}_{R(\tau)}$$

$$\nabla_\theta J(\pi_\theta) \,|_{\theta=\theta_0} := \mathbb{E}_{\tau \sim \rho_{\theta_0}(\tau)} \left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_{\theta_0}(a_h \mid s_h) \right) R(\tau) \right]$$

Recap: the REINFORCE Algorithm

$$\nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_0} := \mathbb{E}_{\tau\sim\rho_{\theta_0}(\tau)}\left[\left(\sum_{h=0}^{H-1}\nabla_\theta \ln \pi_{\theta_0}(a_h\,|\,s_h)\right)R(\tau)\right]$$

How to get an unbiased estimate of the PG?

Recap: the REINFORCE Algorithm

$$\nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_0} := \mathbb{E}_{\tau\sim\rho_{\theta_0}(\tau)}\left[\left(\sum_{h=0}^{H-1}\nabla_\theta \ln \pi_{\theta_0}(a_h\,|\,s_h)\right)R(\tau)\right]$$

How to get an unbiased estimate of the PG?

$$\tau \sim \rho_{\theta_0}$$

Recap: the REINFORCE Algorithm

$$\nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_0} := \mathbb{E}_{\tau\sim\rho_{\theta_0}(\tau)}\left[\left(\sum_{h=0}^{H-1}\nabla_\theta \ln \pi_{\theta_0}(a_h\,|\,s_h)\right)R(\tau)\right]$$

How to get an unbiased estimate of the PG?

$$\tau \sim \rho_{\theta_0}$$

$$g := \sum_{h=0}^{H-1}\left[\nabla \ln \pi_{\theta_0}(a_h\,|\,s_h)R(\tau)\right]$$

Recap: the REINFORCE Algorithm

$$\nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_0} := \mathbb{E}_{\tau\sim\rho_{\theta_0}(\tau)}\left[\left(\sum_{h=0}^{H-1}\nabla_\theta \ln \pi_{\theta_0}(a_h\,|\,s_h)\right)R(\tau)\right]$$

How to get an unbiased estimate of the PG?

$$\tau \sim \rho_{\theta_0}$$

$$g := \sum_{h=0}^{H-1}\left[\nabla \ln \pi_{\theta_0}(a_h\,|\,s_h)R(\tau)\right]$$

We have: $\mathbb{E}[g] = \nabla_\theta J(\pi_{\theta_0})$

Recap: the REINFORCE Algorithm

$$\nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_0} := \mathbb{E}_{\tau\sim\rho_{\theta_0}(\tau)}\left[\left(\sum_{h=0}^{H-1}\nabla_\theta \ln \pi_{\theta_0}(a_h\,|\,s_h)\right)R(\tau)\right]$$

How to get an unbiased estimate of the PG?

$$\tau \sim \rho_{\theta_0}$$

$$g := \sum_{h=0}^{H-1}\left[\nabla \ln \pi_{\theta_0}(a_h\,|\,s_h)R(\tau)\right]$$

We have: $\mathbb{E}[g] = \nabla_\theta J(\pi_{\theta_0})$

This formulation has large variance, i.e.,
$\mathbb{E}\left[\|g - \nabla_\theta J(\pi_{\theta_0})\|_2^2\right]$
could be as large as H^3

Today’s Question:

How to Make Policy Gradient really useful in practice

Outline:

1. A $Q(s, a)$ based Policy Gradient and Variance reduction

2. Proximal Policy Optimization (it trains ChatGPT!)

3. Reset to address the exploration challenge

Value / Q function

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,|\, s_h = s, a_t \sim \pi\right]$$

$$Q_h^\pi(s, a) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,|\, s_h = s, a_h = a, a_t \sim \pi\right]$$

$$= r(s, a) + \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V_{h+1}^\pi(s')$$

Policy Gradient w/ Q_h^π

$$\nabla_\theta J(\pi_\theta) := \sum_{h=0}^{H-1} \mathbb{E}_{s,a\sim d_h^{\pi_\theta}} \left[\underline{\nabla_\theta \ln \pi_\theta(a_h \mid s_h) \cdot Q_h^{\pi_\theta}(s_h, a_h)} \right]$$

State-action
distribution at h

Adjust θ s.t. policy increases
(decreases) prob of a with high (low)
expected reward-to-go

Variance reduction via a Baseline

$$\nabla_\theta J(\pi_\theta) := \sum_{h=0}^{H-1} \mathbb{E}_{s,a\sim d_h^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a_h \mid s_h) \cdot \left(Q_h^{\pi_\theta}(s_h, a_h) - b(s_h) \right) \right]$$

Baseline: as long as it is action-indepenent, it does not affect the gradient

Variance reduction via a Baseline

$$\nabla_\theta J(\pi_\theta) := \sum_{h=0}^{H-1} \mathbb{E}_{s,a\sim d_h^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a_h \,|\, s_h) \cdot \left(Q_h^{\pi_\theta}(s_h, a_h) - b(s_h) \right) \right]$$

Baseline: as long as it is action-indepenent, it does not affect the gradient

Just need to show $\mathbb{E}_{s,a\sim\pi_\theta(.|s)} \nabla \ln \pi_\theta(a \,|\, s) \cdot b(s) = 0$:

Value function as a baseline

$$\nabla_\theta J(\pi_\theta) := \sum_{h=0}^{H-1} \mathbb{E}_{s,a\sim d_h^{\pi_\theta}}\left[\nabla_\theta \ln \pi_\theta(a_h \mid s_h)\cdot\left(\underline{Q_h^{\pi_\theta}(s_h,a_h) - V_h^{\pi_\theta}(s_h)}\right)\right]$$

This is called **Advantage** function:

$$A_h^{\pi_\theta}(s,a) = Q_h^{\pi_\theta}(s,a) - V_h^{\pi_\theta}(s)$$

Value function as a baseline

$$\nabla_\theta J(\pi_\theta) := \sum_{h=0}^{H-1} \mathbb{E}_{s,a\sim d_h^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a_h | s_h) \cdot \left(Q_h^{\pi_\theta}(s_h, a_h) - V_h^{\pi_\theta}(s_h) \right) \right]$$

This is called **Advantage** function:

$$A_h^{\pi_\theta}(s, a) = Q_h^{\pi_\theta}(s, a) - V_h^{\pi_\theta}(s)$$

1. Advantage can be as small as a constant (e.g., this is the condition where DAgger works better than BC)

Value function as a baseline

$$\nabla_\theta J(\pi_\theta) := \sum_{h=0}^{H-1} \mathbb{E}_{s,a\sim d_h^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a_h \mid s_h) \cdot \left(\underline{Q_h^{\pi_\theta}(s_h, a_h) - V_h^{\pi_\theta}(s_h)} \right) \right]$$

This is called **Advantage** function:

$$A_h^{\pi_\theta}(s, a) = Q_h^{\pi_\theta}(s, a) - V_h^{\pi_\theta}(s)$$

1. Advantage can be as small as a constant (e.g., this is the condition where DAgger works better than BC)

2. V is not the theoretically optimal baseline, but is used almost in every practical PG alg/implementation

Outline:

1. A $Q(s, a)$ based Policy Gradient and Variance reduction

2. Proximal Policy Optimization (it trains ChatGPT!)

3. How to address the exploration challenge

Example application of PPO:

Train a robot to “run” forward as fast as possible:

