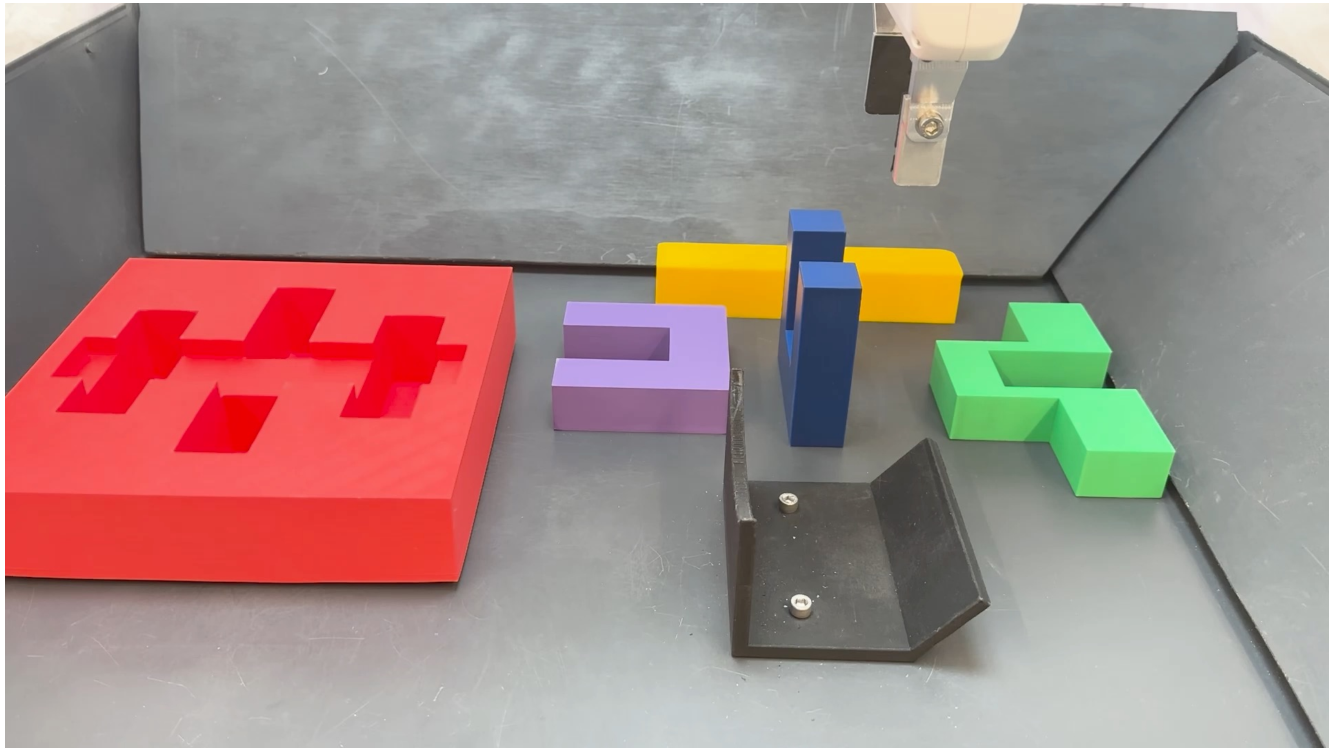
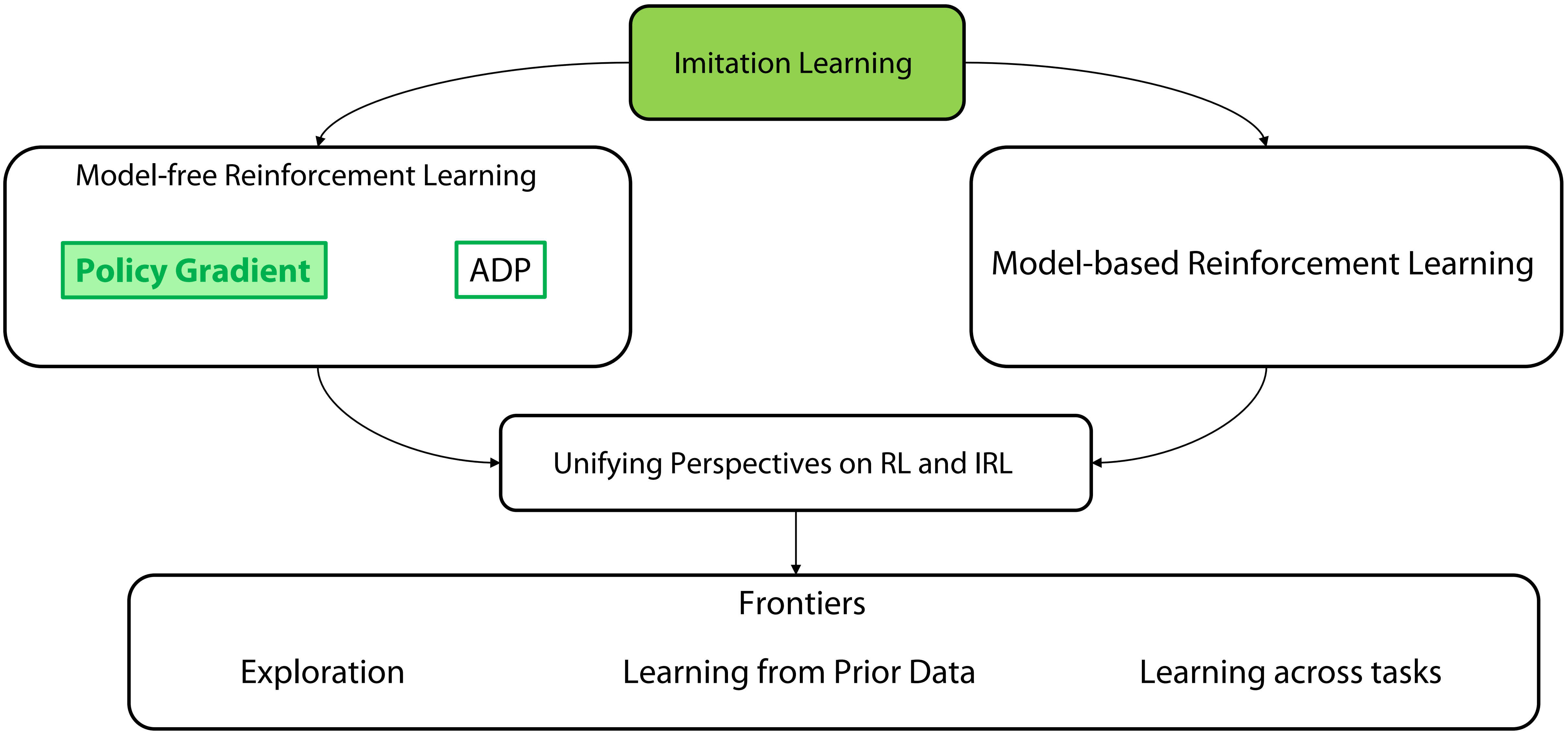
Reinforcement Learning
Spring 2024

Abhishek Gupta

TAs: Patrick Yin, Qiuyu Chen



Class Structure



Last lecture outline

Making NPG Practical → Trust Region Policy Optimization

↓

Reducing Variance of Critic with GAE

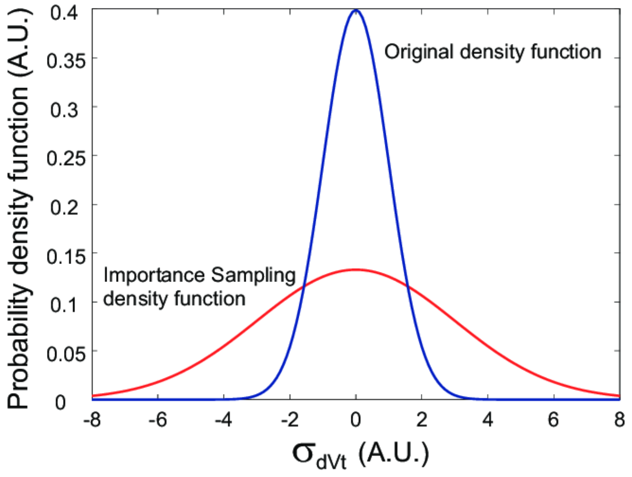
↓

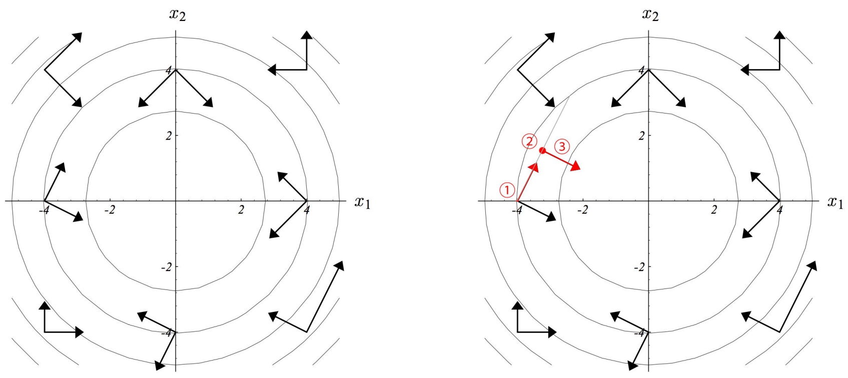
One Algorithm to Rule Them All - Proximal Policy Optimization

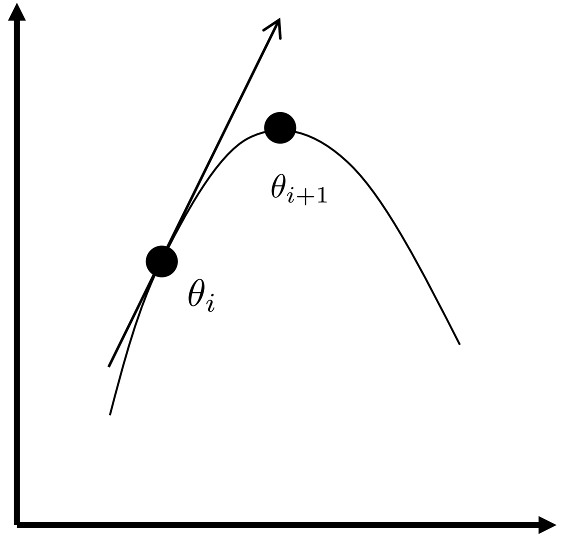
Trust Region Policy Optimization

3 key ideas:

1. On-policy updates → importance sampled objective
2. Huge matrix inversion → conjugate gradient method
3. Step size may be too large → backtracking line search







TRPO, Schulman '15

Generalized Advantage Estimation

Sum up all the estimators in a geometric sum

$$A^\theta_N(s_1, a_1) = r_1 + \gamma r_2 + \cdots + \gamma^{N-1} r_N - V(s_1)$$

$$A^\theta_{N-1}(s_1, a_1) = r_1 + \gamma r_2 + \cdots + \gamma^{N-2} V(s_{N-1}) - V(s_1)$$

$$A^\theta_2(s_1, a_1) = r_1 + \gamma r_2 + \cdots + \gamma^2 V(s_3) - V(s_1)$$

$$A^\theta_1(s_1, a_1) = r_1 + \gamma V(s_2) - V(s_1)$$

Geometric sum

$$A^\theta_\lambda(s_1, a_1) = \sum_{j=1}^{N} \lambda^j A^\theta_j(s, a)$$

λ controls bias-variance tradeoff

Best of both worlds – very similar idea to eligibility traces

Proximal Policy Optimization

$$\mathcal{L}(s, a, \theta_i, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_i}(a|s)} A(s,a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_i}(a|s)}, 1-\epsilon, 1+\epsilon\right) A(s,a)\right)$$

✓ Multiple minibatch gradient steps
✓ No second order optimization
✓ Simple and stable, without huge updates

Lecture outline

Kronecker Factorization (K-FAC)

↓

Frontiers of Policy Gradients

↓

Going from Monte Carlo Returns to Critic Estimation

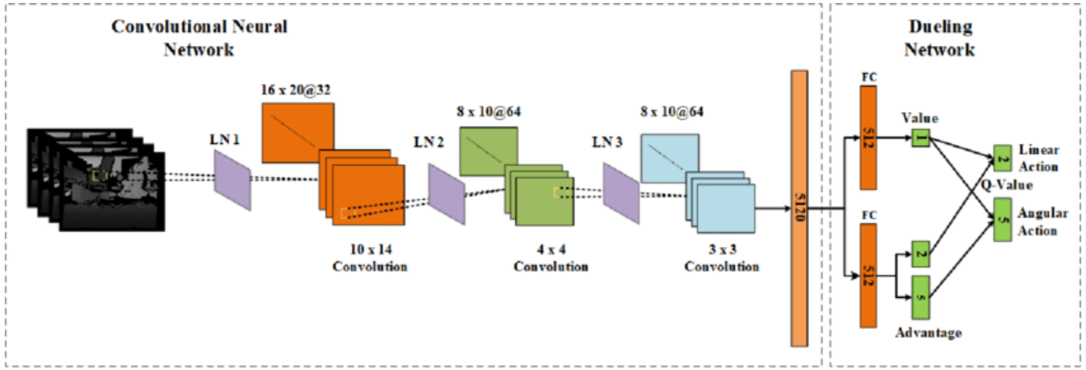
↓

Going from Monte Carlo Returns to Critic Estimation

Approximations to effectively invert FIM

Instead of solving with conjugate gradient, what if we approximated FIM

Major issue with FIM is huge dimensionality



Dimensionality of FIM – huge!
Very challenging to invert

Kronecker factorization (K-FAC) makes it tractable with 2 approx:

1. Block diagonalize FIM by layer
2. Represent each block diagonal as a Kronecker product (easy inversion)

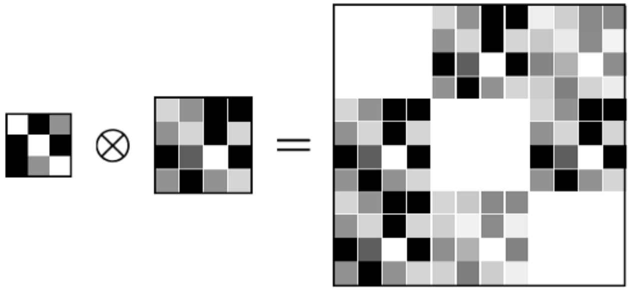
ACKTR, Wu ‘17

What is a Kronecker product?

