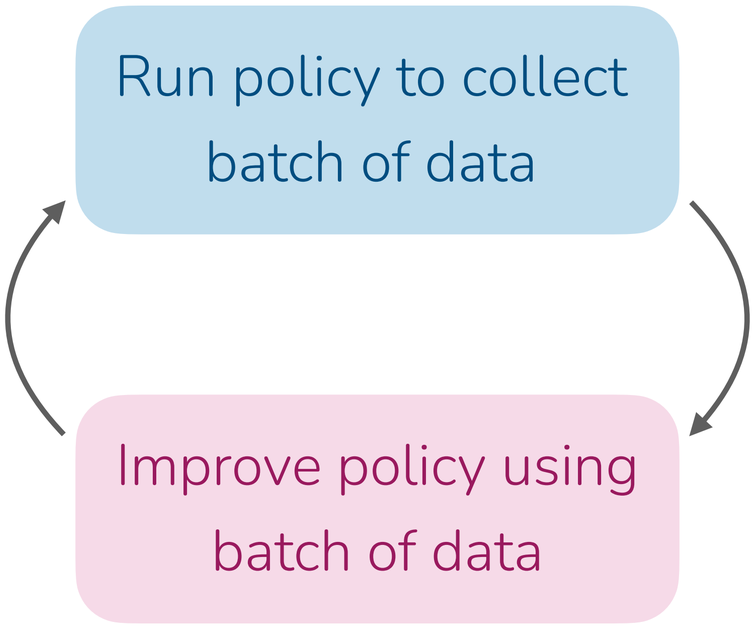
Off-Policy Actor Critic Methods

CS 224R

Course reminders

- **Friday 4:45 pm**: Section on value functions, Q-learning
- Project survey due in one week
- **Homework 2** due next Friday

Recap: Policy Gradients



Online reinforcement learning with policy gradients

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \left(\left(\sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right) - b \right)$$

samples from policy — policy log likelihood — reward to go — baseline

Do more of the above average stuff, less of the below average stuff.

Importance weights for *off-policy* policy gradient:

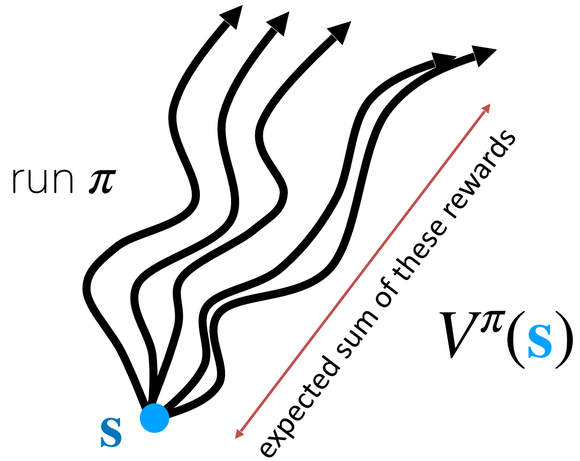
$$\nabla_{\theta'} J(\theta') \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \left(\left(\sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right) - b \right)$$

Recap: Some Useful Objects

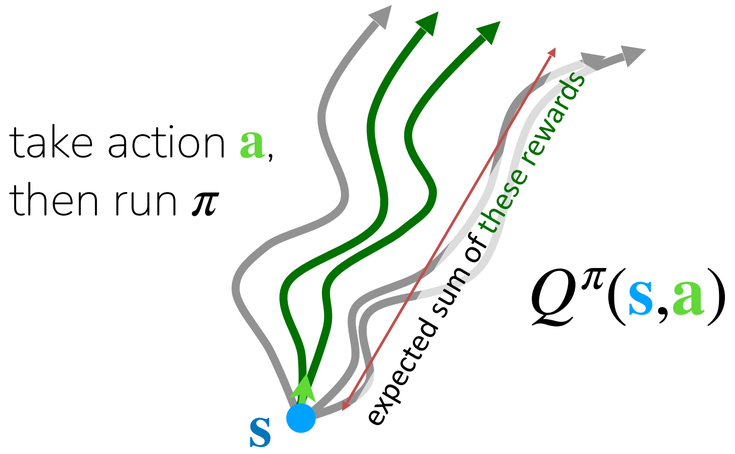
value function $V^{\pi}(\mathbf{s})$ - future expected rewards starting at $\mathbf{s}$ and following π

Q-function $Q^{\pi}(\mathbf{s}, \mathbf{a})$ - future expected rewards starting at $\mathbf{s}$, taking $\mathbf{a}$, then following π

$$V^{\pi}(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t\right]$$



$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$$

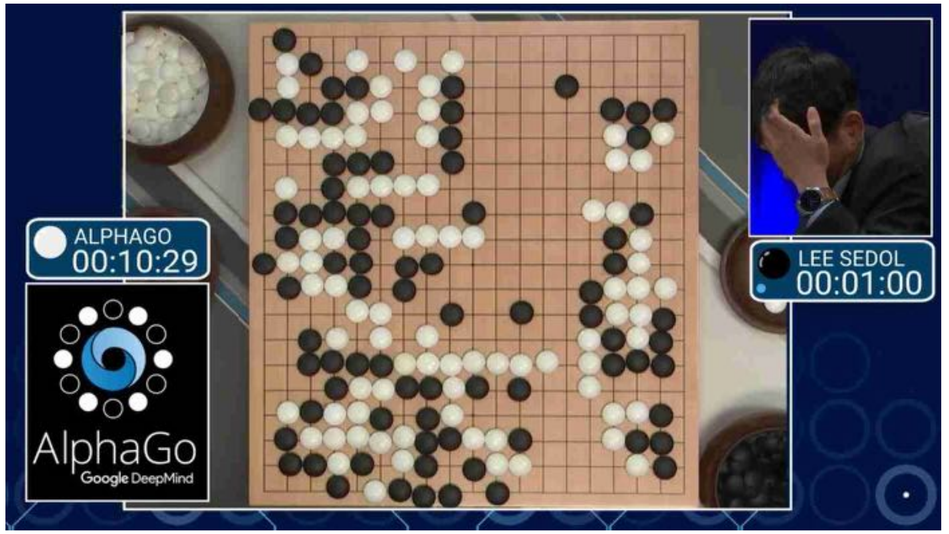


advantage $A^{\pi}(\mathbf{s}, \mathbf{a})$ - how much better it is to take $\mathbf{a}$ than to follow policy π at state $\mathbf{s}$

$$A^{\pi}(\mathbf{s}, \mathbf{a}) = Q^{\pi}(\mathbf{s}, \mathbf{a}) - V^{\pi}(\mathbf{s})$$