State: joint angles, center of mass, velocity, etc

Action: torques on joints

Reward: distance of moving forward between two steps

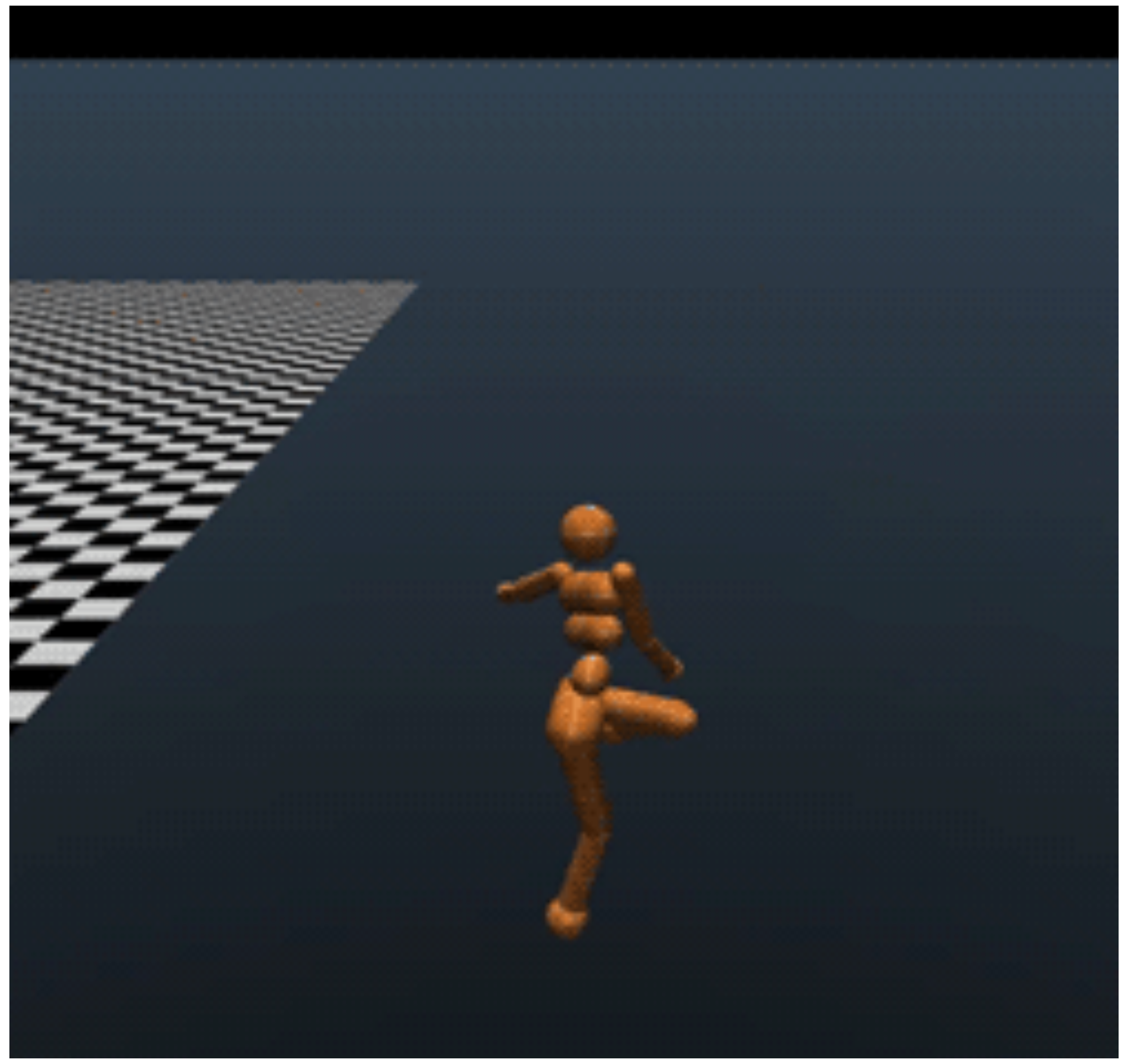
Example application of PPO:

Train a robot to “run” forward as fast as possible:

State: joint angles, center of mass, velocity, etc

Action: torques on joints

Reward: distance of moving forward between two steps



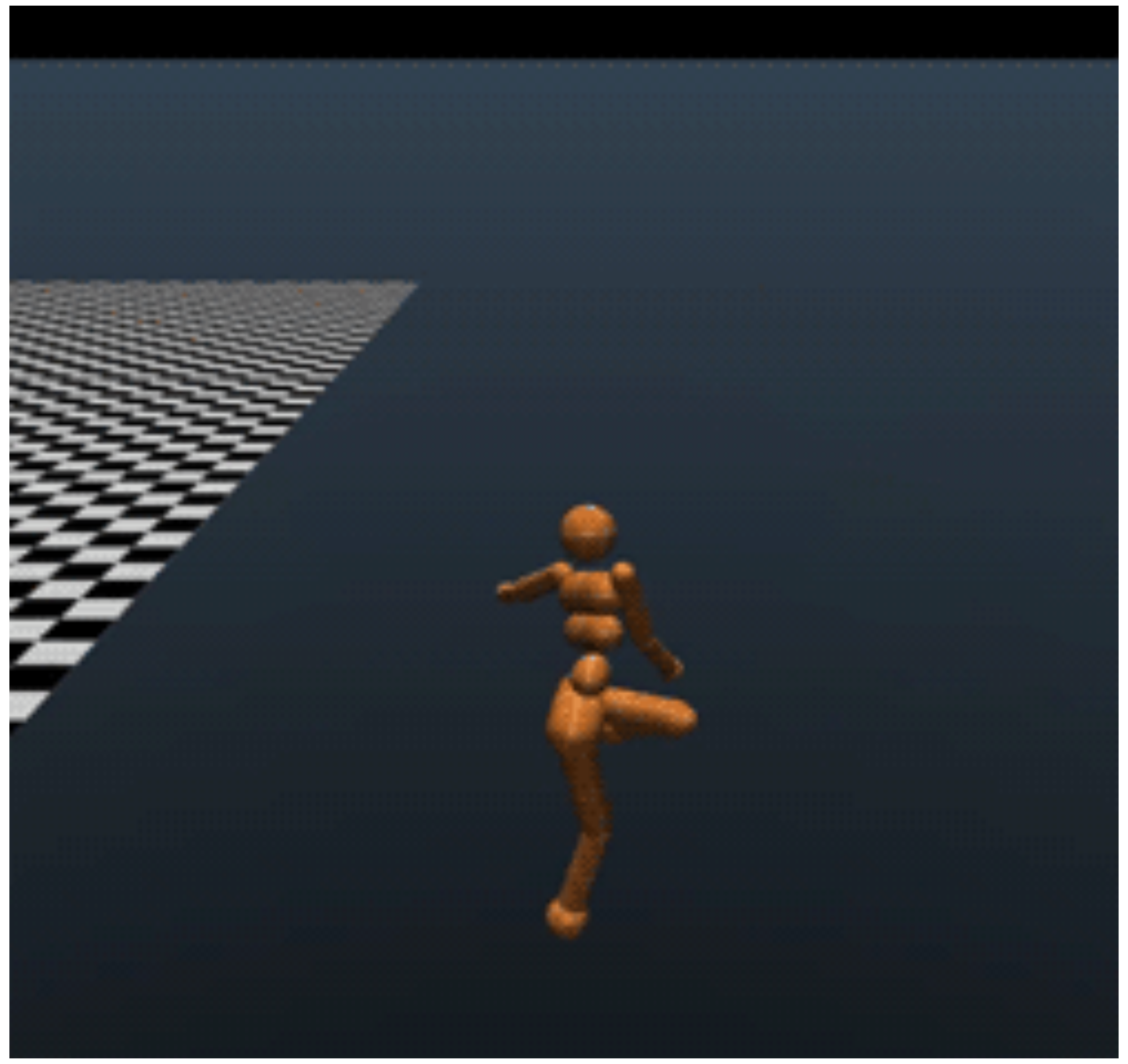
Example application of PPO:

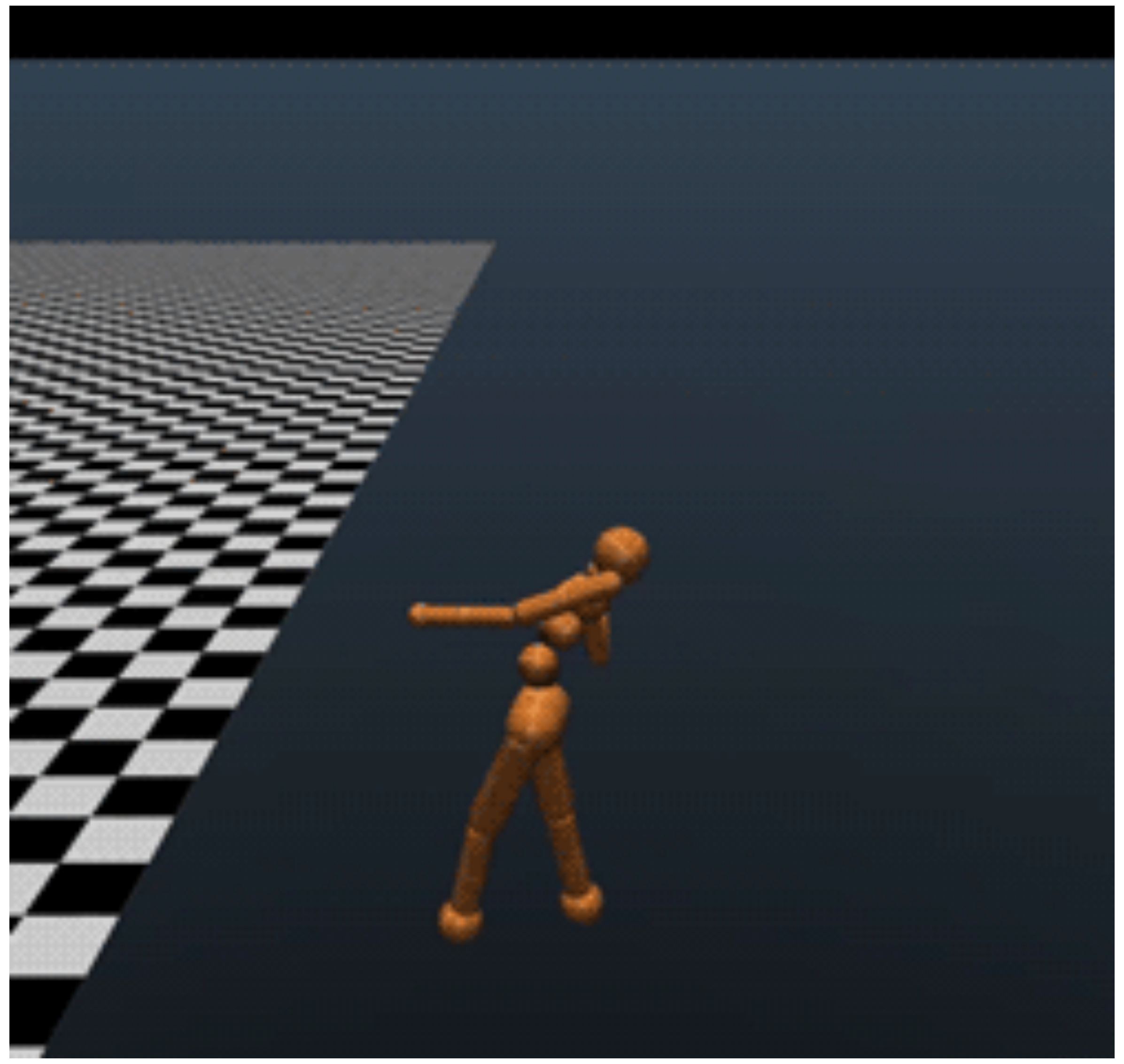
Train a robot to "run" forward as fast as possible:

State: joint angles, center of mass, velocity, etc

Action: torques on joints

Reward: distance of moving forward between two steps





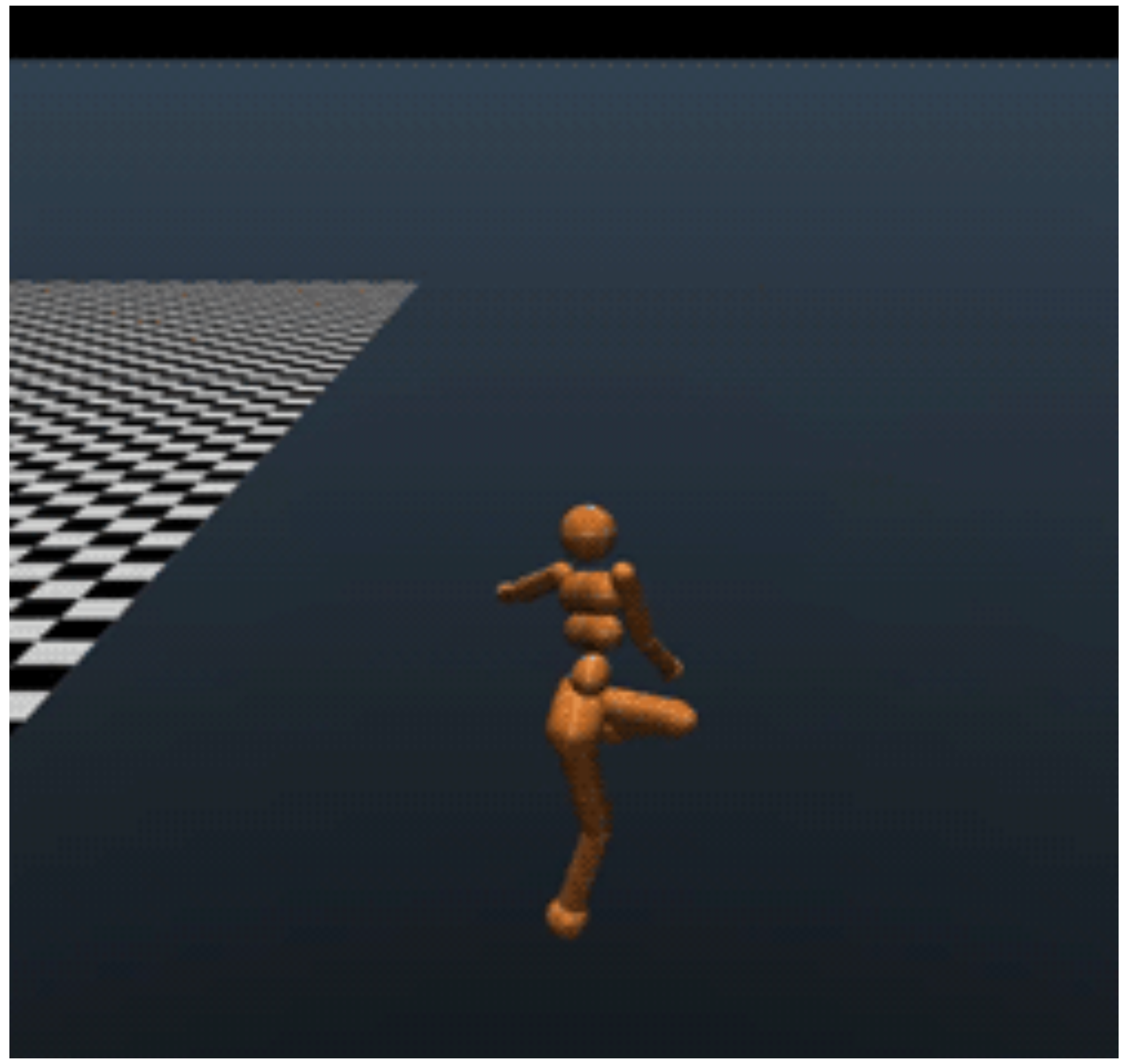
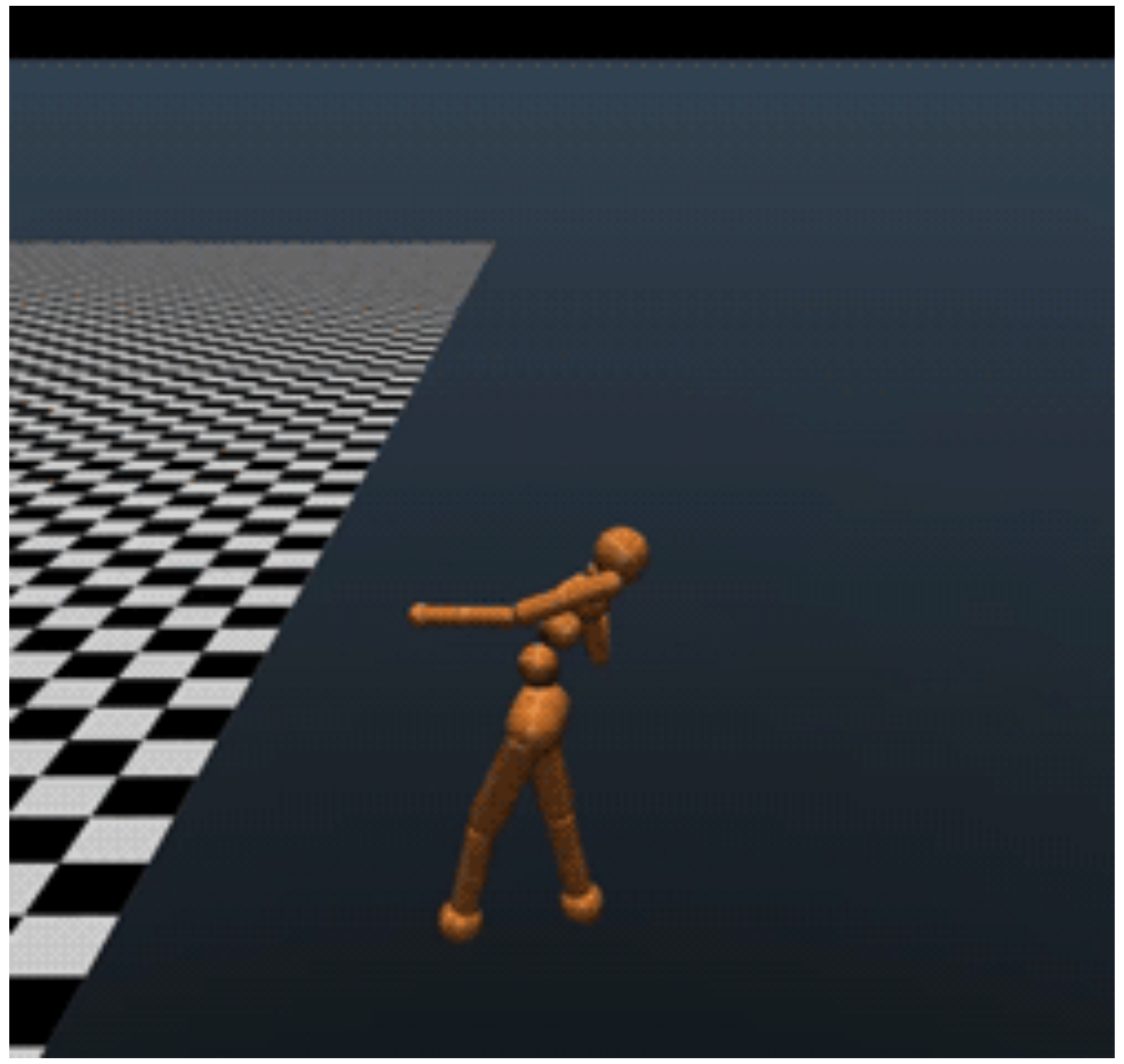
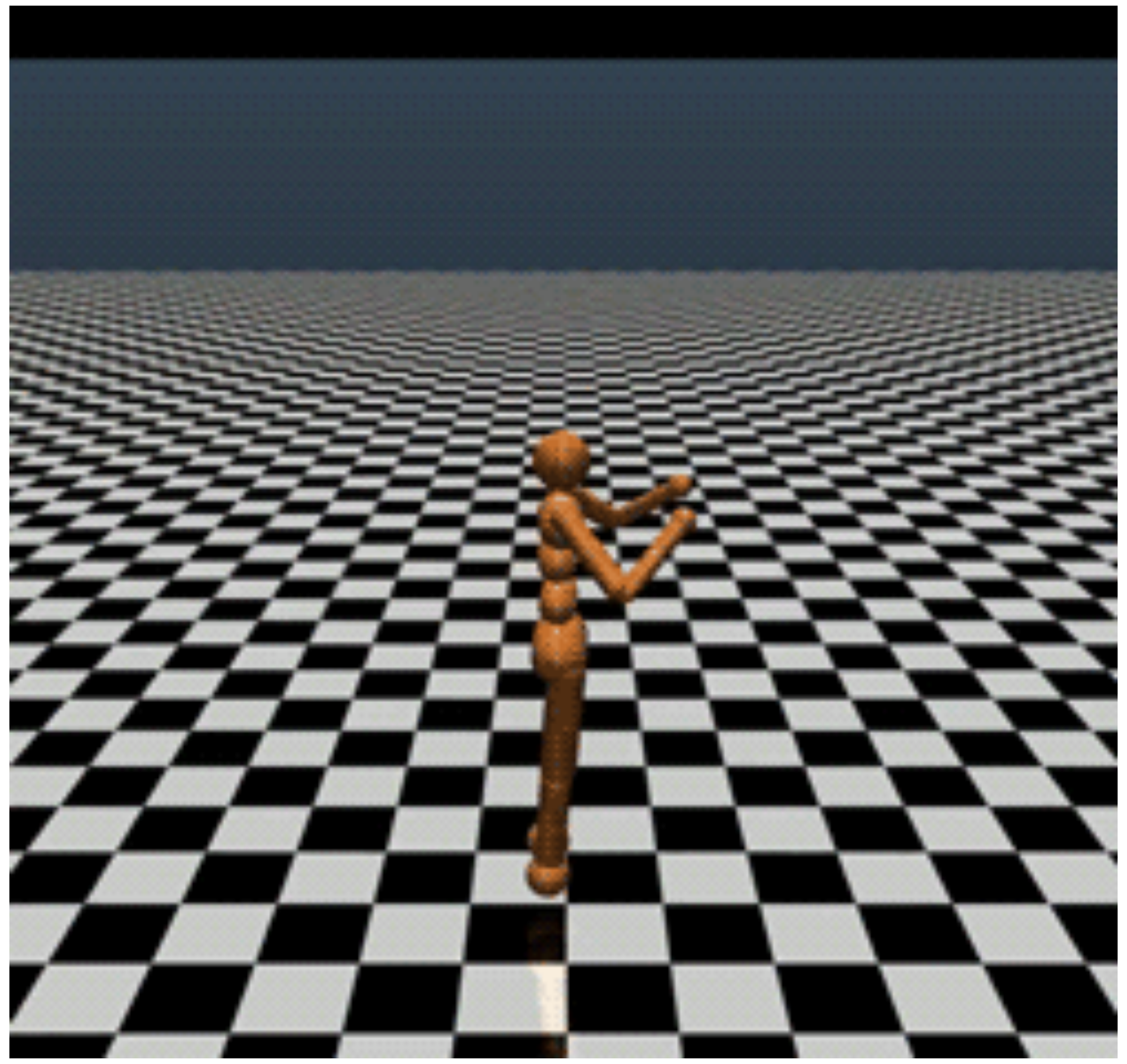
Example application of PPO:

Train a robot to “run” forward as fast as possible:

State: joint angles, center of mass, velocity, etc

Action: torques on joints

Reward: distance of moving forward between two steps

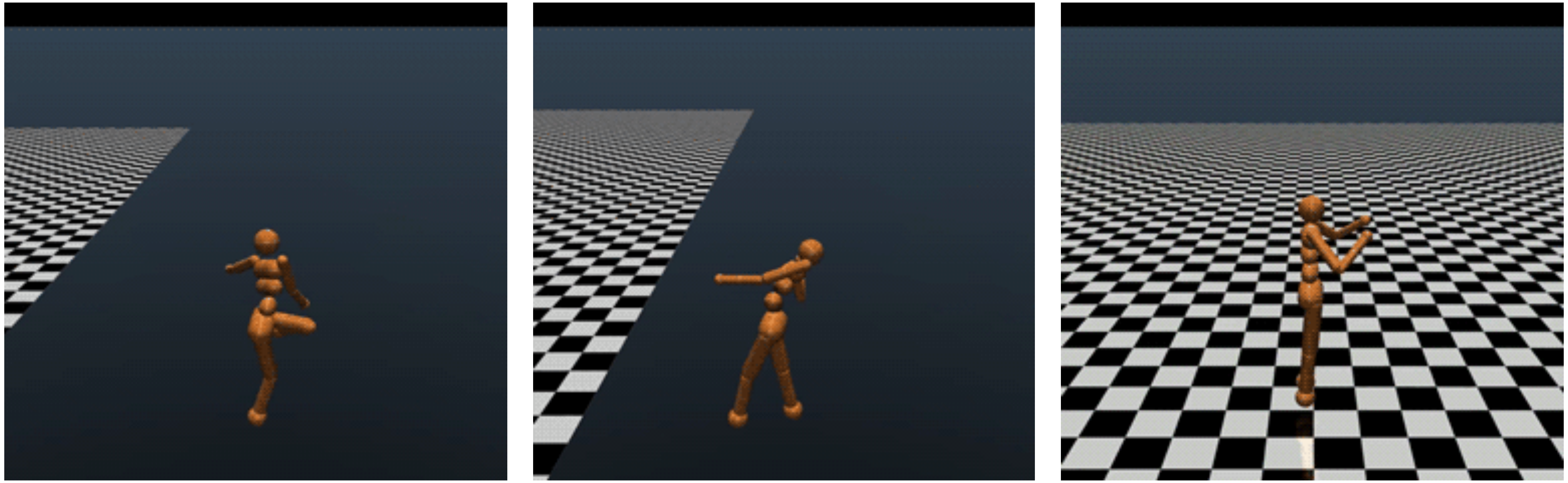
Example application of PPO:

Train a robot to "run" forward as fast as possible:

State: joint angles, center of mass, velocity, etc

Action: torques on joints

Reward: distance of moving forward between two steps

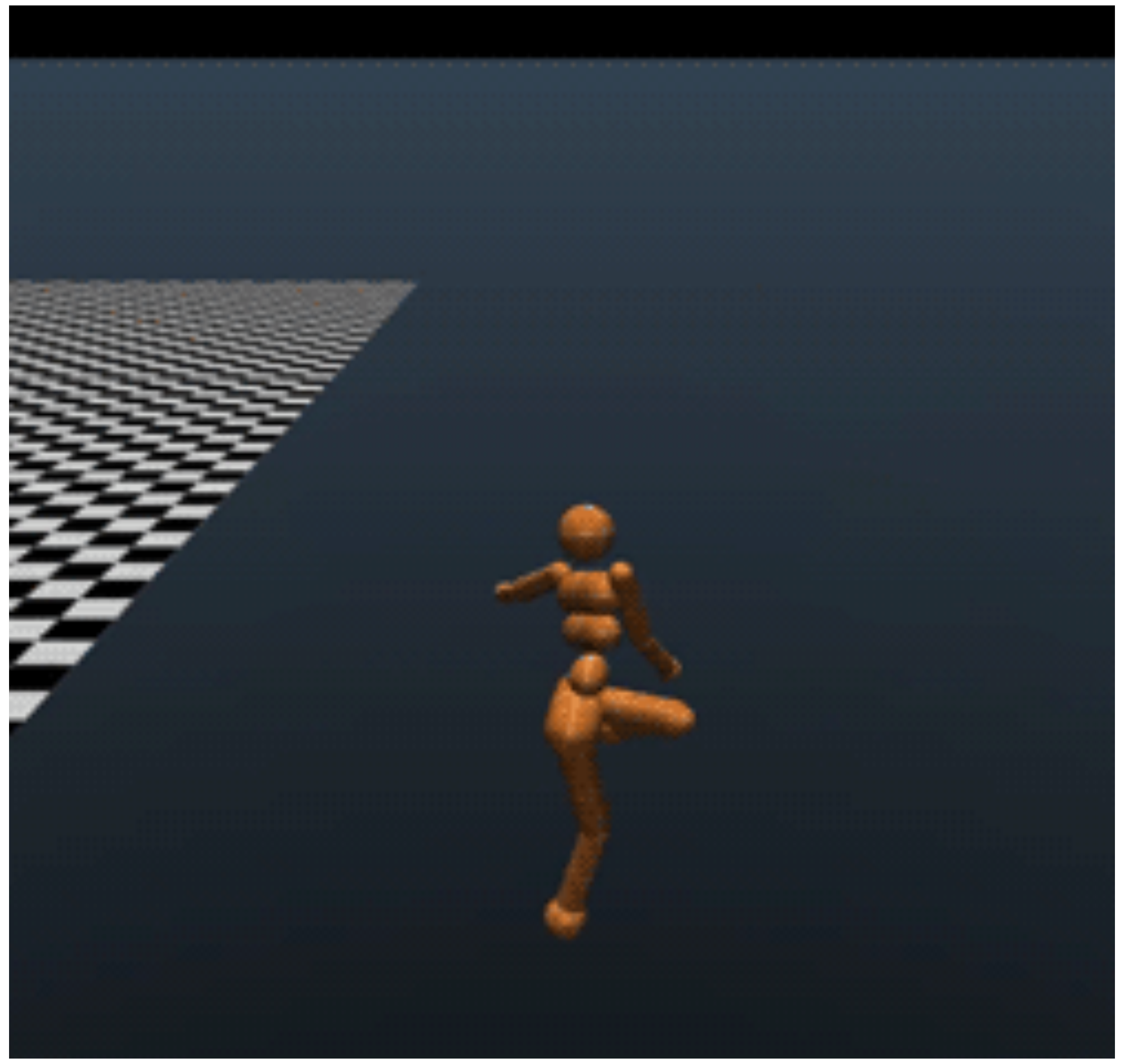


(BTW, This reveals an issue on reward design—you may study it in Inverse RL lectures)

Naive Policy Gradient can unstable and slow

The potential high-variance in PG can make learning very unstable

Naive Policy Gradient can unstable and slow

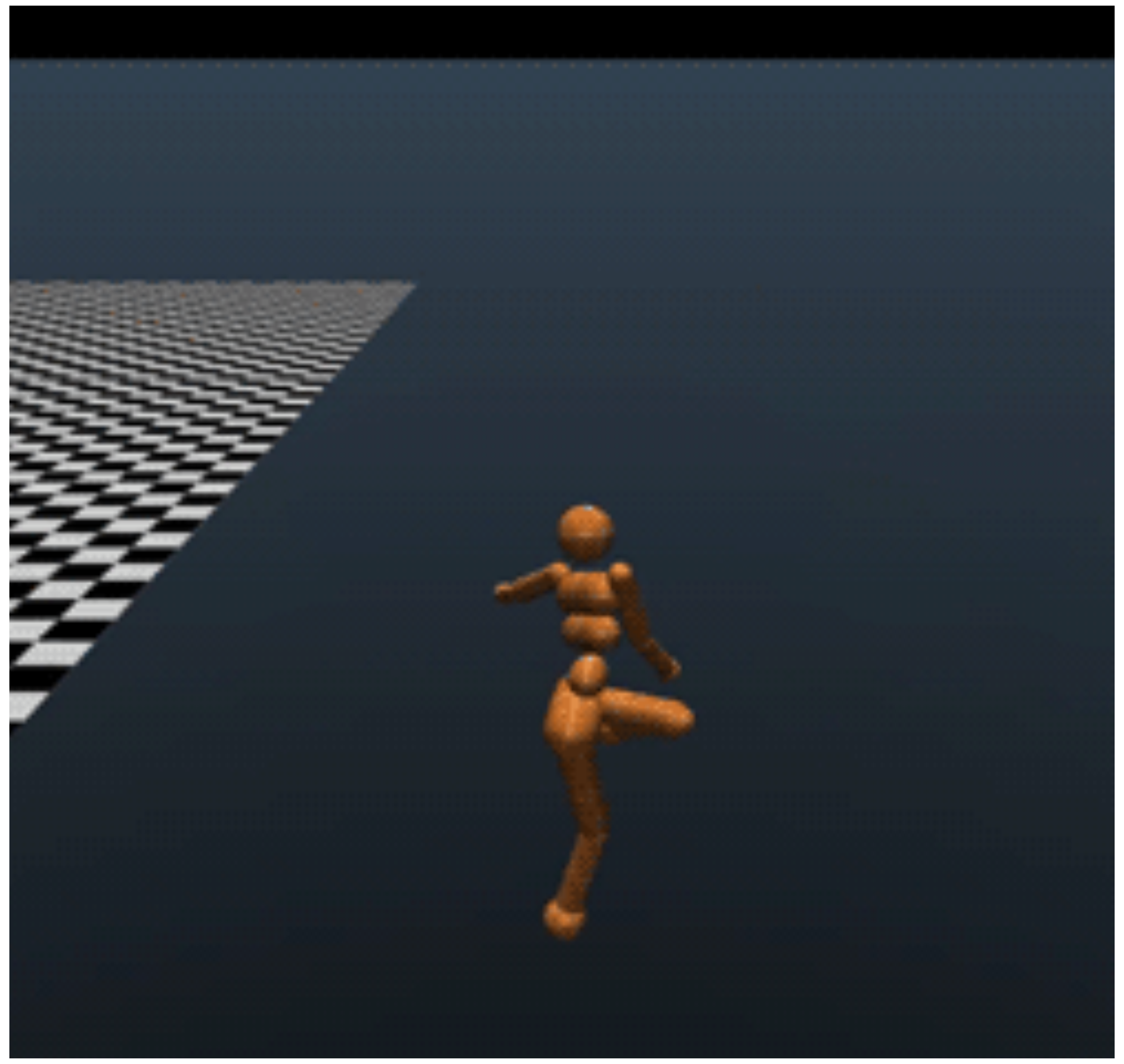


Too frequent!

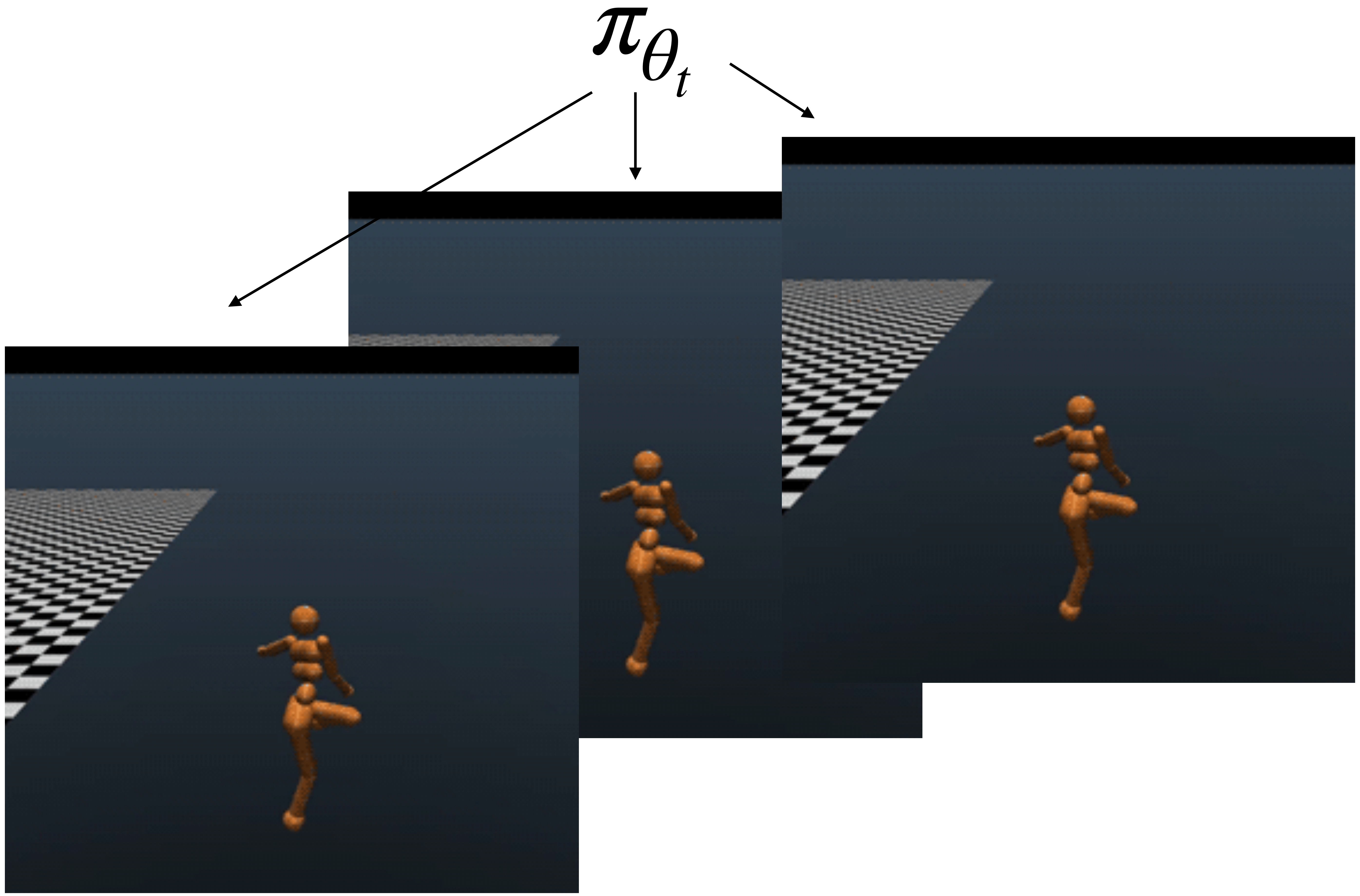
GPU usage can be very low…

Proximal Policy Optimization (PPO)