Generalization of the tensor product

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} [\mathbf{A}]_{1,1}\mathbf{B} & \cdots & [\mathbf{A}]_{1,n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{m,1}\mathbf{B} & \cdots & [\mathbf{A}]_{m,n}\mathbf{B} \end{pmatrix} \in \mathbb{R}^{ma \times nb}$$

$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{a \times b}$: Kronecker factors

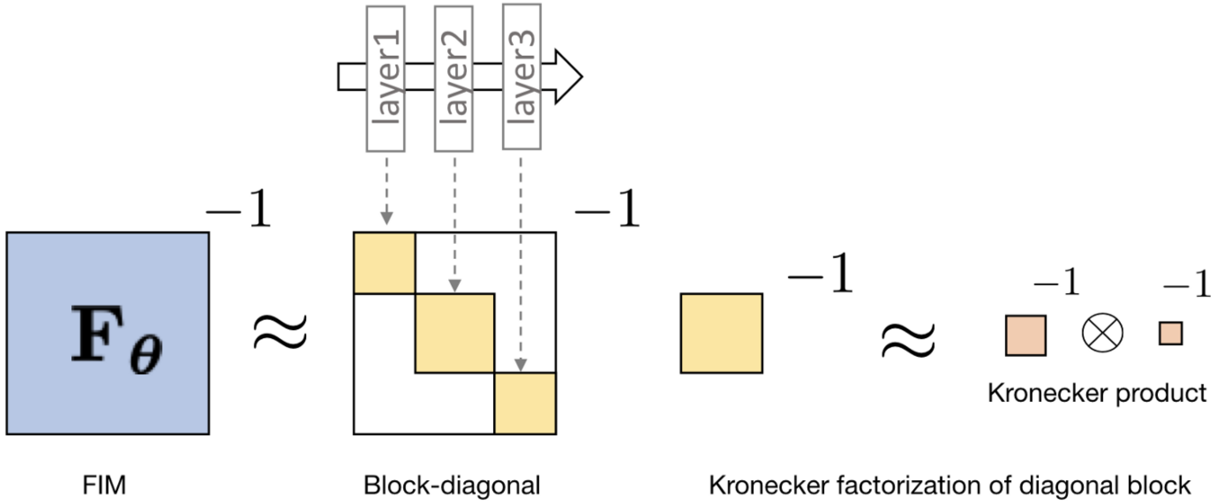


Identities

$$\mathrm{vec}(uv^T) = u \otimes v$$

$$(a \otimes b)(c \otimes d) = (ac \otimes bd)$$

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$$



We will represent the (per layer) fisher information metric as a Kronecker product of two smaller matrices

ACKTR, Wu ‘17

K-FAC approximation for NPG

Kronecker factorization (K-FAC) makes it tractable with 2 approx:

1. Block diagonalize FIM by layer
2. Represent each block diagonal as a Kronecker product (easy inversion)

Weight defs

$$s = Wa$$
$$\nabla_W L = (\nabla_s L) a^\intercal$$

Matrix Identities

$$\text{vec}(uv^T) = u \otimes v$$
$$(a \otimes b)(c \otimes d) = (ac \otimes bd)$$

K-FAC reparam

$$F_\ell = \mathbb{E}[\text{vec}\{\nabla_W L\}\text{vec}\{\nabla_W L\}^\intercal] = \mathbb{E}[aa^\intercal \otimes \nabla_s L(\nabla_s L)^\intercal]$$
$$\approx \mathbb{E}[aa^\intercal] \otimes \mathbb{E}[\nabla_s L(\nabla_s L)^\intercal] := A \otimes S := \hat{F}_\ell,$$

Easy to compute and invert (order of magnitude smaller matrices)

ACKTR, Wu '17

How much does this help?

All layers in AlexNet
60,000,000 parameters

Fisher information matrix

$$\mathbf{F}_{\boldsymbol{\theta}} \in \mathbb{R}^{60{,}000{,}000 \times 60{,}000{,}000}$$

Final layer of AlexNet
Input dimension: 4,096
Output dimension: 1,000
4,096,000 parameters

Fisher block

$$\mathbf{F}_i \in \mathbb{R}^{4{,}096{,}000 \times 4{,}096{,}000}$$

Kronecker factors

$$\mathbf{A}_{i-1} \in \mathbb{R}^{4{,}096 \times 4{,}096}$$

$$\mathbf{G}_i \in \mathbb{R}^{1{,}000 \times 1{,}000}$$

ACKTR, Wu '17

Lecture outline

Kronecker Factorization (K-FAC)

↓

Frontiers of Policy Gradients

↓

Going from Monte Carlo Returns to Critic Estimation

↓

Going from Monte Carlo Returns to Critic Estimation

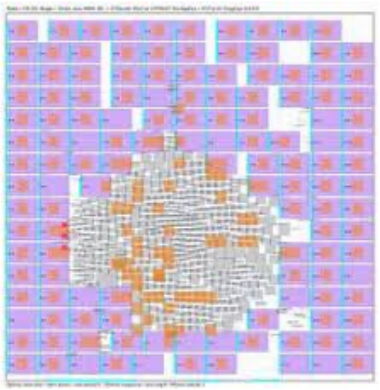
Pros/Cons of Policy Gradient Methods

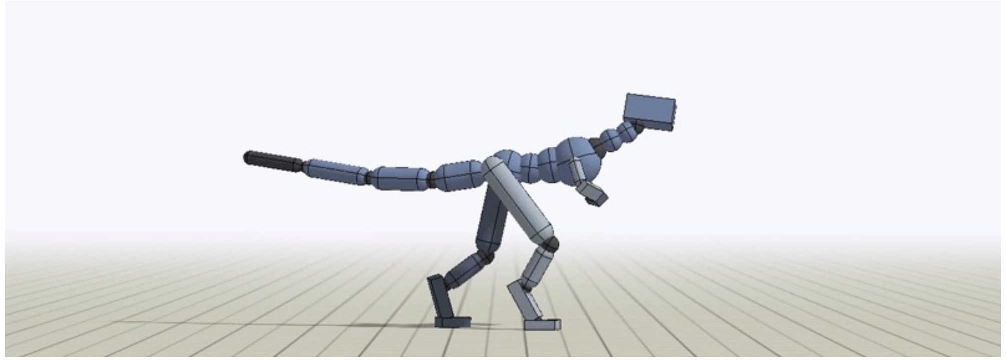
Pros

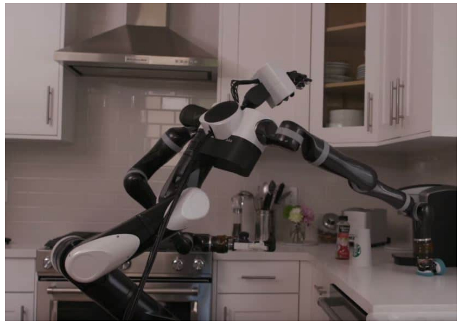
- Conceptually simple, easy to implement
- Stable, good asymptotic performance
- Compatible with deep models
- Require minimal modeling

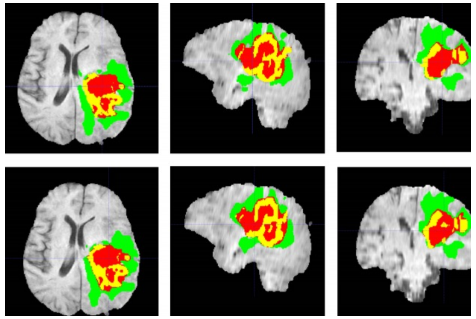
Cons

- Sample inefficient
- Unable to reuse prior data effectively → on-policy
- Blackbox, can be hard to debug









Frontiers of Policy Gradient Research

Major open challenges in policy gradient research:

- Convergence guarantees
- Asynchronous/Parallel Methods
- Better Variance Reduction
- Learning from high-dimensional inputs
- Bootstrapping from prior data
- Multi-agent Policy Gradient

Frontiers of Policy Gradient Research

Convergence guarantees and empirical investigations

Globally Convergent in LQR/LQG Case

- *Gradient descent case: For an appropriate (constant) setting of the stepsize* η,

$$\eta = \text{poly}\left(\frac{\mu\sigma_{min}(Q)}{C(K_0)}, \frac{1}{\|A\|}, \frac{1}{\|B\|}, \frac{1}{\|R\|}, \sigma_{min}(R)\right)$$

and for

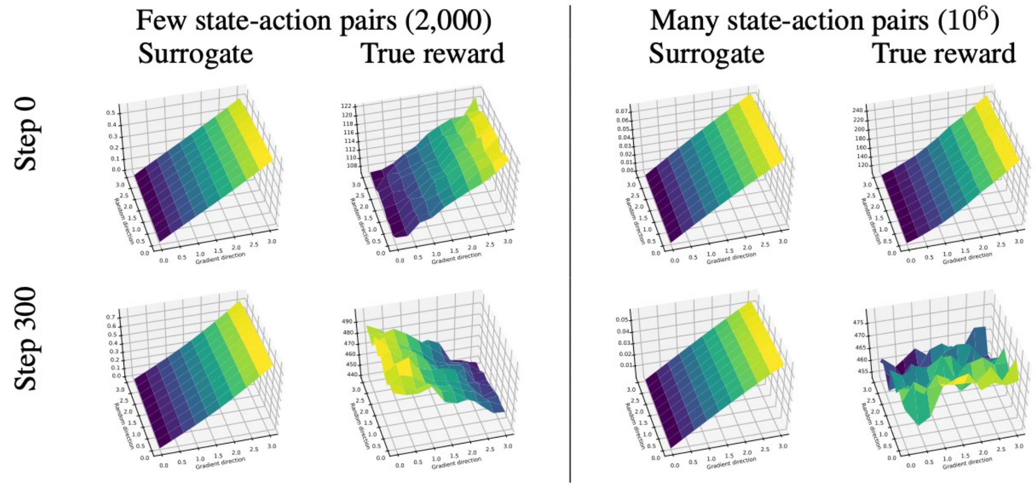
$$N \geq \frac{\|\Sigma_{K^*}\|}{\mu} \log \frac{C(K_0) - C(K^*)}{\varepsilon} \times \text{poly}\left(\frac{C(K_0)}{\mu\sigma_{min}(Q)}, \|A\|, \|B\|, \|R\|, \frac{1}{\sigma_{min}(R)}\right),$$

then, with high probability, gradient descent (Equation 8) enjoys the following performance bound:

$$C(K_N) - C(K^*) \leq \varepsilon \,.$$

Global Convergence of Policy Gradient Methods for the Linear Quadratic Regulator, Fazel et al '19
Global Convergence of Policy Gradient Methods to (Almost) Locally Optimal Policies, Zhang et al, '19
Globally convergent policy search over dynamic filters for output estimation, Umenberger '21