Policy evaluation example

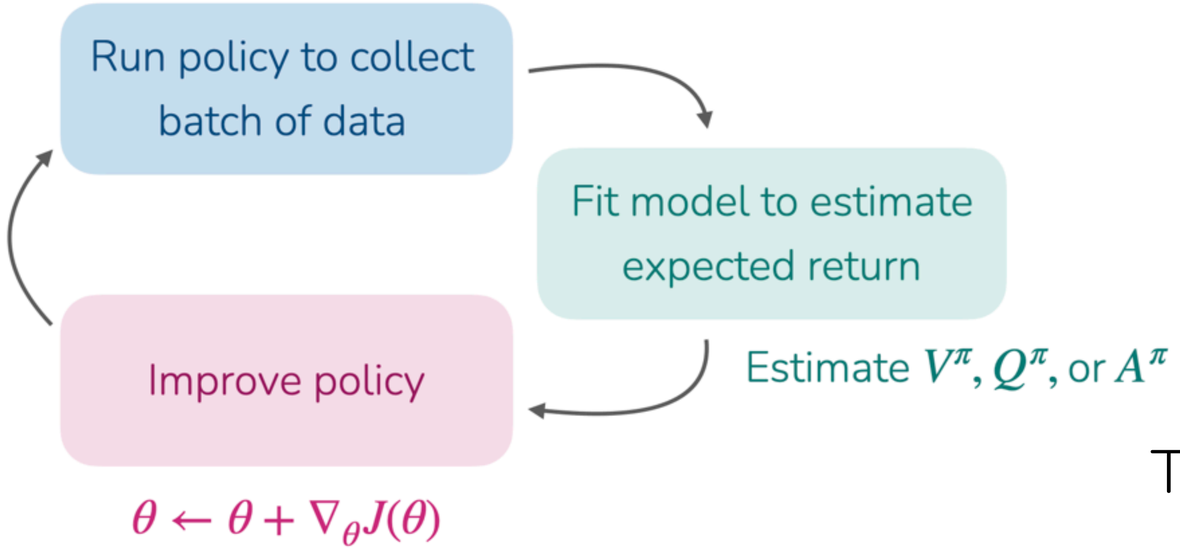
AlphaGo, Silver et al. 2016



reward: game outcome

value function $\hat{V}_\phi^\pi(\mathbf{s}_t)$:
expected outcome given board state

Recap: Actor-Critic Methods



Online reinforcement learning with actor-critic

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) A^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

Estimate what is good and bad, then do more of the good stuff.

To estimate value, fit $\hat{V}^\pi$:

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}^\pi_\phi(\mathbf{s}_i) - y_i \right\|^2$$

$$\text{where } y_{i,t} = \sum_{t'=t}^{t+n-1} \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) + \gamma^n \hat{V}^\pi_\phi(\mathbf{s}_{i,t+n})$$

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$$

The plan for today

Off-policy actor critic methods

1. Taking multiple gradient steps
    a. Importance weights
    b. Constraining step size with KL penalty or clipping
    c. Practical PPO algorithm
2. Even more off-policy algorithm
    d. Fitting Q-value functions with data from other policies
    e. Practical SAC algorithm
3. Comparisons between PPO, SAC, imitation learning

Key learning goals:

- All of the key concepts for practical algorithms like PPO and SAC

Off-Policy Actor-Critic Methods

Version 1: Multiple Gradient Steps

1. Sample batch of data $\{(\mathbf{s}_{1,i}, \mathbf{a}_{1,i}, \ldots, \mathbf{s}_{T,i}, \mathbf{a}_{T,i})\}$ from π_θ
2. Fit $\hat{V}^{\pi_\theta}_\phi$ to summed rewards in data
3. Evaluate $\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) = r(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \gamma \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t+1,i}) - \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t,i}) \quad \forall t, i$
4. Evaluate $\nabla_\theta J(\theta) \approx \sum_{t,i} \nabla_\theta \log \pi_\theta(\mathbf{a}_{t,i} \,|\, \mathbf{s}_{t,i}) \; \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$ <- use importance weights here
5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Off-Policy Actor-Critic Methods

Version 1: Multiple Gradient Steps

1. Sample batch of data $\{(\mathbf{s}_{1,i}, \mathbf{a}_{1,i}, \ldots, \mathbf{s}_{T,i}, \mathbf{a}_{T,i})\}$ from π_θ
2. Fit $\hat{V}^{\pi_\theta}_\phi$ to summed rewards in data
3. Evaluate $\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) = r(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \gamma \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t+1,i}) - \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t,i}) \quad \forall t, i$
4. Evaluate $\nabla_{\theta'} J(\theta') \approx \sum_{t,i} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{t,i}|\mathbf{s}_{t,i}) \; \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$ <- use importance weights here
5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

What can go wrong?

Off-Policy Actor-Critic Methods

Version 1: Multiple Gradient Steps

1. Sample batch of data $\{(\mathbf{s}_{1,i}, \mathbf{a}_{1,i}, \ldots, \mathbf{s}_{T,i}, \mathbf{a}_{T,i})\}$ from π_θ
2. Fit $\hat{V}^{\pi_\theta}_\phi$ to summed rewards in data
3. Evaluate $\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) = r(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \gamma \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t+1,i}) - \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t,i}) \quad \forall t, i$
4. Evaluate $\nabla_{\theta'} J(\theta') \approx \sum_{t,i} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{t,i}|\mathbf{s}_{t,i}) \; \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$ <- use importance weights here
5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Let's look at surrogate objective:

$$\tilde{J}(\theta') \approx \sum_{t,i} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$$

Advantages based on old policy.