π_{θ_t}

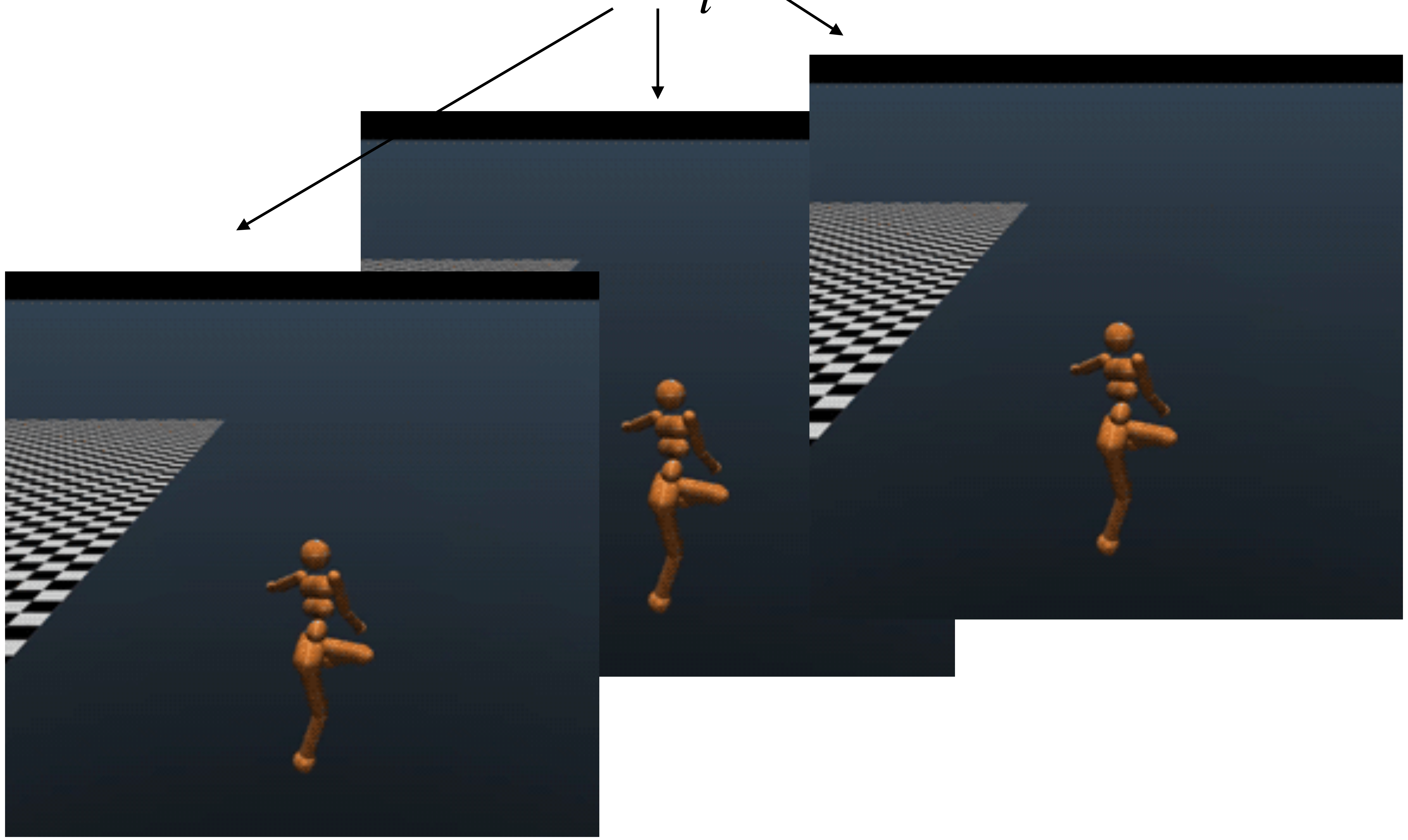


Proximal Policy Optimization (PPO)



Proximal Policy Optimization (PPO)

π_{θ_t}

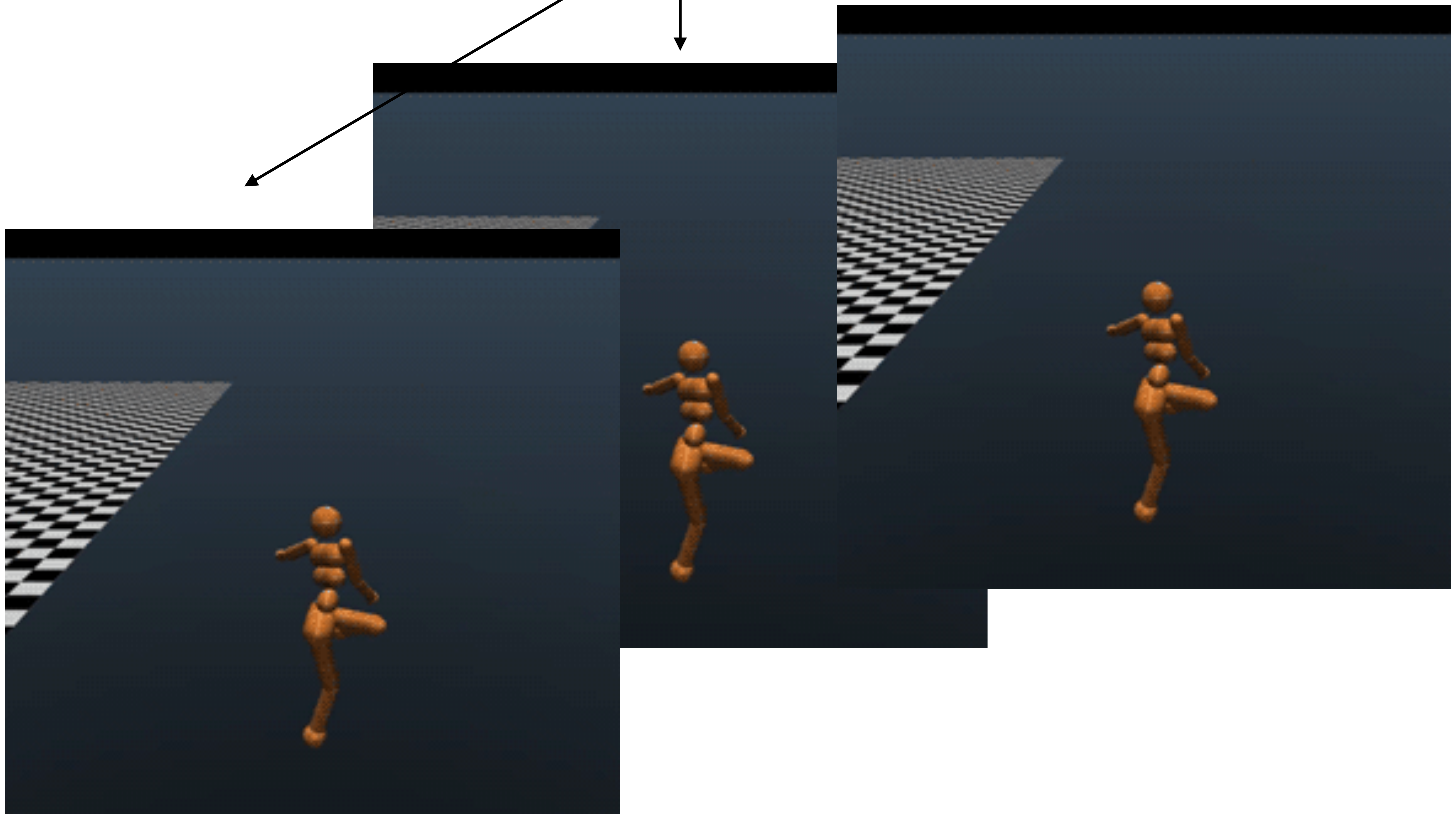


Collect a large dataset

$$\longrightarrow \mathcal{D} = \left\{ s, a, A^{\pi_{\theta_t}}(s, a) \right\}$$

Proximal Policy Optimization (PPO)