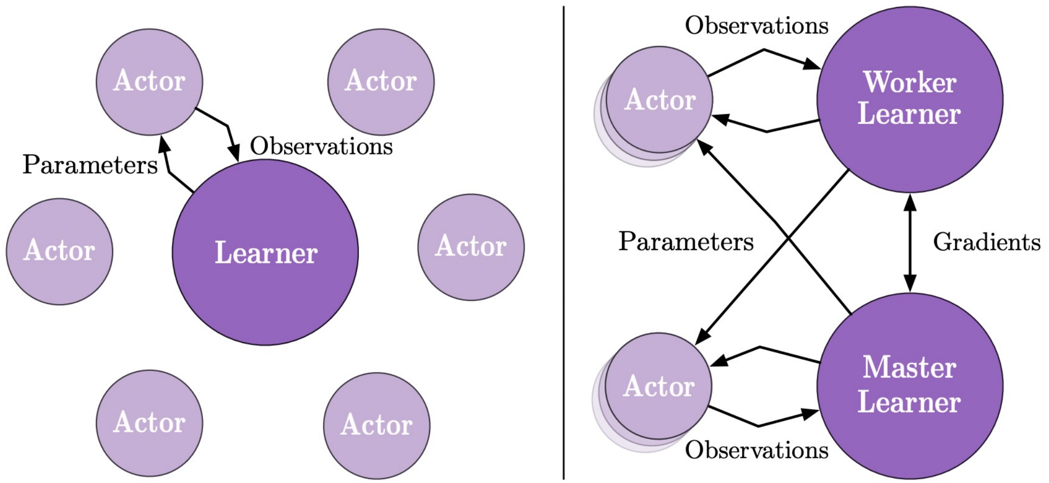
Practical Algorithms Deviate from Theory



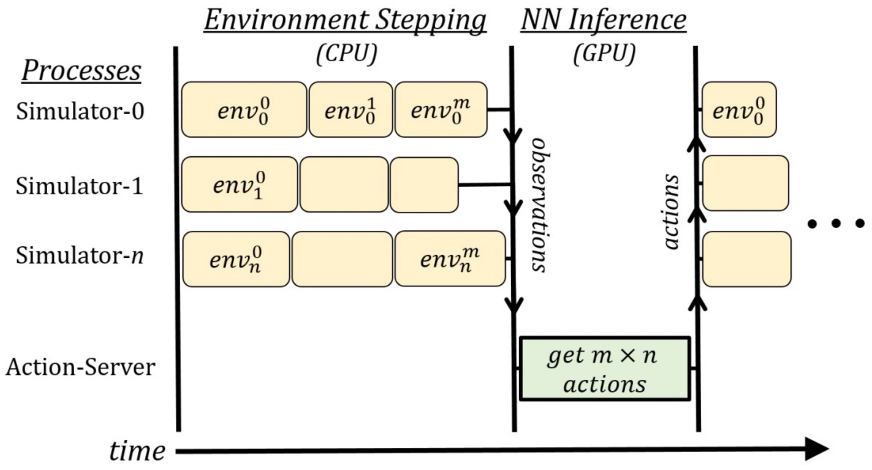
Is the Policy Gradient a Gradient?, Nota et al, '19
A Closer Look at Deep Policy Gradients, Ilyas et al '19
An Empirical Analysis of Proximal Policy Optimization with Kronecker-factored Natural Gradients, Song et al '18
What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study, Andrychowicz et al '20

Frontiers of Policy Gradient Research

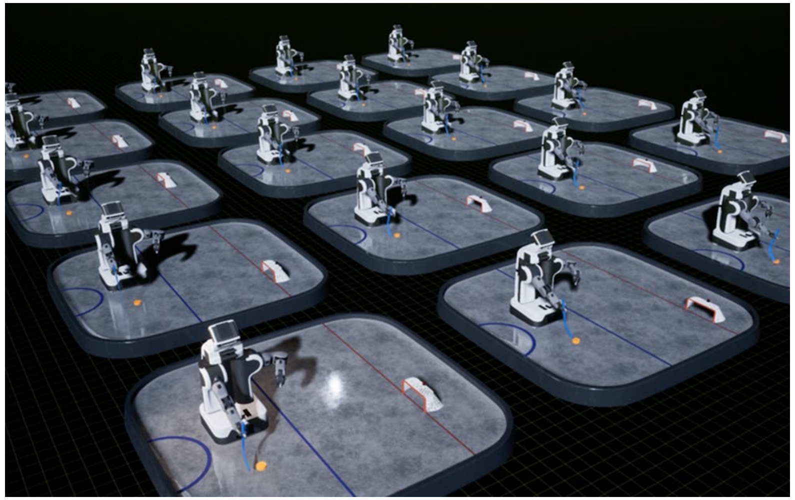
Asynchronous methods for large scale speedup



IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures, Espeholt '18



Accelerated Methods for Deep Reinforcement Learning, Stooke et al '19



Frontiers of Policy Gradient Research

Better Variance Reduction Methods

Action dependent baselines

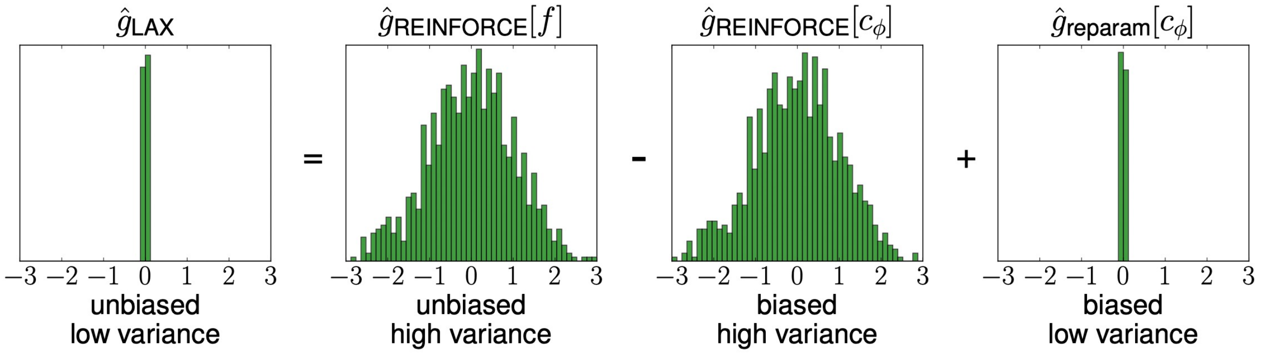
$$\pi_\theta(a_t|s_t) = \prod_{i=1}^{m} \pi_\theta(a_t^i|s_t)$$

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi,\pi}\left[\sum_{i=1}^{m} \nabla_\theta \log \pi_\theta(a_t^i|s_t)\left(\hat{Q}(s_t,a_t) - b_i(s_t,a_t^{-i})\right)\right]$$

For factorized spaces, baselines can depend on independent action factors

The Mirage of Action-Dependent Baselines in Reinforcement Learning, Tucker et al '18
Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines, Wu et al '18

Alternative Estimators



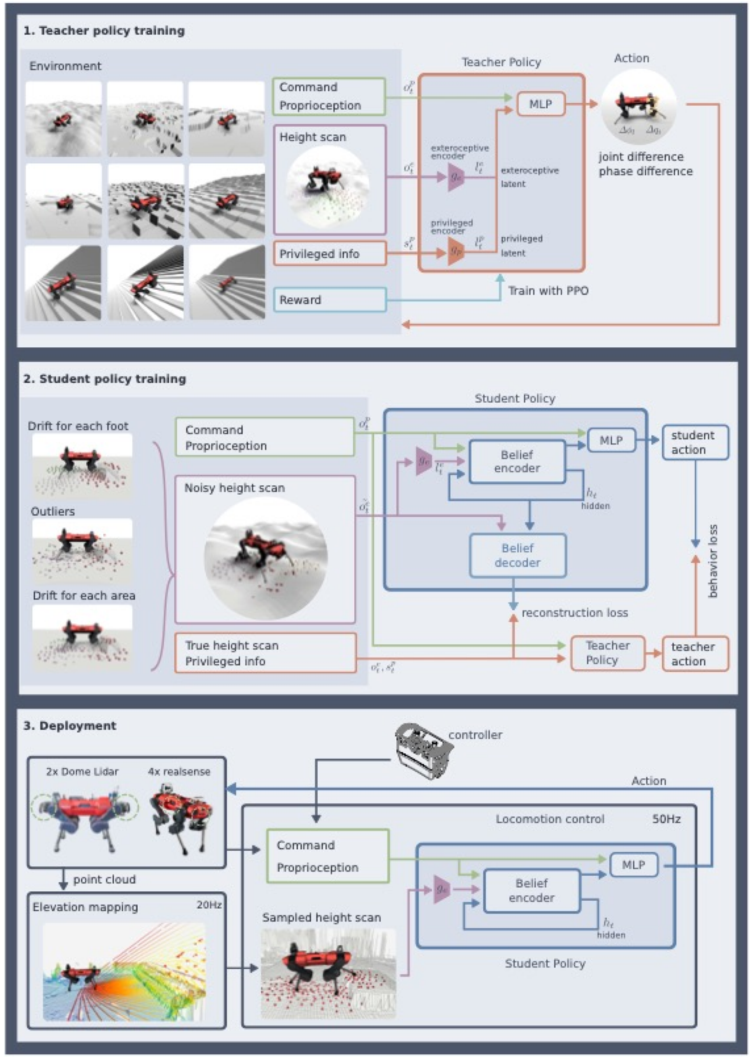
Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic, Gu et al '16
Backpropagation through the Void: Optimizing control variates for black-box gradient estimation, Grathwohl et al '17
Categorical Reparameterization with Gumbel-Softmax, Jang et al '16

Frontiers of Policy Gradient Research

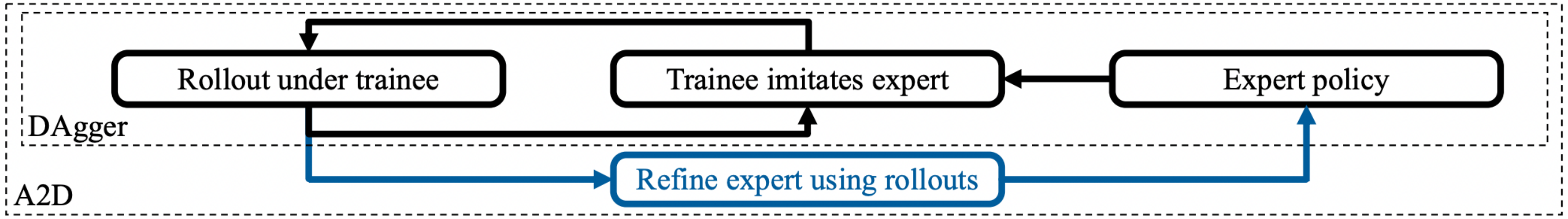
Learning from High Dimensional Observations

Learning Quadrupedal Locomotion over Challenging Terrain, Lee et al '20

A System for General In-Hand Reorientation, Chen et al '21



Challenging to provide guarantees in partially observed settings!



Robust Asymmetric Learning in POMDPs, Warrington et al '20

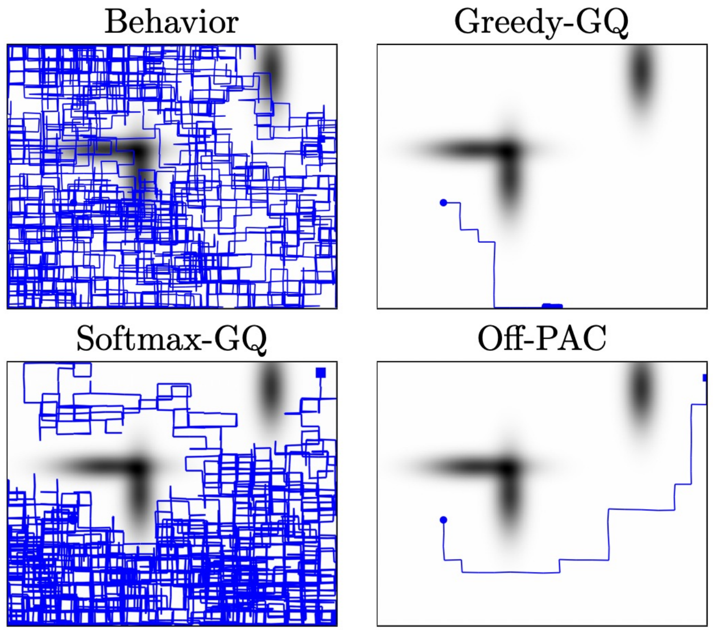
Learning robust perceptive locomotion for quadrupedal robots in the wild, Miki et al '22