Policy will increase probability on actions with high advantages.

a convenient identity

$$p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) = \nabla_\theta p_\theta(\tau)$$

What can go wrong if you take a lot of gradient steps?

Policy is incentivized to differ significantly from old policy, can easily overfit.

Off-Policy Actor-Critic Methods

Version 1: Multiple Gradient Steps

Let's look at surrogate objective:

$$\tilde{J}(\theta') \approx \sum_{t,i} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$$

Advantages based on old policy.

Policy will increase probability on actions with high advantages.

a convenient identity

$$p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) = \nabla_\theta p_\theta(\tau)$$

What can go wrong if you take a lot of gradient steps?

Policy is incentivized to differ significantly from old policy, can easily overfit.

💡Idea 1: Use KL constraint on policy.

Very common. Will see in LLM preference optimization

$$\mathbb{E}_{\mathbf{s}\sim\pi_\theta}\left[D_{KL}(\pi_{\theta'}(\cdot \mid \mathbf{s}) \| \pi_\theta(\cdot \mid \mathbf{s}))\right] \leq \delta$$

💡Idea 2: Can we bound the importance weights?

Doesn't directly constrain policy, but removes incentives

-> A key idea behind proximal policy optimization (PPO)

Proximal Policy Optimization (PPO)

Off-policy actor-critic with a few tricks.

Surrogate objective:

$$\tilde{J}(\theta') \approx \sum_{t,i} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$$

Trick #1: Clip the importance weights:

$$\tilde{J}(\theta') \approx \sum_{t,i} \text{clip}\left(\frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}, 1-\epsilon, 1+\epsilon\right) \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$$

Policy no longer incentivized to deviate significantly

Trick #2: Take minimum w.r.t. original objective: (in rare event where clipping makes objective better)

$$\tilde{J}(\theta') \approx \sum_{t,i} \min\left(\frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}), \text{clip}\left(\frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}, 1-\epsilon, 1+\epsilon\right) \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})\right)$$

This is the final PPO surrogate objective!

How do we estimate the advantage?

Proximal Policy Optimization (PPO)

Off-policy actor-critic with a few tricks.

Surrogate objective:

$$\tilde{J}(\theta') \approx \sum_{t,i} \min\left(\frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i},\mathbf{a}_{t,i}), \text{clip}\left(\frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}, 1-\epsilon, 1+\epsilon\right)\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i},\mathbf{a}_{t,i})\right)$$

Trick #3: "Generalized advantage estimation" (GAE)

Fit V^π with Monte Carlo or bootstrapping.

Then, use varying horizon to estimate advantage:

$$\hat{A}^\pi_n(\mathbf{s}_t,\mathbf{a}_t) = \sum_{t'=t}^{t+n}\gamma^{t'-t}r(\mathbf{s}_{t'},\mathbf{a}_{t'}) - \hat{V}^\pi_\phi(\mathbf{s}_t) + \gamma^n\hat{V}^\pi_\phi(\mathbf{s}_{t+n})$$

$$\hat{A}^\pi_{\text{GAE}}(\mathbf{s}_t,\mathbf{a}_t) = \sum_{n=1}^{\infty} w_n \hat{A}^\pi_n(\mathbf{s}_t,\mathbf{a}_t)$$

How to choose weights? cut earlier for less variance

$$w_n \propto \lambda^{n-1} \text{ for } n = 1,\ldots,N, \text{ where } N \ll T$$

Why?

from one of the GAE authors:

Proximal Policy Optimization (PPO)

1. Sample batch of data $\{(\mathbf{s}_{1,i}, \mathbf{a}_{1,i}, \ldots, \mathbf{s}_{T,i}, \mathbf{a}_{T,i})\}$ from π_θ
2. Fit $\hat{V}^{\pi_\theta}_\phi$ to summed rewards in data
3. Evaluate $\hat{A}^\pi_{\text{GAE}}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{n=1}^{\infty} w_n \left(\sum_{t'=t}^{t+n} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}^\pi_\phi(\mathbf{s}_t) + \gamma^n \hat{V}^\pi_\phi(\mathbf{s}_{t+n}) \right)$
4. Update policy with M gradient steps on surrogate objective:

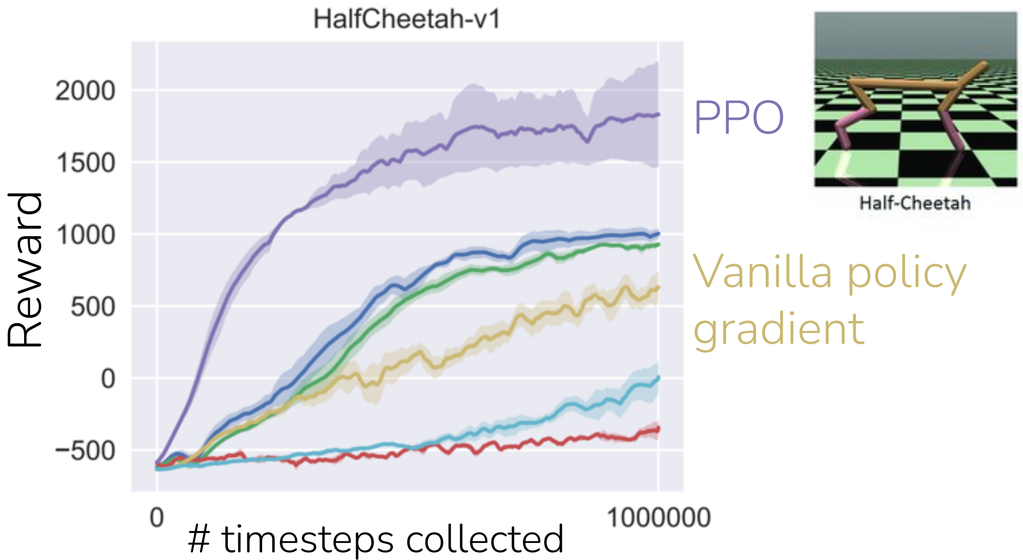
$$\tilde{J}(\theta') \approx \sum_{t,i} \min \left(\frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}), \text{clip} \left(\frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}, 1-\epsilon, 1+\epsilon \right) \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) \right)$$

Some example hyperparameters:

~2000 timesteps in batch of data

~10 epochs when updating policy clipping range ϵ=0.2
(M=~300 gradient steps with batch size 64)

~500 iterations —> 1M total timesteps of experience



Off-Policy Actor-Critic Methods

So far:
- use one batch of policy data for one gradient step (fully on-policy)
- use one batch of policy data for multiple gradient steps (starting to be off-policy)

Can we be even more off-policy?