π_{θ_t}

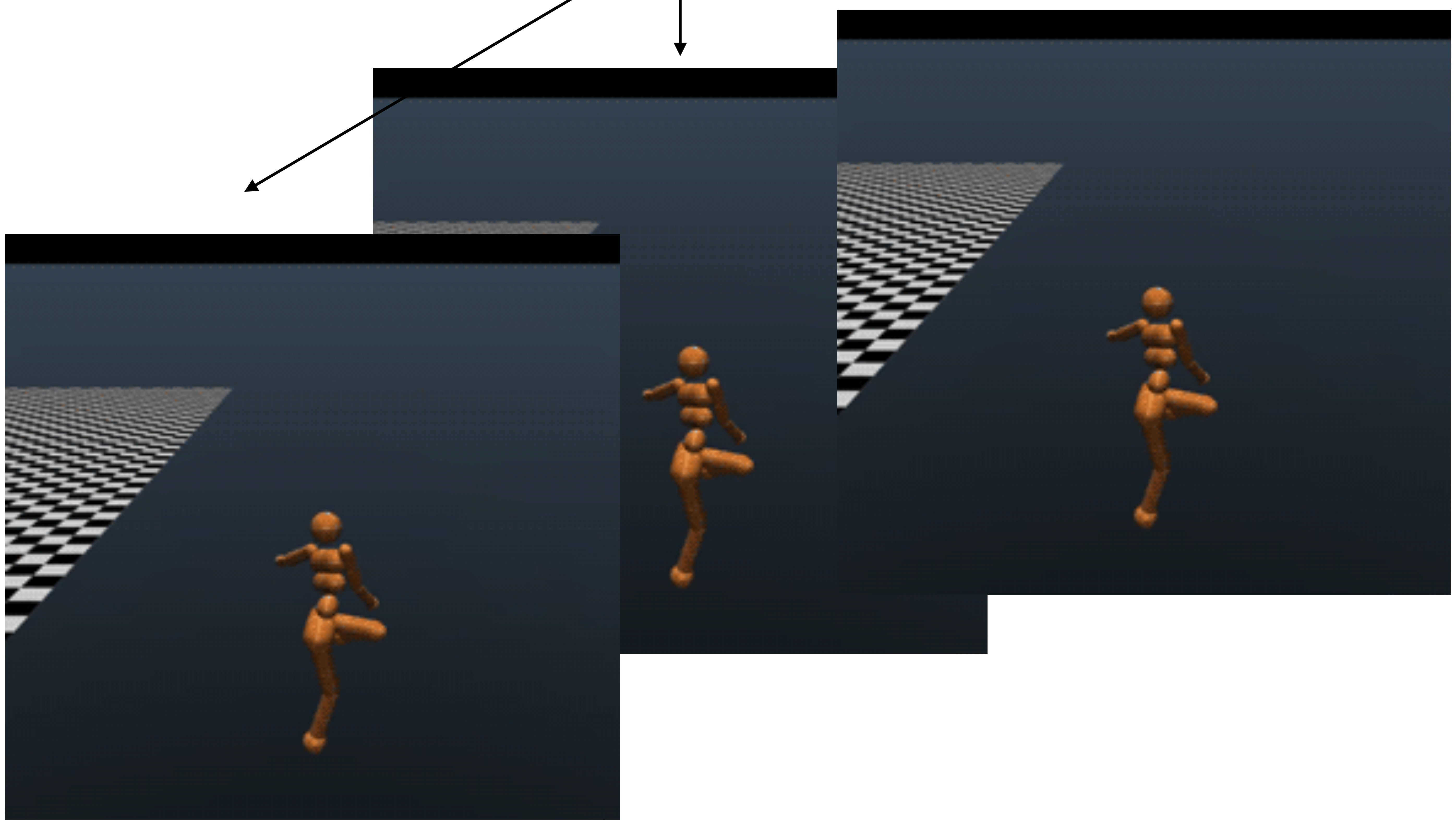


Collect a large dataset

$$\mathscr{D} = \left\{ s, a, A^{\pi_{\theta_t}}(s, a) \right\}$$

Proximal Policy Optimization (PPO)

π_{θ_t}



Collect a large dataset

$$\mathscr{D} = \left\{ s, a, A^{\pi_{\theta_t}}(s, a) \right\}$$

Now let's do multiple epoches of min-batch gradient update on the dataset

Proximal Policy Optimization (PPO)

Construct a batch Supervised Learning style objective using $\mathscr{D} = \{s, a, A^{\pi_{\theta_t}}(s, a)\}$

$$\max_\theta \ell(\theta) = \max_\theta \mathbb{E}_{s\sim d^{\pi_{\theta_t}}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

Proximal Policy Optimization (PPO)

Construct a batch Supervised Learning style objective using $\mathscr{D} = \{s, a, A^{\pi_{\theta_t}}(s, a)\}$

$$\max_\theta \ell(\theta) = \max_\theta \mathbb{E}_{s\sim d^{\pi_{\theta_t}}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

IW trick $\rightarrow \mathbb{E}_{s\sim d^{\pi_{\theta_t}}}\mathbb{E}_{a\sim\pi_{\theta_t}(\cdot|s)} \dfrac{\pi_\theta(\,\cdot\,|\,s)}{\pi_{\theta_t}(a\,|\,s)} \cdot A^{\pi_{\theta_t}}(s, a)$

Proximal Policy Optimization (PPO)

Construct a batch Supervised Learning style objective using $\mathscr{D} = \{s, a, A^{\pi_{\theta_t}}(s, a)\}$

$$\max_\theta \ell(\theta) = \max_\theta \mathbb{E}_{s\sim d^{\pi_{\theta_t}}} \mathbb{E}_{a\sim\pi_\theta(\cdot|s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

IW trick $$\to \mathbb{E}_{s\sim d^{\pi_{\theta_t}}} \mathbb{E}_{a\sim\pi_{\theta_t}(\cdot|s)} \frac{\pi_\theta(\,\cdot \mid s)}{\pi_{\theta_t}(a \mid s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

$$\approx \sum_{s,a} \frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

Proximal Policy Optimization (PPO)

Construct a batch Supervised Learning style objective using $\mathcal{D} = \{s, a, A^{\pi_{\theta_t}}(s, a)\}$

$$\hat{\ell}(\theta) = \sum_{s,a} \frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

Proximal Policy Optimization (PPO)

Construct a batch Supervised Learning style objective using $\mathcal{D} = \{s, a, A^{\pi_{\theta_t}}(s, a)\}$

$$\hat{\ell}(\theta) = \sum_{s,a} \frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

Trick 1: clipping to make sure π_θ stay close to π_{θ_t} (ensuring stability in training)

Proximal Policy Optimization (PPO)

Construct a batch Supervised Learning style objective using $\mathscr{D} = \{s, a, A^{\pi_{\theta_t}}(s, a)\}$

$$\hat{\ell}(\theta) = \sum_{s,a} \frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

Trick 1: clipping to make sure π_θ stay close to π_{θ_t} (ensuring stability in training)

$$\hat{\ell}_{clip}(\theta) = \sum_{s,a} \text{clip}\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)}, 1 - \epsilon, 1 + \epsilon\right) \cdot A^{\pi_{\theta_t}}(s, a)$$

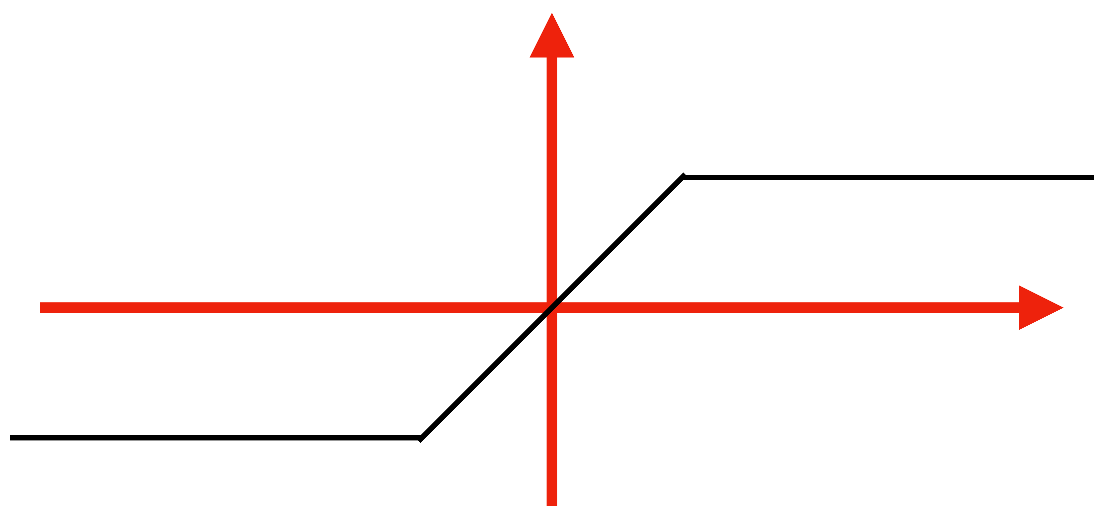
Proximal Policy Optimization (PPO)