Frontiers of Policy Gradient Research

Bootstrapping from Prior/Off-Policy Data

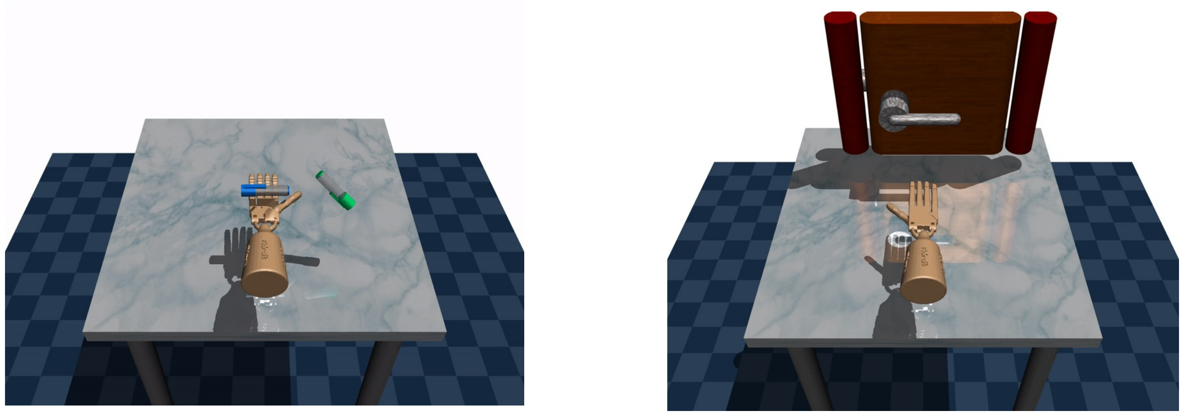
Off-policy policy gradient

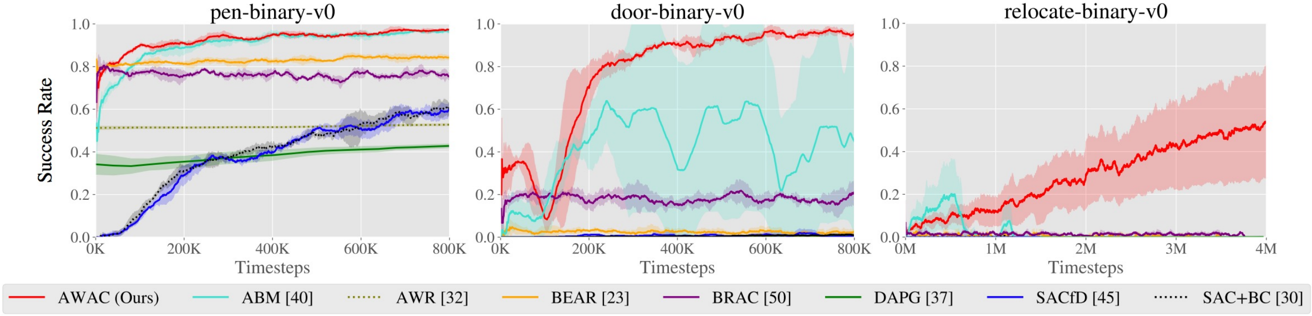
$$\mathbb{E}_\beta\left[\frac{\pi_\theta(a|s)}{\beta(a|s)} Q^\pi(s,a) \nabla_\theta \ln \pi_\theta(a|s)\right]$$



Off-Policy Actor-Critic, Degris et al '13

Learning from Prior Data

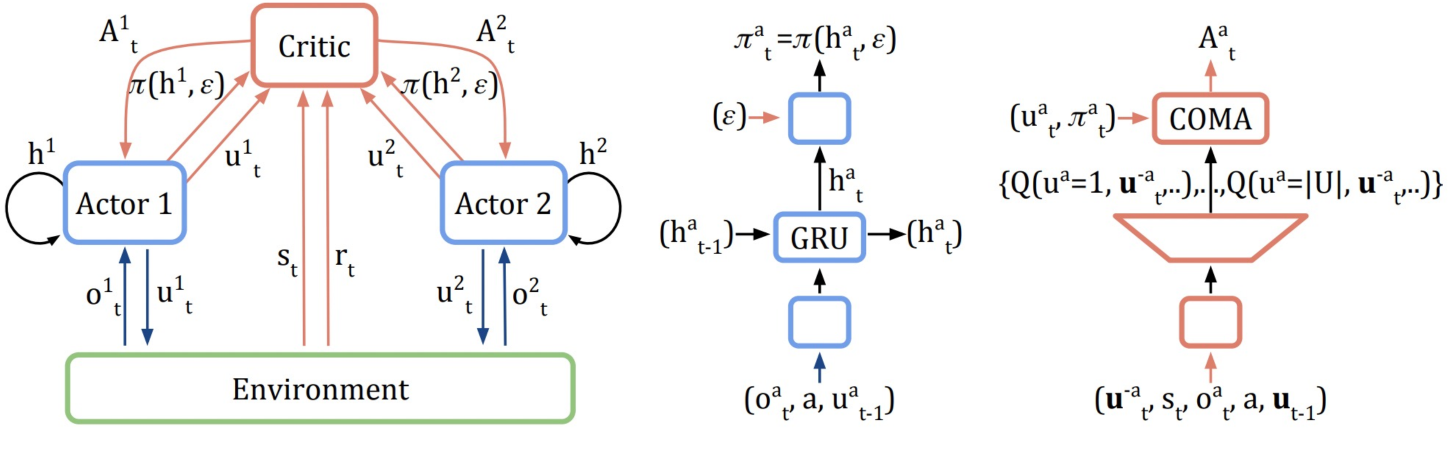




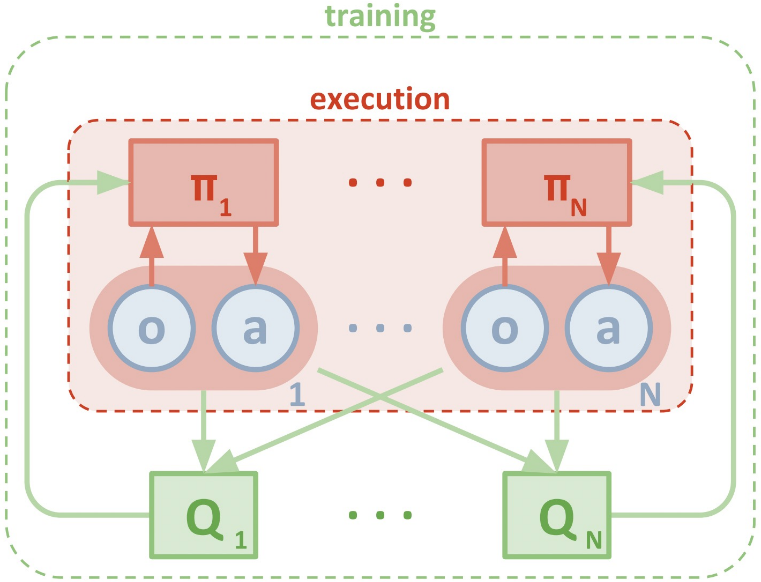
Advantage Weighted Actor Critic, Nair et al '20
DDPGfD, Vecerik '17
DAPG, Rajeswaran '17

Frontiers of Policy Gradient Research

Multi-agent policy gradient



Counterfactual Multi-Agent Policy Gradients, Foerster et al '17



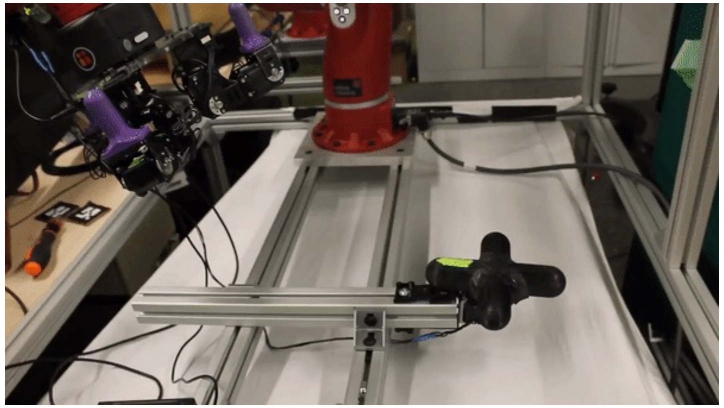
Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments, Lowe et al '17

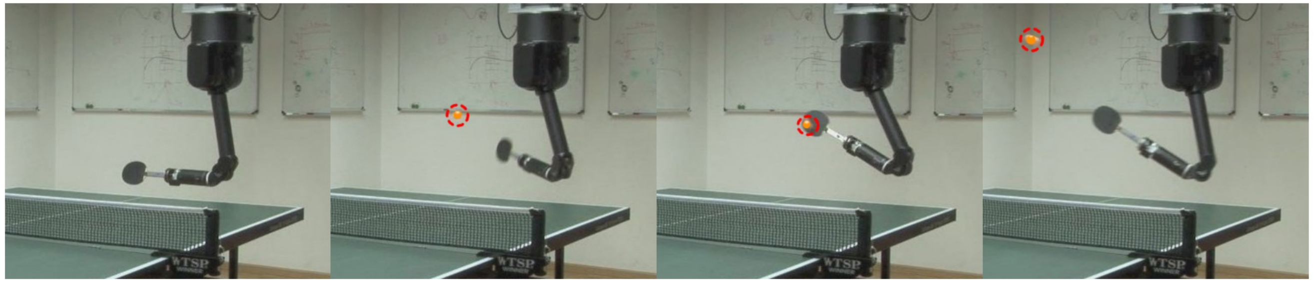
Primary challenges:

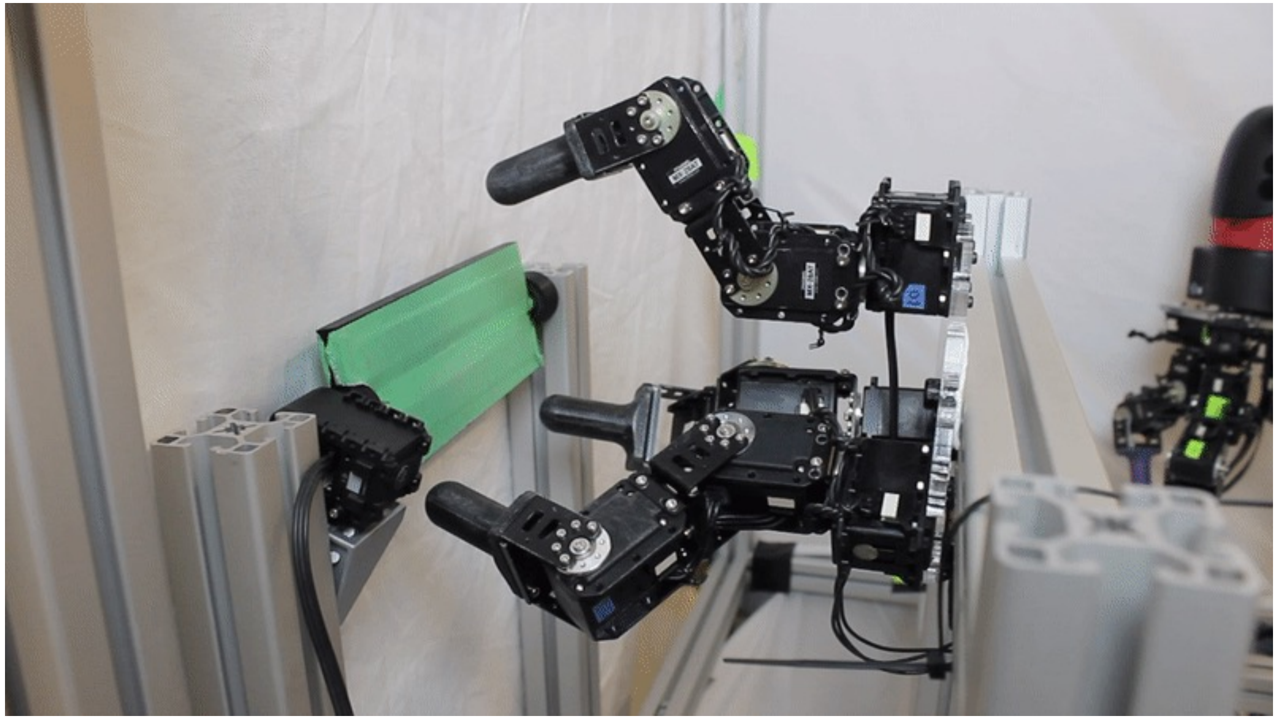
1. Non-stationarity
2. Data-efficiency
3. Communication

How is this useful for robotics?