Can we reuse data from previous batches, i.e. all of the past trial-and-error data?

Two key ideas:
- maintain a buffer of all past data *"replay buffer"*
- adjust equations to remove on-policy assumptions

Off-Policy Actor-Critic Methods

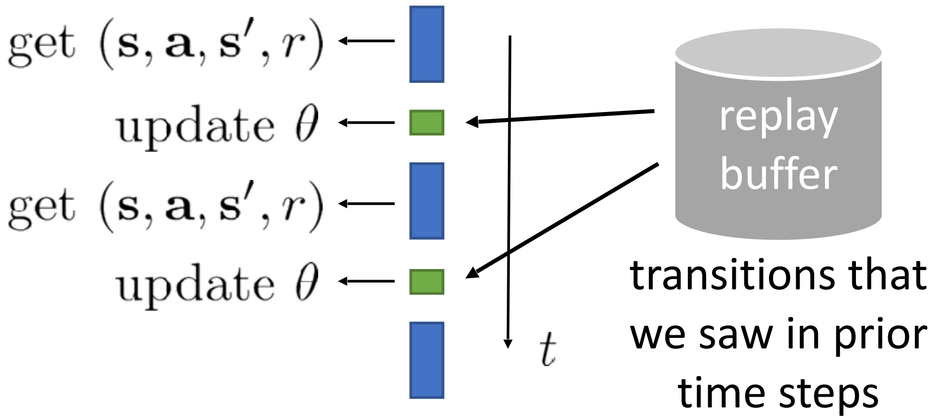
Version 2: Replay buffers

actor-critic algorithm:

1. collect experience $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ } Add this to replay buffer
2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

} Do this on minibatch sampled from **all previous data**

off-policy actor-critic



transitions that we saw in prior time steps

Slide adapted from Sergey Levine

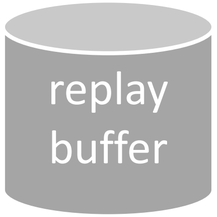
Off-Policy Actor-Critic Methods

Version 2: Replay buffers

1. collect experience $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (& add to replay buffer)
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{V}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i)$ for each $\mathbf{s}_i$
4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

not the action π_θ would have taken!

not the right target value



$$\mathcal{L}(\phi) = \frac{1}{N} \sum_i \left\| \hat{V}^\pi_\phi(\mathbf{s}_i) - y_i \right\|^2$$

mini batch size

Question: When we fit V^π on all data from replay buffer, what policy π is it learning a value function for?

The algorithm is currently broken 😢

Slide adapted from Sergey Levine

Off-Policy Actor-Critic Methods

How to we fit a value function for π_θ using replay buffer of data from past policies?

💡What if we fit $Q(\mathbf{s}, \mathbf{a})$ instead of $V(\mathbf{s})$?

depends on past policy's actions

The datapoints we have: $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$ + future reward

-> Don't want to use this.

action from past policy

-> If we fit $Q(\mathbf{s}, \mathbf{a})$, we pass the action as input
(okay if $\mathbf{a}$ is a bit different from what policy would have done)

Recall: *Definition of Q-values*

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$$

"total reward we get if we take $\mathbf{a}_t$ in $\mathbf{s}_t$...
... and then follow the policy π"

For *any* $(\mathbf{s}, \mathbf{a})$:

$$Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a}), \bar{\mathbf{a}}' \sim \pi_\theta(\cdot|\mathbf{s}')}\left[Q^{\pi_\theta}(\mathbf{s}', \bar{\mathbf{a}}')\right]$$

Off-Policy Actor-Critic Methods

How to we fit a value function for π_θ using replay buffer of data from past policies?

💡What if we fit $Q(\mathbf{s}, \mathbf{a})$ instead of $V(\mathbf{s})$?

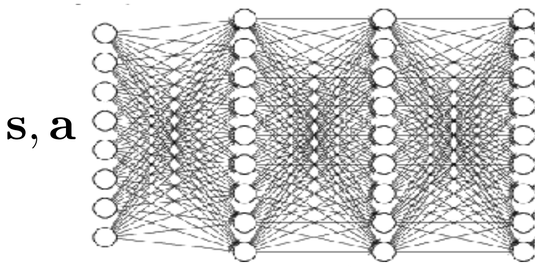
The datapoints we have: $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$ + future reward

For *any* $(\mathbf{s}, \mathbf{a})$: $Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a}), \bar{\mathbf{a}}' \sim \pi_\theta(\cdot|\mathbf{s}')} \left[Q^{\pi_\theta}(\mathbf{s}', \bar{\mathbf{a}}') \right]$

Approach:

1. Sample $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}_i')$ from the buffer
2. Sample $\bar{\mathbf{a}}_i' \sim \pi_\theta(\cdot|\mathbf{s}_i')$ from *current* policy

$$Q^{\pi_\theta}(\mathbf{s}_i, \mathbf{a}_i) \approx \underbrace{r(\mathbf{s}_i, \mathbf{a}_i) + \gamma Q^{\pi_\theta}(\mathbf{s}_i', \bar{\mathbf{a}}_i')}_{y_i}$$



$$\mathcal{L}(\phi) = \frac{1}{N} \sum_i \left\| \hat{Q}^\pi_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i \right\|^2$$

Note: To be accurate, targets y_i need sufficient action coverage

Off-Policy Actor-Critic Methods

Version 2: Fixing the value function

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{V}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i)$ for each $\mathbf{s}_i$
4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

not the right target value

3. update $\hat{Q}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$

not from replay buffer $\mathcal{R}$!