Construct a batch Supervised Learning style objective using $\mathscr{D} = \{s, a, A^{\pi_{\theta_t}}(s, a)\}$

$$\hat{\ell}(\theta) = \sum_{s,a} \frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

Trick 1: clipping to make sure π_θ stay close to π_{θ_t} (ensuring stability in training)

$$\hat{\ell}_{clip}(\theta) = \sum_{s,a} \text{clip}\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)}, 1 - \epsilon, 1 + \epsilon \right) \cdot A^{\pi_{\theta_t}}(s, a)$$



$\text{clip}(x, 1 - \epsilon, 1 + \epsilon)$

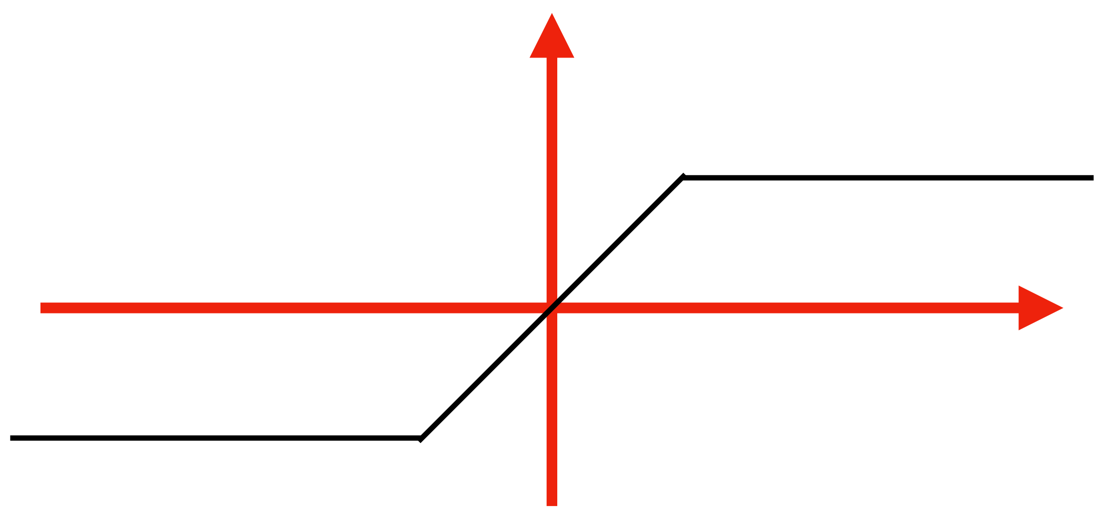
Proximal Policy Optimization (PPO)

Construct a batch Supervised Learning style objective using $\mathscr{D} = \{s, a, A^{\pi_{\theta_t}}(s, a)\}$

$$\hat{\ell}(\theta) = \sum_{s,a} \frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)} \cdot A^{\pi_{\theta_t}}(s, a)$$

Trick 1: clipping to make sure π_θ stay close to π_{θ_t} (ensuring stability in training)

$$\hat{\ell}_{clip}(\theta) = \sum_{s,a} \text{clip}\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)}, 1-\epsilon, 1+\epsilon \right) \cdot A^{\pi_{\theta_t}}(s, a)$$



clip$(x, 1-\epsilon, 1+\epsilon)$

Stop updating $\pi_\theta(a \mid s)$ if it is too different from $\pi_{\theta_t}(a \mid s)$

Proximal Policy Optimization (PPO)

Trick 2, take the min of the clipped and uncipped (original) obj

$$\hat{\ell}_{final}(\theta) = \sum_{s,a} \min \left\{ \frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)} \cdot A^{\pi_{\theta_t}}(s, a), \quad \text{clip}\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_t}(a \mid s)}, 1 - \epsilon, 1 + \epsilon \right) \cdot A^{\pi_{\theta_t}}(s, a) \right\}$$

Original obj

clipped obj which ensures no abrupt change in action probabilities

Proximal Policy Optimization (PPO)

Trick 2, take the min of the clipped and uncipped (original) obj

$$\hat{\ell}_{final}(\theta) = \sum_{s,a} \min \left\{ \frac{\pi_\theta(a|s)}{\pi_{\theta_t}(a|s)} \cdot A^{\pi_{\theta_t}}(s,a), \quad \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_t}(a|s)}, 1-\epsilon, 1+\epsilon \right) \cdot A^{\pi_{\theta_t}}(s,a) \right\}$$

Original obj

clipped obj which ensures no abrupt change in action probabilities

We compute $\theta_{t+1} \approx \arg\max_\theta \hat{\ell}_{final}(\theta)$, via performing a few epoches of minbatch SG ascent (or Adam/Adagrad) on $\hat{\ell}_{final}$

Outline:

1. A $Q(s, a)$ based Policy Gradient and Variance reduction

2. Proximal Policy Optimization (it trains ChatGPT!)

3. How to address the exploration challenge

Policy gradient cannot do exploration…

Initialization: s_0

Length of chain is H

Policy gradient cannot do exploration...

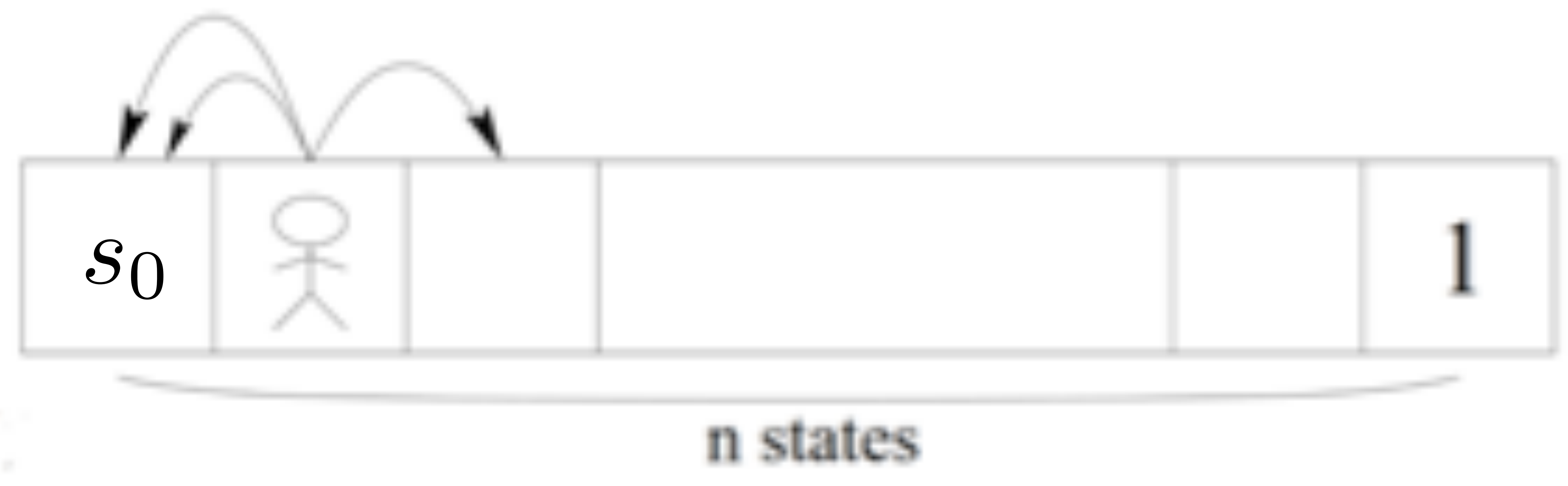
Initialization: s_0

Length of chain is H

Probability of random walk hitting reward 1 is $(1/3)^{-H}$

Policy gradient cannot do exploration…

Initialization: s_0



Thrun '92

Length of chain is H

Probability of random walk hitting reward 1 is $(1/3)^{-H}$

Unless we take exponentially many trajs, empirical policy gradient is zero

Common fixes

1. Instead of starting from randomly initialized policy, starting from a good pre-trained policy

Common fixes

1. Instead of starting from randomly initialized policy, starting from a good pre-trained policy

BC -> PG (e.g., GPT3->ChatGPT)

Common fixes

1. Instead of starting from randomly initialized policy, starting from a good pre-trained policy

BC -> PG (e.g., GPT3->ChatGPT)

2. Reset based on some informative state distribution (e.g., explorative distribution, expert demos)

Common fixes

1. Instead of starting from randomly initialized policy, starting from a good pre-trained policy

BC -> PG (e.g., GPT3->ChatGPT)

2. Reset based on some informative state distribution (e.g., explorative distribution, expert demos)

e.g., start uniform randomly along the chain instead of just s_0

Thrun '92

Summary

The advantage A^π based policy gradient formulation

PPO: the cliping trick to stabilize training; fast, efficient, and scalable

PG can stuck at local optimal; use BC or reset to help PG succeed