Can be used to train robots in the real world but only in limited settings

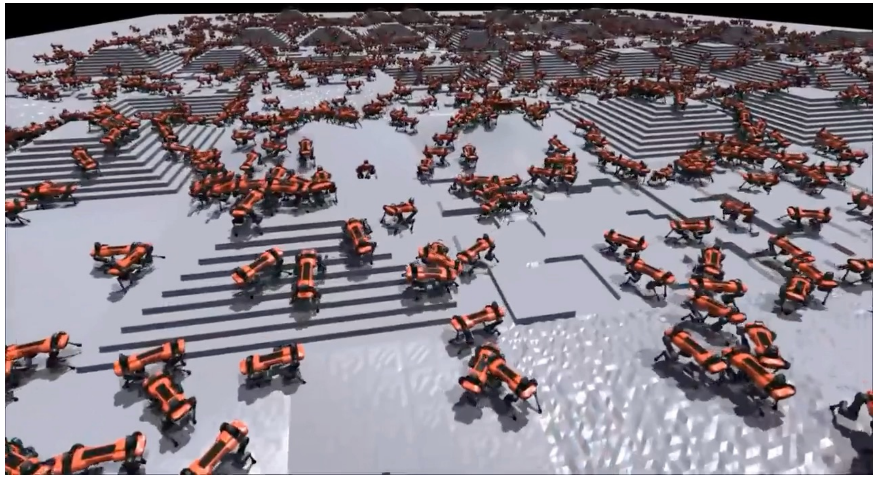


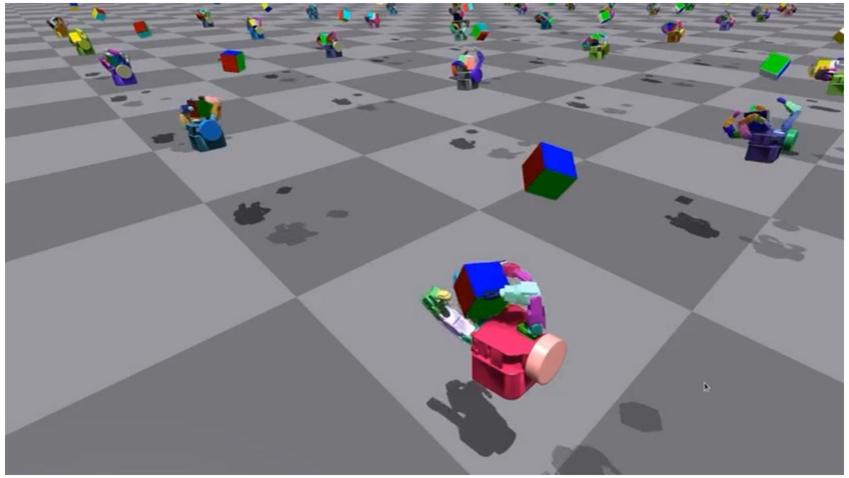


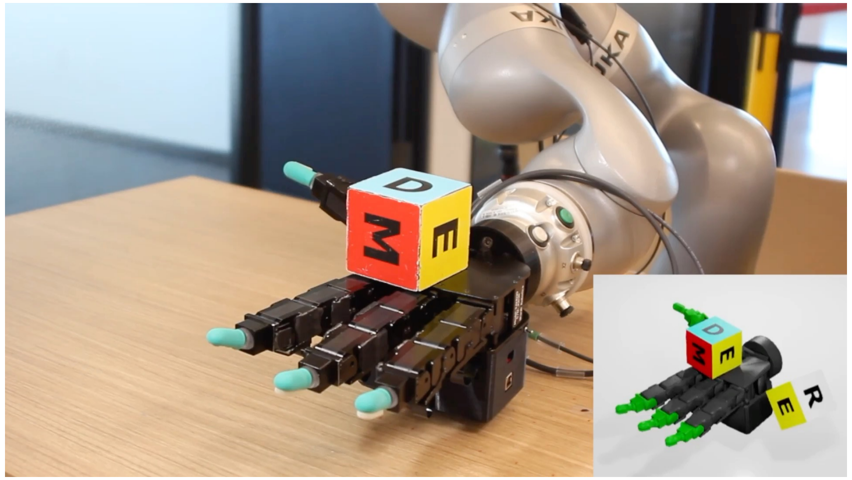


How is this useful for robotics?

Largely useful for pretraining in simulation







Lecture outline

Kronecker Factorization (K-FAC)

↓

Frontiers of Policy Gradients

↓

Going from Monte Carlo Returns to Critic Estimation

↓

Going from Monte Carlo Returns to Critic Estimation

What can we do to make PG suitable for robots?

Why is Policy Gradient sample inefficient?

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=t}^{T} r(s_t^i, a_t^i)$$

On-policy, unable to effectively use past data

High Variance Estimator

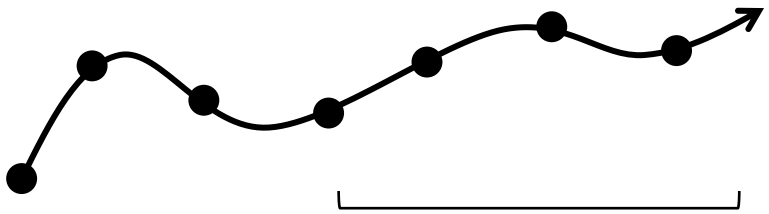
Can we develop a **<u>low variance</u>** **<u>off-policy</u>** RL algorithm that can bootstrap from prior data?

What can we do to lower variance?

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau$$

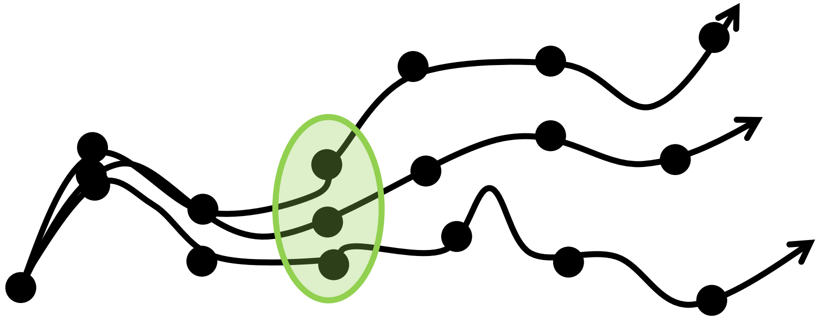
$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=t}^{T} r(s_t^i, a_t^i)$$

What we do



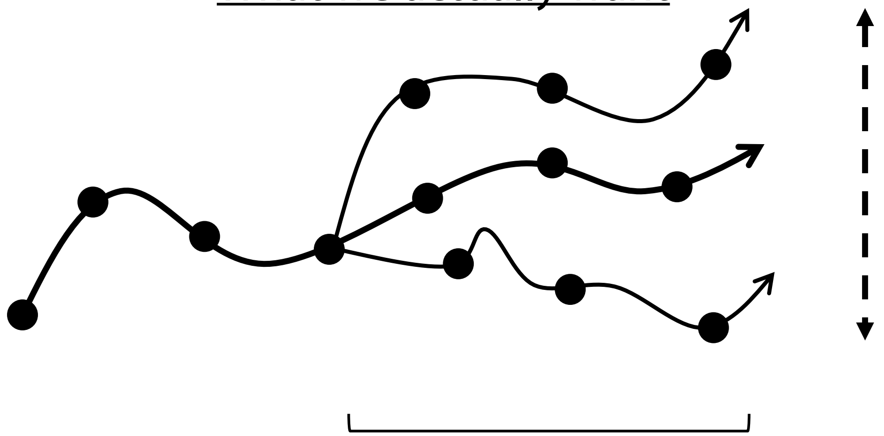
Single sample estimate

Idea: bundle this across many (s, a) with a function approximator



Function approximator bundles return estimates across states

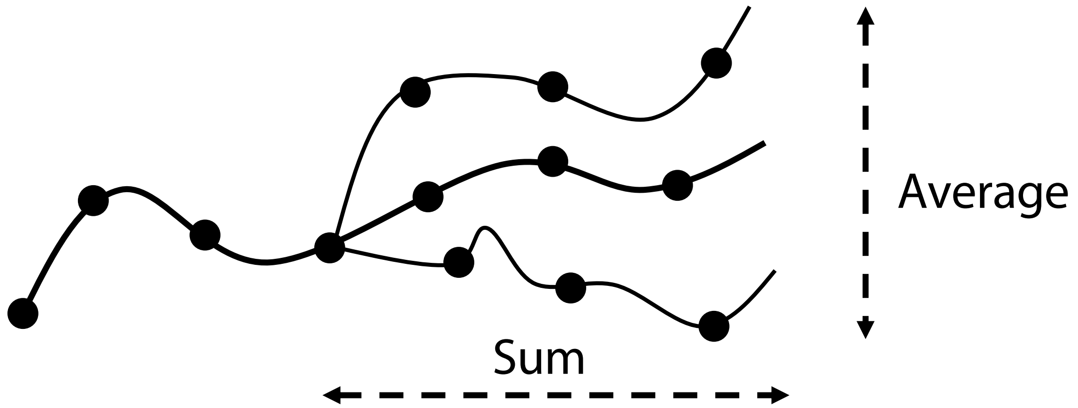
What we actually want



Averaged return estimate

Notation: Q functions

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=t}^{T} r(s_t^i, a_t^i)$$



Expected sum of rewards in the future, starting from (s, a) on first step, then π

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

Bundles estimates across (s, a)

Use the magic of (deep) function approximation

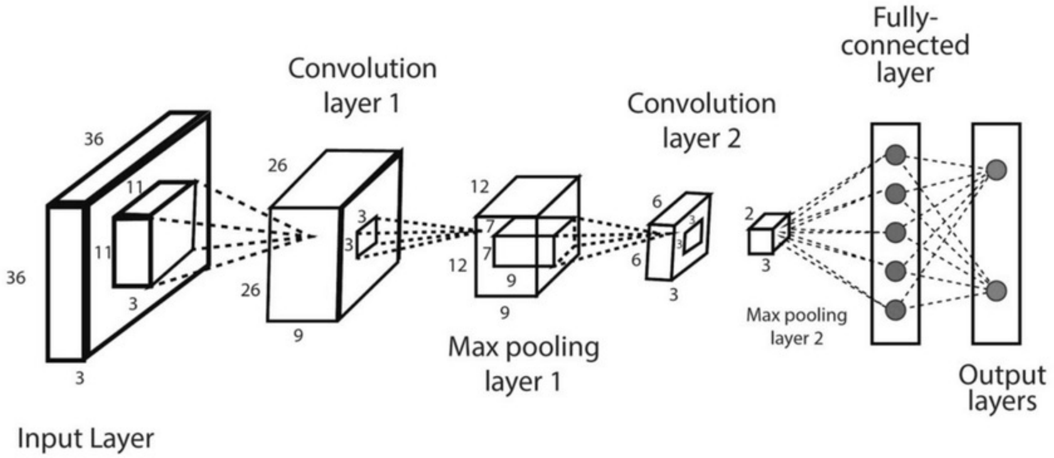
Attempt 0: Monte-Carlo Estimation of Q-Functions

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)Q^\pi(s_{t'}^i, a_{t'}^i)$$

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s_t', a_t')|s_t, a_t\right]$$ ← Monte-carlo approximation

Idea: Regression from (s, a) to Monte-Carlo estimate

State
Action



Return to Go

Unbiased, but high variance!

Can we do better?

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=t}^{T} r(s_t^i, a_t^i)$$

⇩

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i) Q^\pi(s_t^i, a_t^i)$$

Much lower variance if estimated well

Can be learned off-policy!

Has special structure we can exploit!!