$\mathbf{a}'_i \sim \pi_\theta(\mathbf{a}'_i|\mathbf{s}'_i)$

Slide adapted from Sergey Levine

Off-Policy Actor-Critic Methods

Version 2: Fixing the value function

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ where $\mathbf{a}'_i \sim \pi_\theta(\cdot|\mathbf{s}'_i)$
4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

First: convenient to use $\hat{Q}^\pi$ instead of $\hat{A}^\pi$. (i.e. no average reward baseline)

higher variance, but okay b/c we're now using a lot more data (all data in buffer)

Second: current policy's actions likely better than past policy's actions

sample $\mathbf{a}^\pi_i \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$ $\quad$ $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}^\pi_i|\mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}^\pi_i)$

not from replay buffer $\mathcal{R}$!

Slide adapted from Sergey Levine

Off-Policy Actor-Critic Methods

Version 2: Anything else?

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ where $\mathbf{a}'_i \sim \pi_\theta(\cdot|\mathbf{s}'_i)$
4. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}^\pi_i|\mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}^\pi_i)$ where $\mathbf{a}^\pi_i \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Any remaining problems?

$\mathbf{s}_i$ didn't come from $p_\theta(\mathbf{s})$

nothing we can do here, just accept it

intuition: we want optimal policy on $p_\theta(\mathbf{s})$
but we get optimal policy on a *broader* distribution

Slide adapted from Sergey Levine

Off-Policy Actor-Critic Methods

Version 2: Some implementation details

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ where $\mathbf{a}'_i \sim \pi_\theta(\cdot|\mathbf{s}'_i)$
4. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}^\pi_i|\mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}^\pi_i)$ where $\mathbf{a}^\pi_i \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

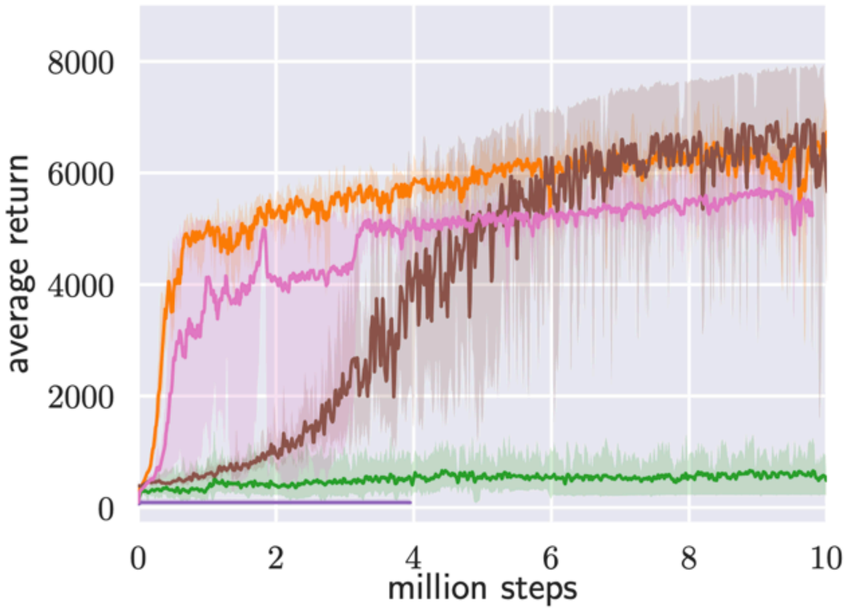
also fancier ways to fit Q-functions (more on this in next two lectures)

can also use **reparameterization trick** to better estimate the gradient (for Gaussian policy)

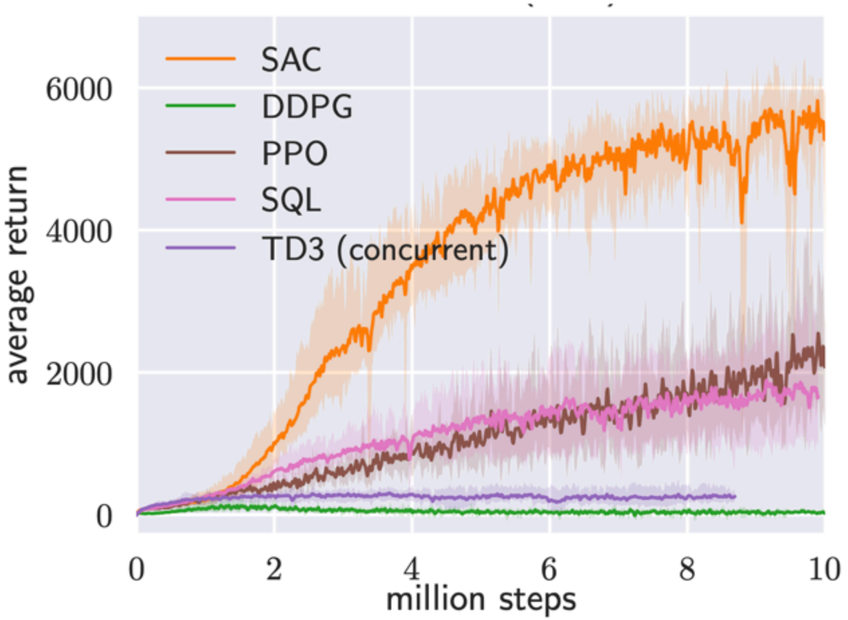
Example practical algorithm:

Haarnoja, Zhou, Abbeel, Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. 2018.

More off-policy vs. less off-policy actor critic?