Attempt 1: Using Recursive Structure

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)Q^\pi(s_t^i, a_t^i)$$

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

Note the definition of a value function $V^\pi(s_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'})|s_t\right] = \mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}\left[Q(s_t, a_t)\right]$

Average Q-function over actions sampled from policy

Value functions are recursive

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

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

$$V^\pi(s_t) = \mathbb{E}_{\pi_\theta}\left[r(s_t, a_t) + V^\pi(s_{t+1})\right]$$

Attempt 1: Using Recursive Structure

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)Q^\pi(s_t^i,a_t^i)$$

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

Value functions are recursive

$$V^\pi(s_t) = \mathbb{E}_{\pi_\theta}\left[r(s_t,a_t) + V^\pi(s_{t+1})\right]$$

Recipe for policy gradient

$$\min_\phi \mathbb{E}_{(s_i,a_i,s_i{}')\sim\pi}\left[(V_\phi^\pi(s_i) - y_i)^2\right]$$
$$y_i = r(s_i,a_i) + V(s_i{}')$$

Value Bellman equation

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)(\underbrace{r(s_t,a_t) + V(s_{t+1})}_{\text{Better estimate of future return}} - V(s_t))$$

Attempt 1: Using Recursive Structure

TODO replace this $$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i) Q^\pi(s_{t'}^i, a_{t'}^i)$$

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

Fit a value function on on-policy data

$$\min_\phi \mathbb{E}_{(s_i, a_i, s_i{}')\sim\pi}\left[(V_\phi^\pi(s_i) - y_i)^2\right]$$

$$y_i = r(s_i, a_i) + V(s_i{}')$$

Compute the policy gradient $$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)(r(s_t, a_t) + V(s_{t+1}) - V(s_t))$$

Collect more data

+ lowers variance

- Still on-policy

Revisit: Generalized Advantage Estimation

Sum up all the estimators in a geometric sum

$$A_N^\theta(s_1, a_1) = r_1 + \gamma r_2 + \cdots + \gamma^{N-1} r_N - V(s_1)$$

$$A_{N-1}^\theta(s_1, a_1) = r_1 + \gamma r_2 + \cdots + \gamma^{N-2} V(s_{N-1}) - V(s_1)$$

$$A_2^\theta(s_1, a_1) = r_1 + \gamma r_2 + \cdots + \gamma^2 V(s_3) - V(s_1)$$

$$A_1^\theta(s_1, a_1) = r_1 + \gamma V(s_2) - V(s_1)$$

Geometric sum

$$A_\lambda^\theta(s_1, a_1) = \sum_{j=1}^{N} \lambda^j A_j^\theta(s, a)$$

λ controls bias-variance tradeoff

Best of both worlds – very similar idea to eligibility traces

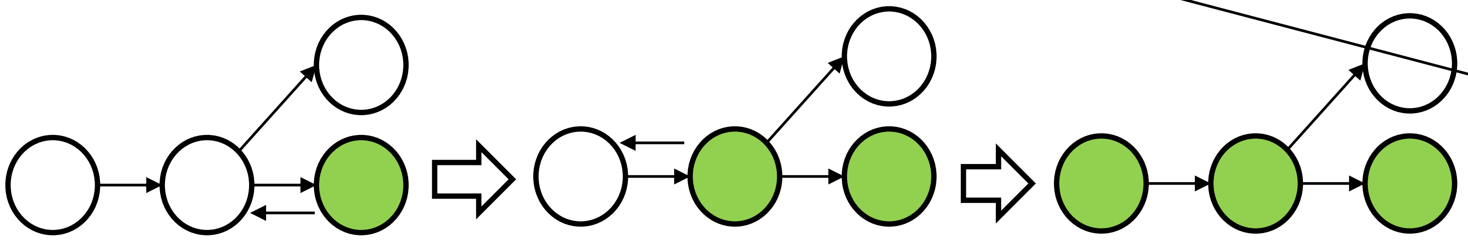
Attempt 2: Recursive structure in Q functions directly

Q functions have special recursive structure themselves!

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

$$= r(s_t, a_t) + \mathbb{E}_\pi\left[\sum_{t'=t+1} r(s_{t'}, a_{t'}) | s_{t+1}, a_{t+1} \sim \pi(.|s_{t+1})\right]$$

Bellman equation $Q^\pi(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{\substack{s_{t+1} \sim p(.|s_t, a_t) \\ a_{t+1} \sim \pi_\theta(.|s_{t+1})}} [Q^\pi(s_{t+1}, a_{t+1})]$



Can be from different policies

Decompose temporally via dynamic programming

Off-policy!

Learning Q-functions via Dynamic Programming

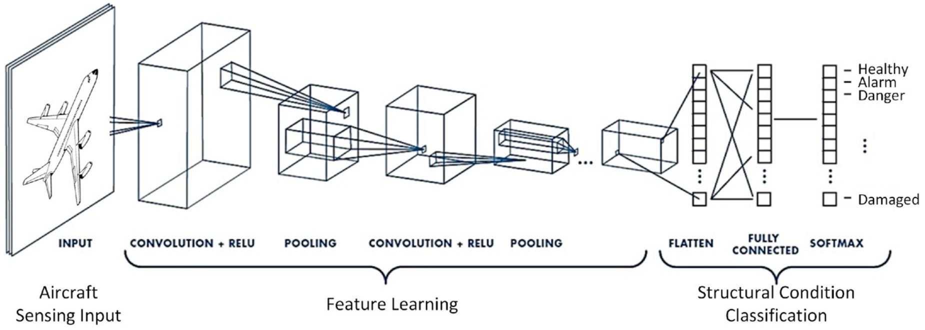
Policy Evaluation: Try to minimize Bellman Error (almost)

Bellman equation

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{\substack{s_{t+1} \sim p(.|s_t, a_t) \\ a_{t+1} \sim \pi_\theta(.|s_{t+1})}} [Q^\pi(s_{t+1}, a_{t+1}]$$

Same function approximator

How can we convert this recursion into an off-policy learning objective?

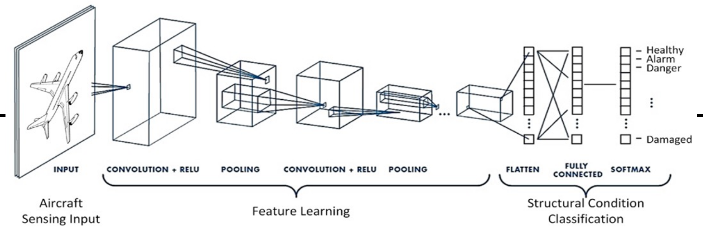


Why is this not just the gradient of the Bellman Error?

$$\min_{\phi} \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}} \left(Q^{\pi}_{\phi}(s_t, a_t) - \left(r(s_t, a_t) + \mathbb{E}_{a_{t+1}\sim\pi_\theta(a_{t+1}|s_{t+1})} \left[Q^{\pi}_{\hat{\phi}}(s_{t+1}, a_{t+1}) \right] \right) \right)^2$$

Approximate using stochastic optimization

$$\min_{\phi} \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}} \left(Q^{\pi}_{\phi}(s_t, a_t) - \left(r(s_t, a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1}, a_{t+1})\right)\right)^2 \quad a_{t+1} \sim \pi(\cdot|s_{t+1})$$



"Target" moves too much

Often tough empirically with function approximators

Expectation inside the square, hard to be unbiased

Note: this may look like gradient descent on Bellman error, it is not!

Improving Policies with Learned Q-functions

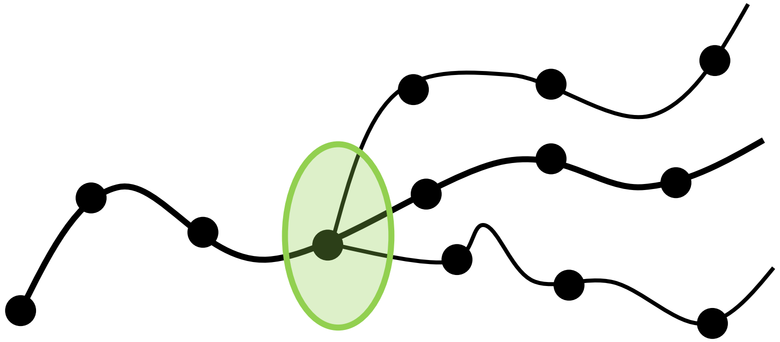
<u>Policy Improvement:</u> Improve policy with **<u>policy gradient</u>**

$$\max_{\theta} \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta(a|s)}\left[Q^{\pi_\theta}(s,a)\right]$$

Replace Monte-Carlo sum of rewards with learned Q function

Lowers variance compared to policy gradient!

+ off-policy



Policy Updates – REINFORCE or Reparameterization

Let's look a little deeper into the policy update

$$\max_{\theta} J(\theta) = \max_{\theta} \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{a \sim \pi_\theta(.|s)} \left[Q^\pi(s,a) \right]$$

Likelihood Ratio/Score Function

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{a \sim \pi_\theta(.|s)} \left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a) \right]$$

Easier to Apply to Broad Policy Class

Pathwise derivative/Reparameterization

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{z \sim p(z)} \left[\nabla_a Q^\pi(s,a)|_{a=\mu_\theta + z\sigma_\theta} \nabla_\theta(\mu_\theta + z\sigma_\theta) \right]$$

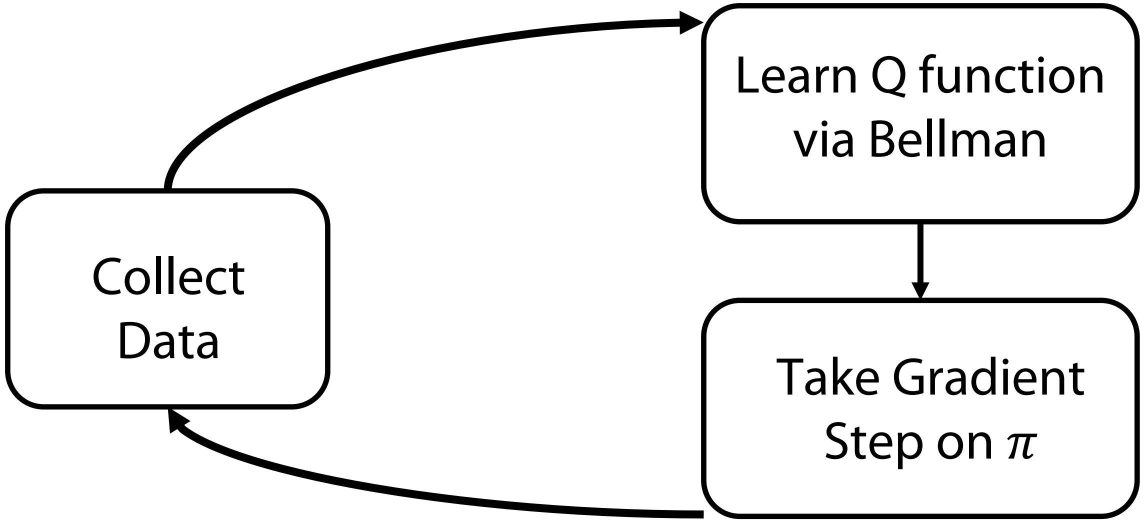
Lower variance (empirically)

Remember Lecture 2 and discussion of when gradients can be moved inside

Actor-Critic: Policy Gradient in terms of Q functions

Critic: learned via the Bellman update (Policy Evaluation)

$$\min_{\phi} \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left(Q^{\pi}_{\phi}(s_t, a_t) - (r(s_t, a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1}, a_{t+1})) \right)^2 \quad a_{t+1} \sim \pi(\cdot | s_{t+1})$$

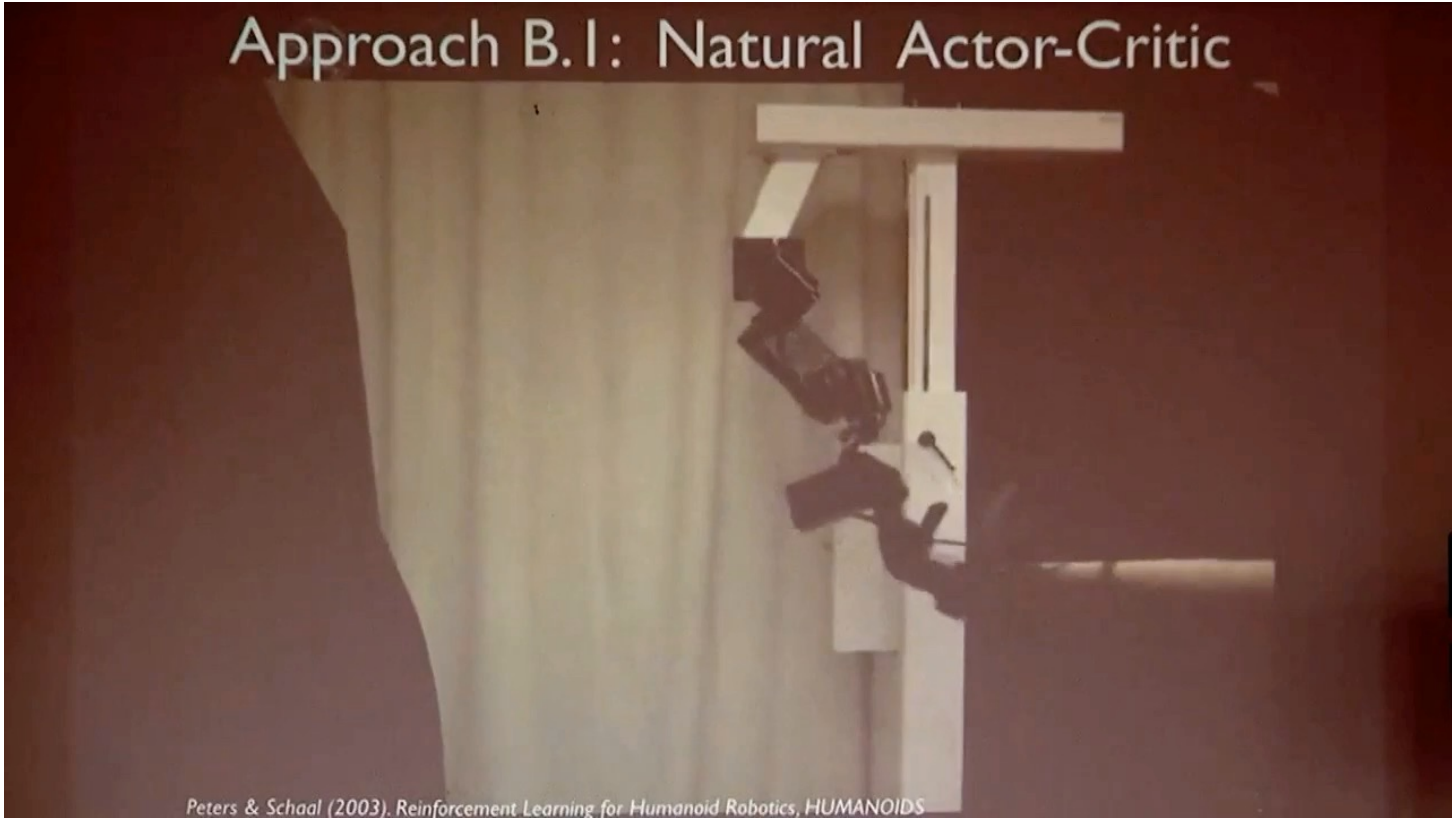


Lowers variance and is off-policy!

Actor: updated using learned critic (Policy Improvement)

$$\max_{\pi} \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{a \sim \pi(.|s)} \left[Q^{\pi}(s, a) \right]$$

Actor-Critic in Action



Lecture outline

Kronecker Factorization (K-FAC)

↓

Frontiers of Policy Gradients

↓

Going from Monte Carlo Returns to Critic Estimation

↓

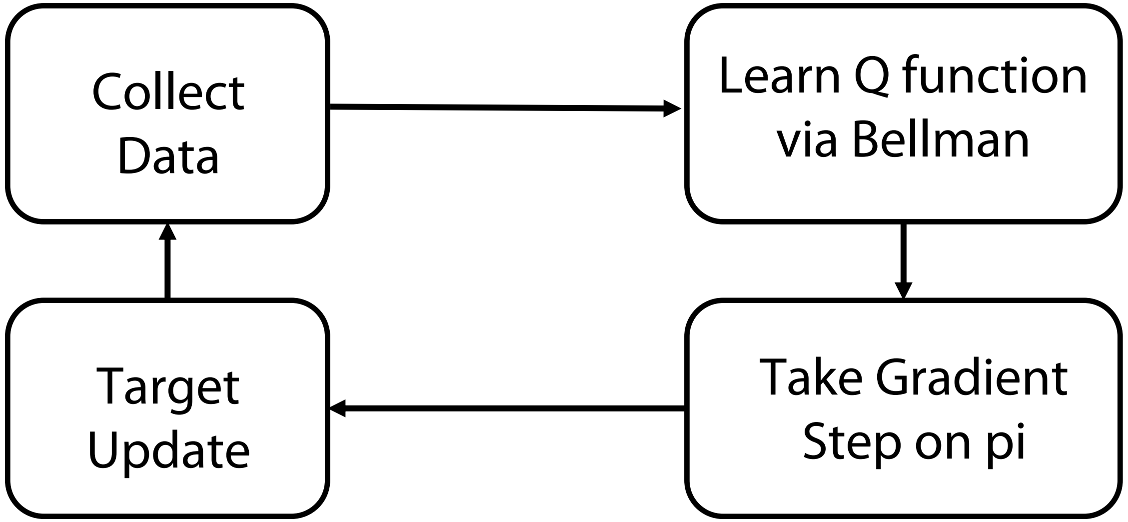
Getting Actor Critic to Work in Practice

What can we do to make off-policy algorithms work in practice?

Going from Batch Updates to Online Updates

This algorithm can go from full batch mode to fully online updates

1 sample



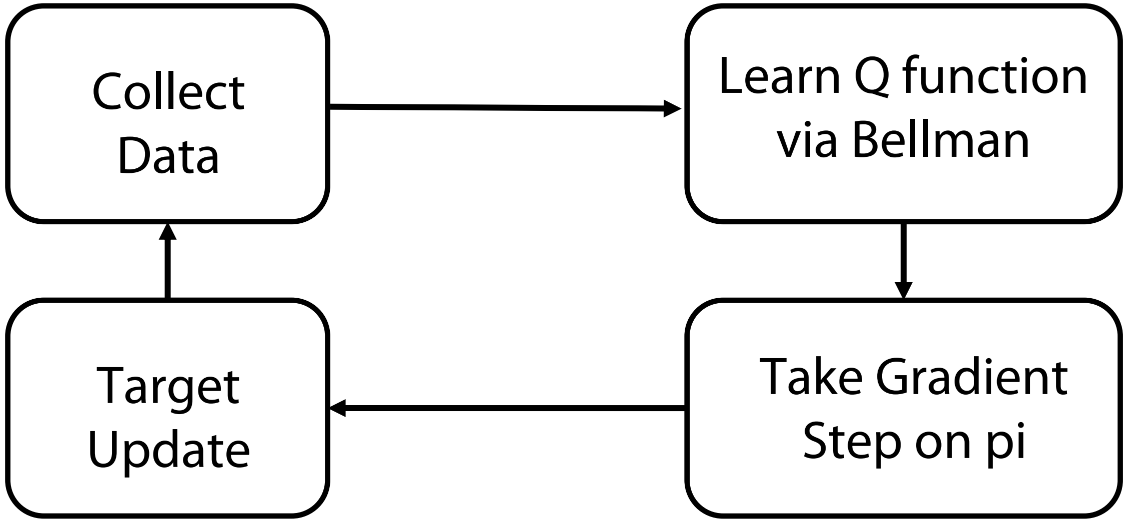
Critic: 1 gradient step on Bellman error

Actor: 1 gradient step on policy gradient

Allows for much more immediate updates

Challenges of doing online updates

1 sample



Critic: 1 gradient step on Bellman error

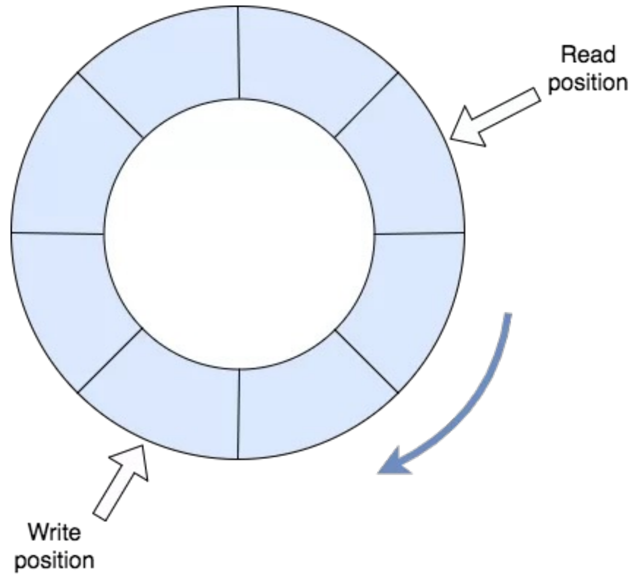
Actor: 1 gradient step on policy gradient

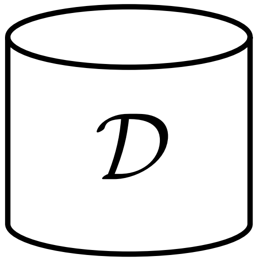
When updates are performed online, two issues persist:

1. Correlated updates since samples are correlated
2. Optimization objective changes constantly, unstable

Decorrelating updates with replay buffers

Updates can be decorrelated by storing and shuffling data in a replay buffer





Instead of doing updates in order, sample batches from replay buffer

↓

How?

Sampled from replay buffer

$$\min_\phi \mathbb{E}_{\substack{(s_t,a_t,s_{t+1})\sim\mathcal{D}\\ a_{t+1}\sim\pi(\cdot|s_{t+1})}}\left[\left(Q^\pi_\phi(s_t,a_t)-(r(s_t,a_t)+Q^\pi_{\hat{\phi}}(s_{t+1},a_{t+1}))\right)^2\right]$$

$$\max_\pi \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi(.|s)}\left[Q^\pi(s,a)\right]$$

1. Sample uniformly
2. Prioritize by TD-error
3. Prioritize by target error
4. … open area of research!

Slowing moving targets with target networks

Continuous updates can be unstable since there is a churn of projection and backup

$$\min_{\phi} \mathbb{E}_{\substack{(s_t, a_t, s_{t+1}) \sim \mathcal{D} \\ a_{t+1} \sim \pi(\cdot | s_{t+1})}} \left[\left(Q^{\pi}_{\phi}(s_t, a_t) - \left(r(s_t, a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1}, a_{t+1}) \right) \right)^2 \right]$$

If we set $\bar{\phi}$ to ϕ every update, the update becomes very unstable

$\Downarrow$

Move $\bar{\phi}$ to ϕ slowly!

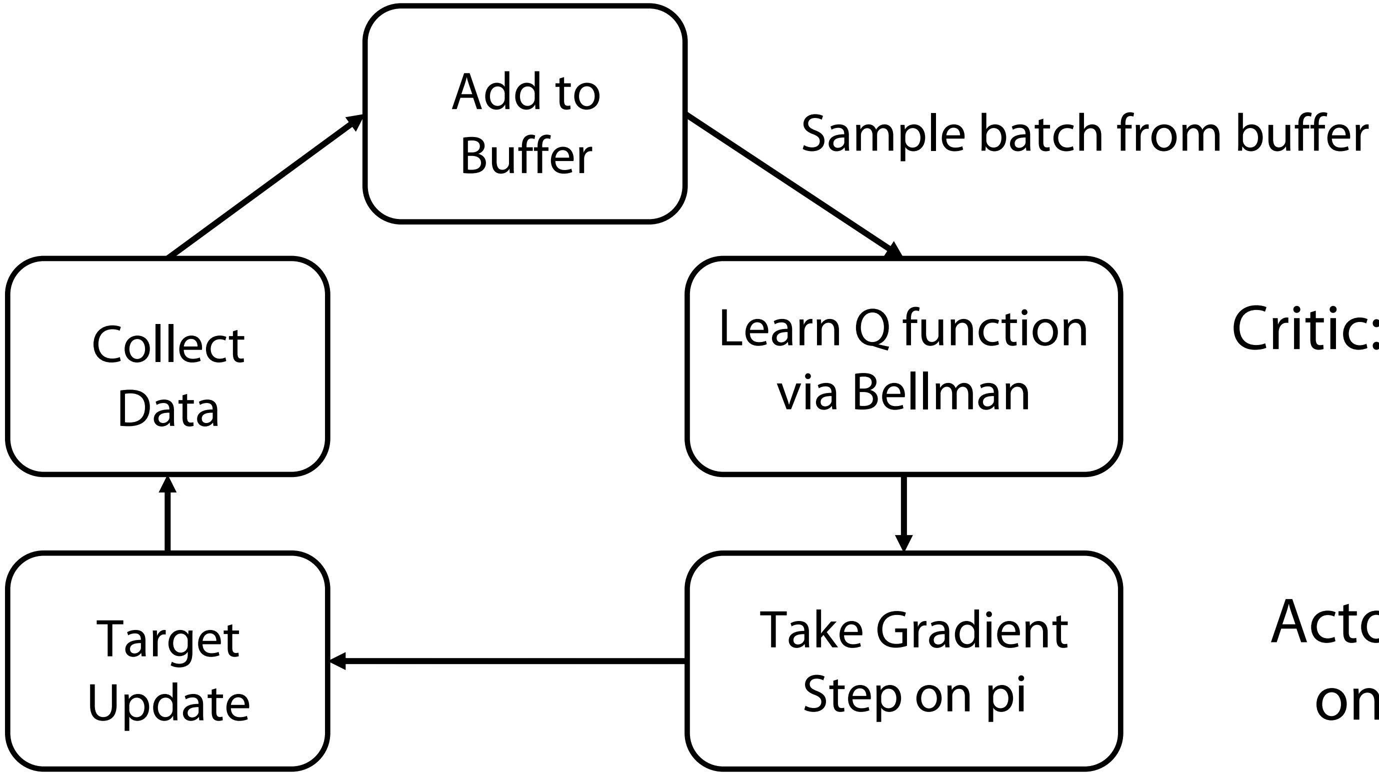
$$\bar{\phi} = (1 - \tau)\phi + \tau\bar{\phi}$$