(e) Humanoid-v1



(f) Humanoid (rllab)

<- more off-policy
(i.e. with replay buffer)

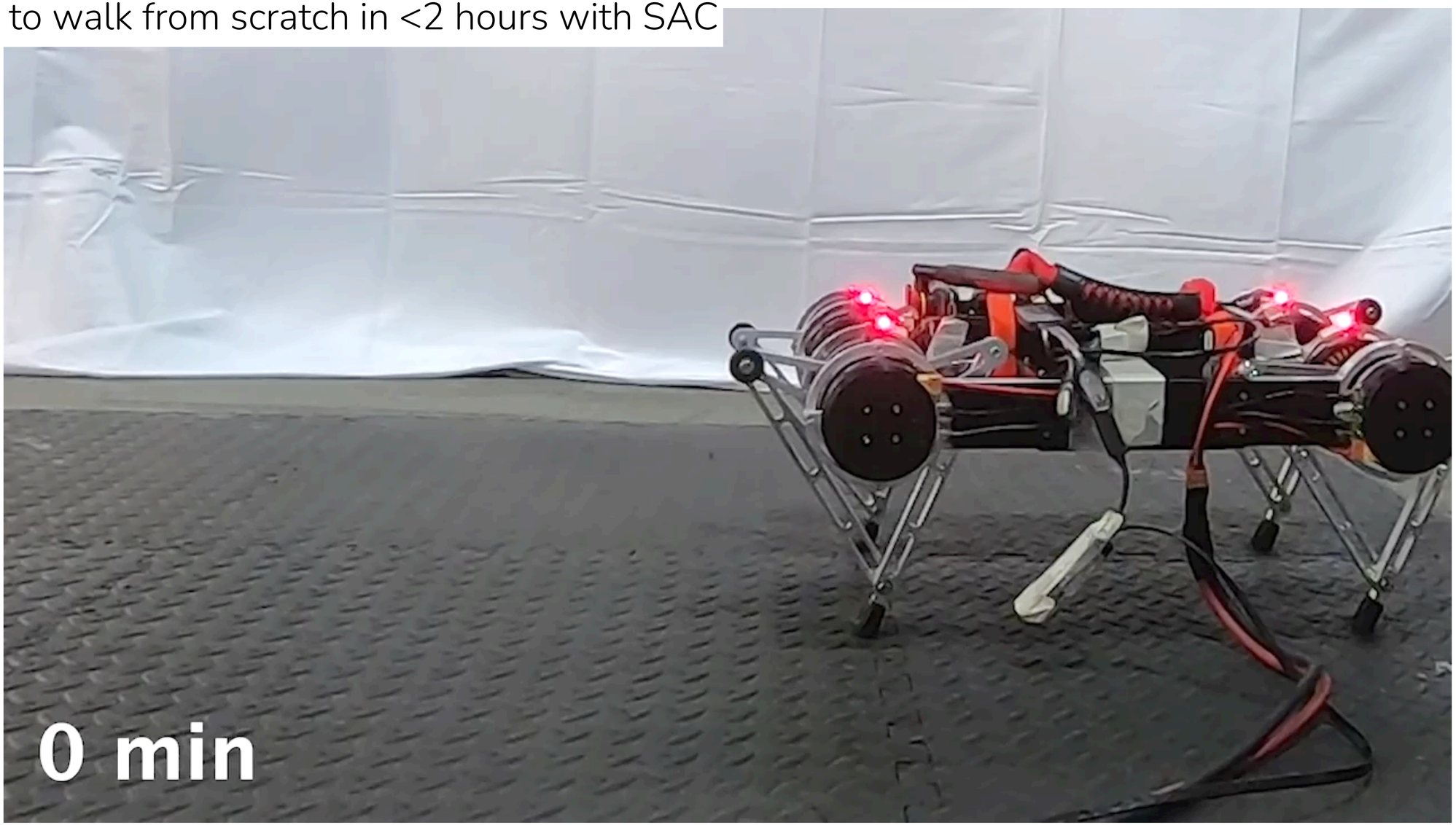
<- less off-policy
(i.e. no replay buffer)

+ Off-policy with replay buffer (e.g. soft actor-critic) can be far more data efficient

- They can also generally be a lot harder to tune hyperparameters, less stable (than e.g. PPO)

Fully Off-Policy = Can be efficient enough for real-world RL

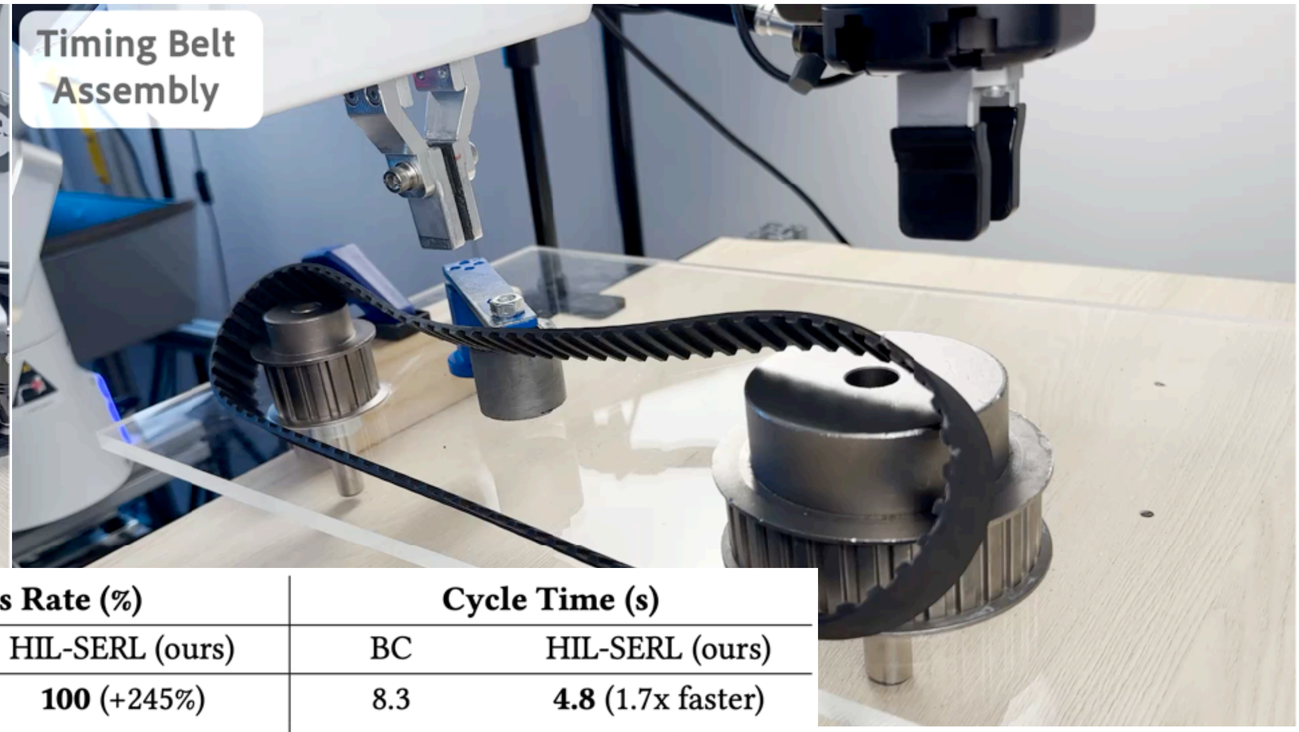
Learning to walk from scratch in <2 hours with SAC



Haarnoja, Zhou, Hartikainen, Tucker, Ha, Tan, Kumar, Zhu, Gupta, Abbeel, Levine. Soft Actor-Critic Algorithms and Applications. 2018.

Fully Off-Policy = Can be efficient enough for real-world RL

Learning precise assembly with algorithm based on SAC (seeded with demonstrations)



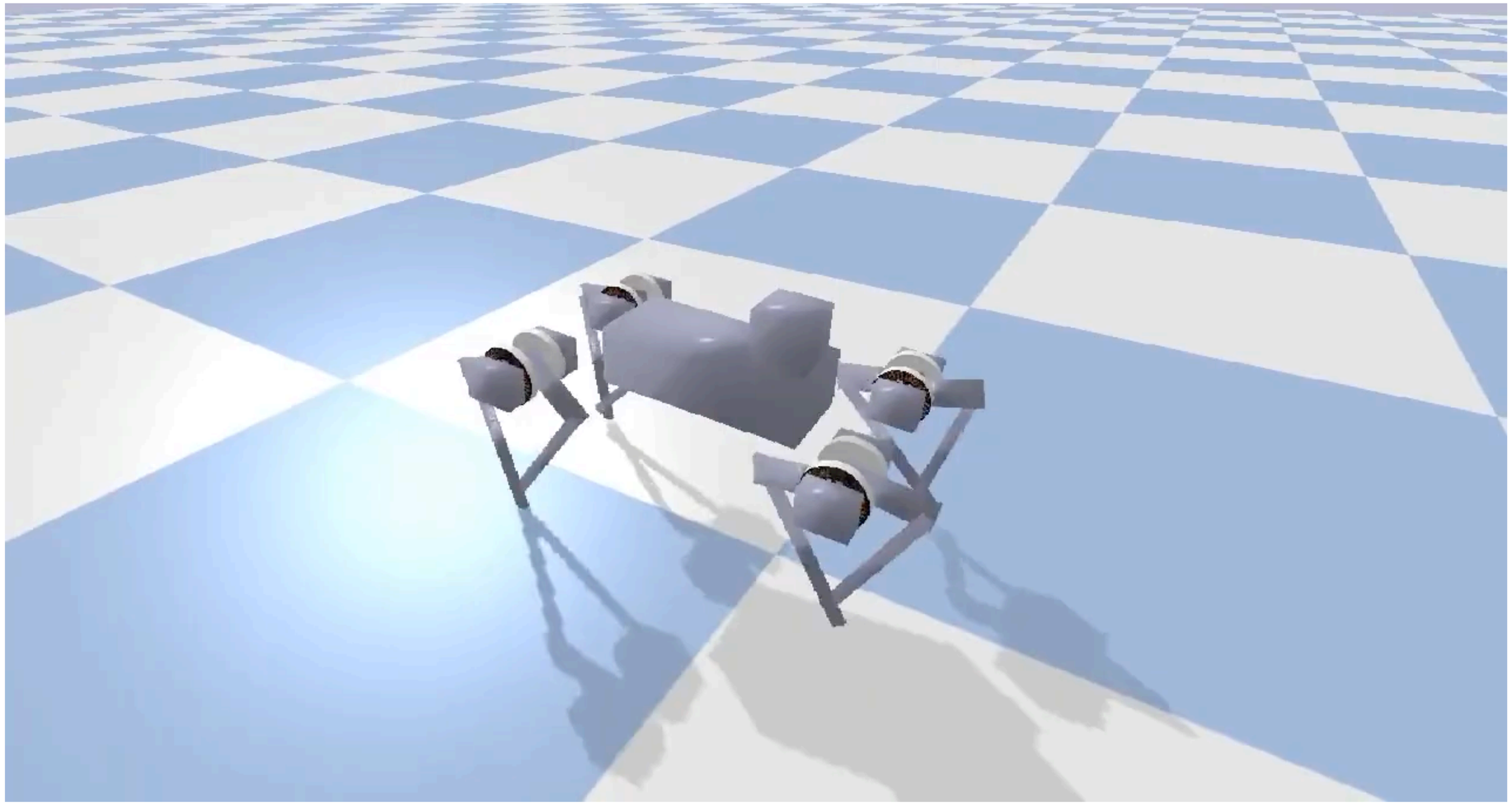
Task	Training Time (h)	Success Rate (%)		Cycle Time (s)	
		BC	HIL-SERL (ours)	BC	HIL-SERL (ours)
RAM Insertion	1.5	29	**100** (+245%)	8.3	**4.8** (1.7x faster)
SSD Assembly	1	79	**100** (+27%)	6.7	**3.3** (2x faster)
IKEA - Side Panel 1	2	77	**100** (+30%)	6.5	**2.7** (2.4x faster)
IKEA - Side Panel 2	1.75	79	**100** (+27%)	5.0	**2.4** (2.1x faster)
IKEA - Top Panel	1	35	**100** (+186%)	8.9	**2.4** (3.7x faster)
IKEA - Whole Assembly	–	1/10	**10/10** (+900%)	–	–
Car Dashboard Assembly	2	41	**100** (+144%)	20.3	**8.8** (2.3x faster)
Timing Belt Assembly	6	2	**100** (+4900%)	9.1	**7.2** (1.3x faster)

<- better, faster than imitation learning

Luo, Xu, Wu, Levine. HIL-SERL: Precise and Dexterous Robotic Manipulation via Human-in-the-Loop Reinforcement Learning. 2024.

PPO = Common algorithm of choice for stable, less efficient learning

Learning to walk in simulation with PPO -> transfer to real world (with careful simulation choices)

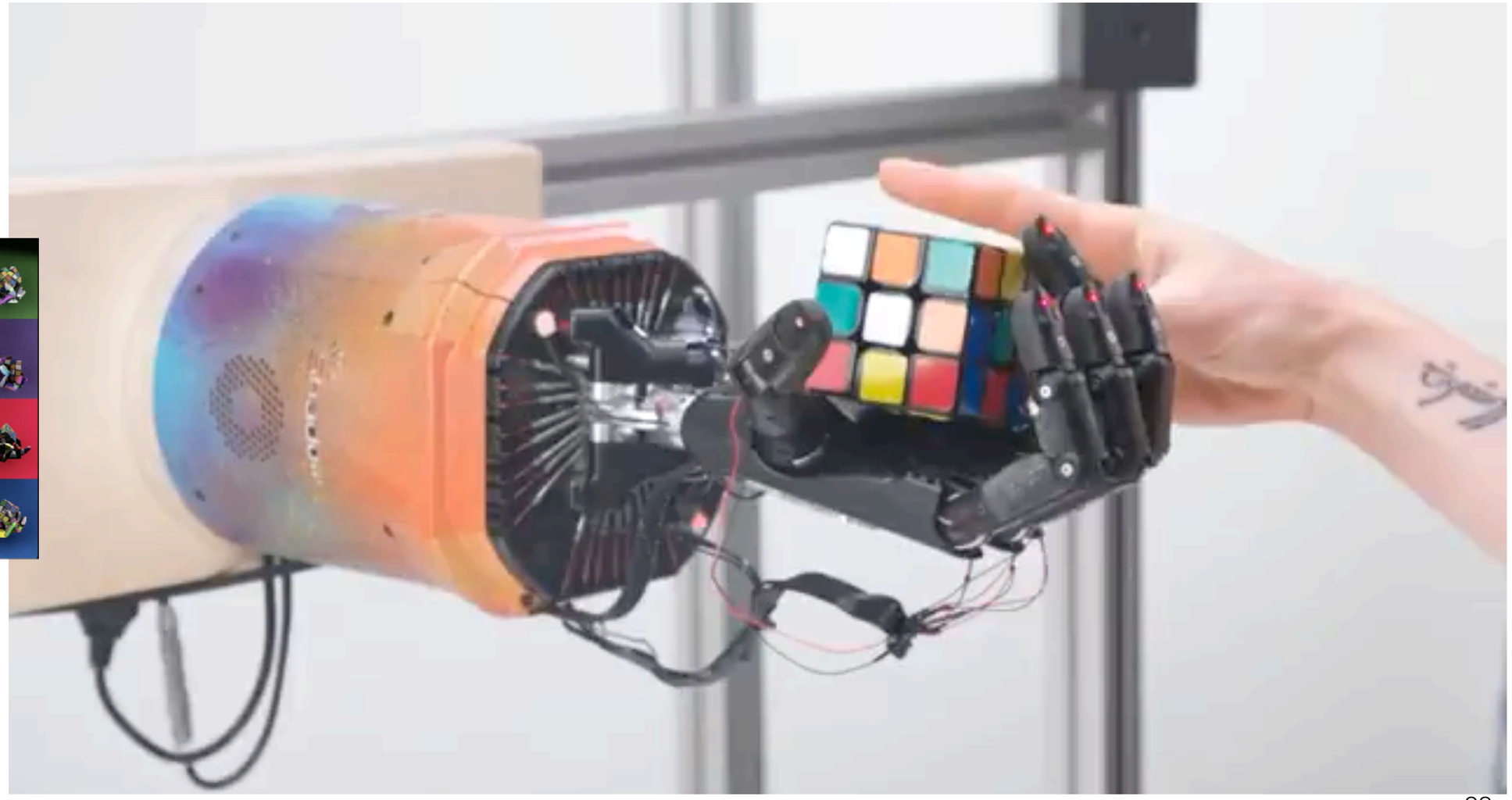


Tan, Zhang, Coumans, Iscen, Bai, Hafner, Bohez, Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. 2018.

PPO = Common algorithm of choice for stable, less efficient learning

Learning to solve Rubik's cube in simulation -> transfer to real hand

(though, transfer from simulation generally much harder, less successful for robot manipulation)



OpenAI. Solving Rubik's Cube with a Robot Hand. 2019.

What about RL for language models?

PPO is a common choice!

Will start to discuss them when talking about reward modeling next week.

Online RL practical guidelines so far

Proximal policy optimization (PPO)

Much more *on-policy*

(Multiple gradient steps on batch of roll-outs, then recollect)

Key benefit: stability, more "plug & play"

Key detriment: data inefficient

Can remove reliance on Markov property if use Monte Carlo value function

Soft actor-critic (SAC)

(And newer methods like RLPD, EXPO)

Much more *off-policy*

(Uses replay buffer of past experience)

Key benefit: data efficiency

Key detriment: less stable, often requires more tuning

Heavy reliance on Markov property

Both: helpful to seed with imitation/demonstrations if available

Initialize policy weights

Add demos to replay buffer

The plan for today

Off-policy actor critic methods

1. Taking multiple gradient steps
    a. Importance weights
    b. Constraining step size with KL penalty or clipping
    c. Practical PPO algorithm
2. Even more off-policy algorithm
    d. Fitting Q-value functions with data from other policies
    e. Practical SAC algorithm
3. Comparisons between PPO, SAC, imitation learning

Key learning goals:

- All of the key concepts for practical algorithms like PPO and SAC

Next Time

Q-learning (last big online RL method)

+

Practical implementation of Q-learning

Course reminders

- **Friday 4:45 pm**: Section on value functions, Q-learning
- Project survey due in one week
- **Homework 2** due next Friday