Polyak averaging

A Practical Off-Policy RL Algorithm

1 sample

Polyak Averaging



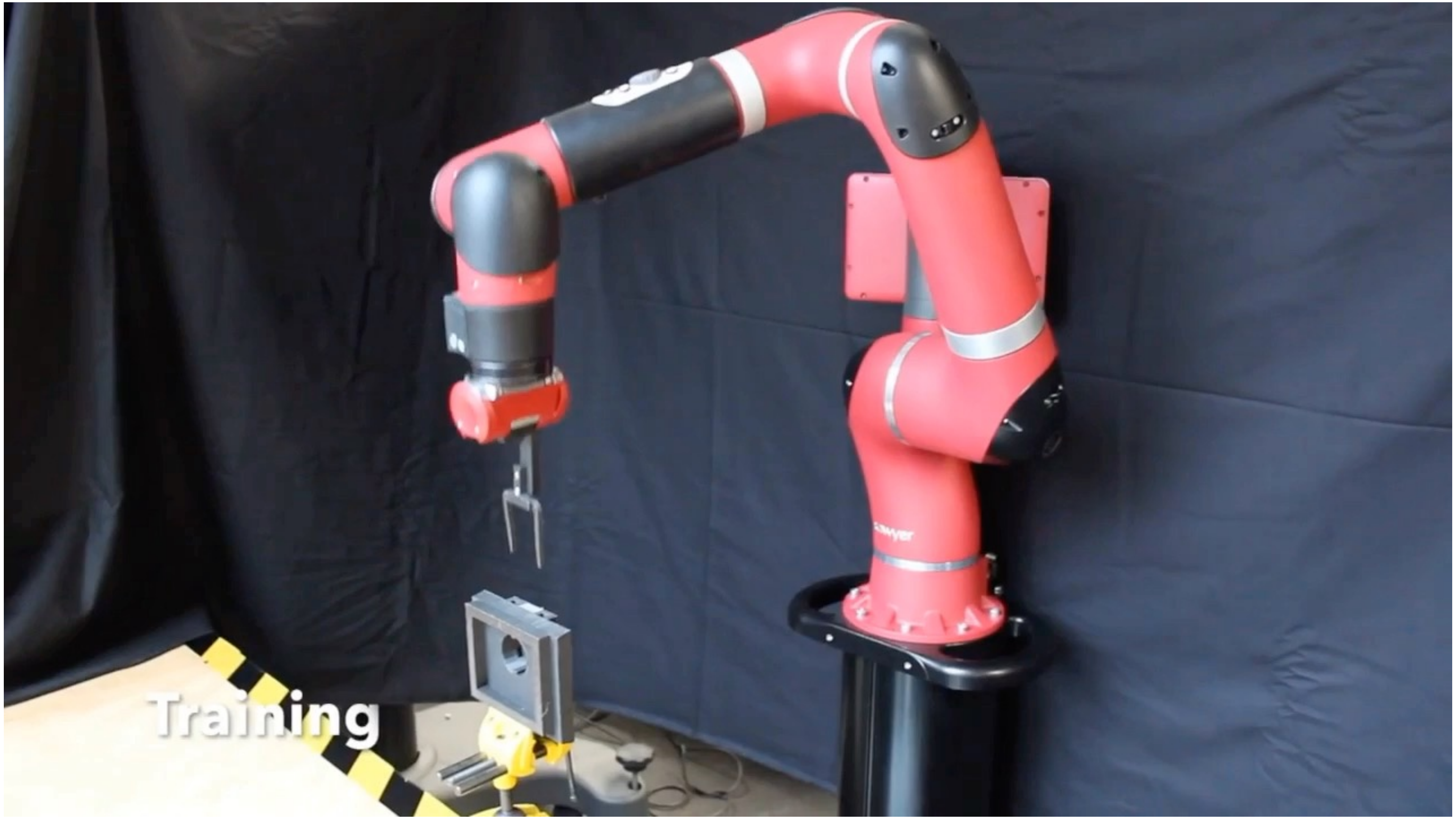
Critic: 1 gradient step on Bellman error

Actor: 1 gradient step on policy gradient

Practical Actor-Critic in Action

Trained using QT-Opt

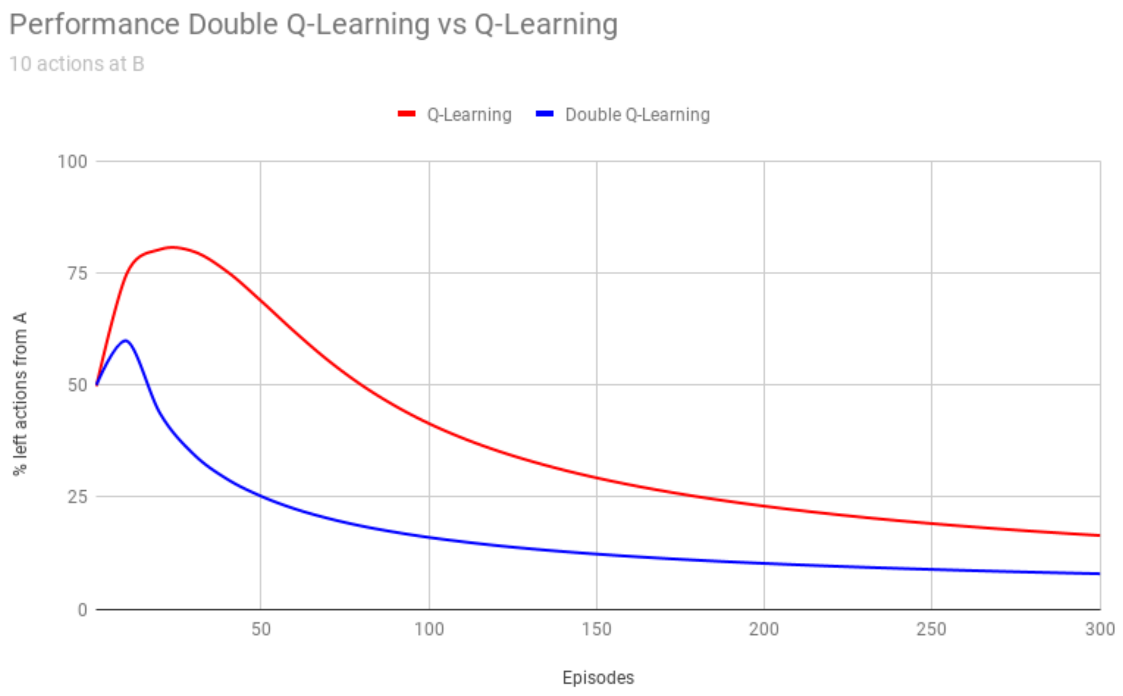
Practical Actor-Critic in Action



Trained using DDPG

What can we do to make them match on-policy algorithms in asymptotic performance?

Where does this fail?



Some issues remain:

1. **Overestimation bias**
2. Insufficient exploration

Let's try and understand these!

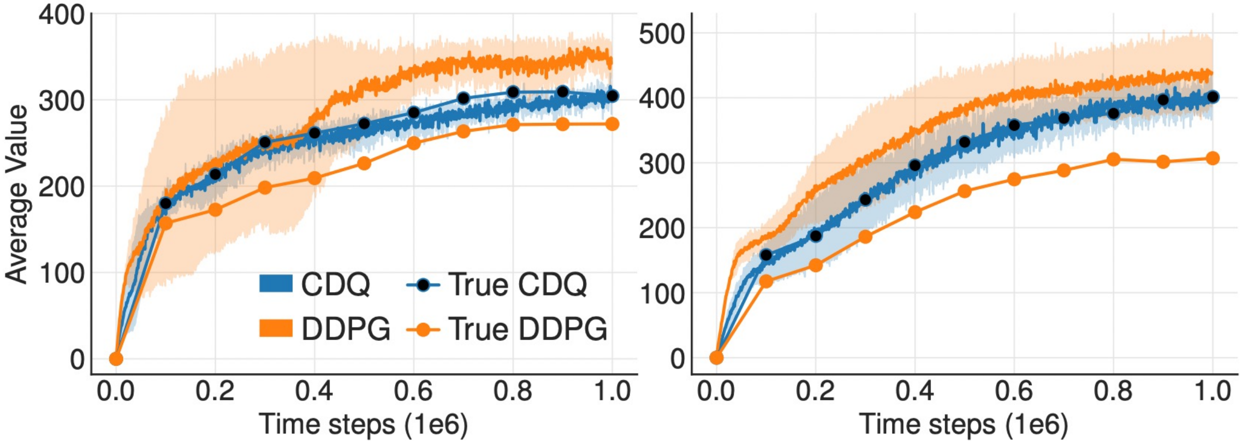
Overestimation Bias in Actor-Critic

Optimized Q's are often overly optimistic

$$\min_{\phi} \mathbb{E}_{\substack{(s_t, a_t, s_{t+1}) \sim \mathcal{D} \\ a_{t+1} \sim \pi(\cdot|s_{t+1})}} \left[\left(Q^{\pi}_{\phi}(s_t, a_t) - (r(s_t, a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1}, a_{t+1})) \right)^2 \right]$$

Q is meant to be an expectation

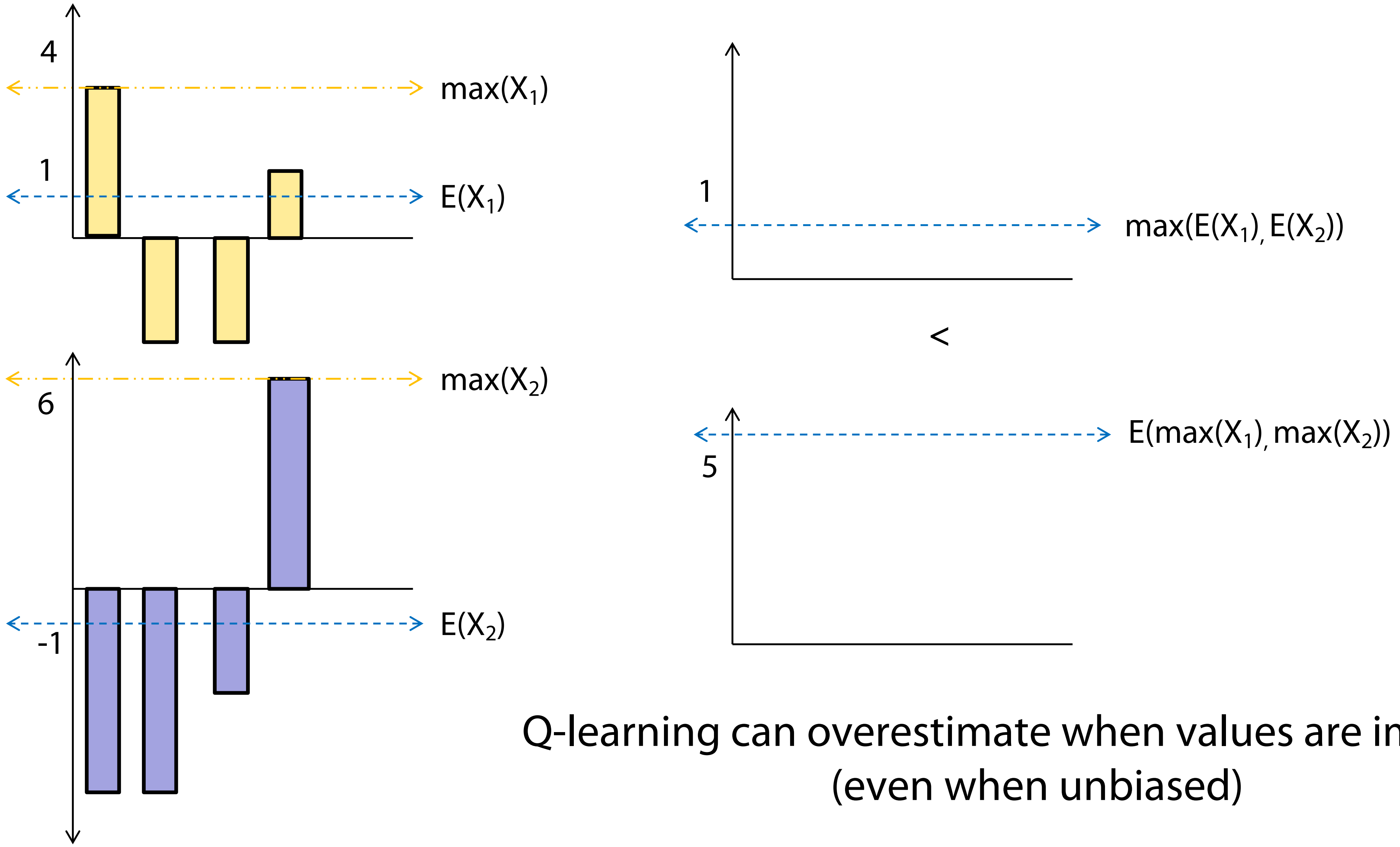
→ actually a random variable because of limited data/stochasticity



(a) Hopper-v1 (b) Walker2d-v1

E(max) > max(E), so values are optimistic

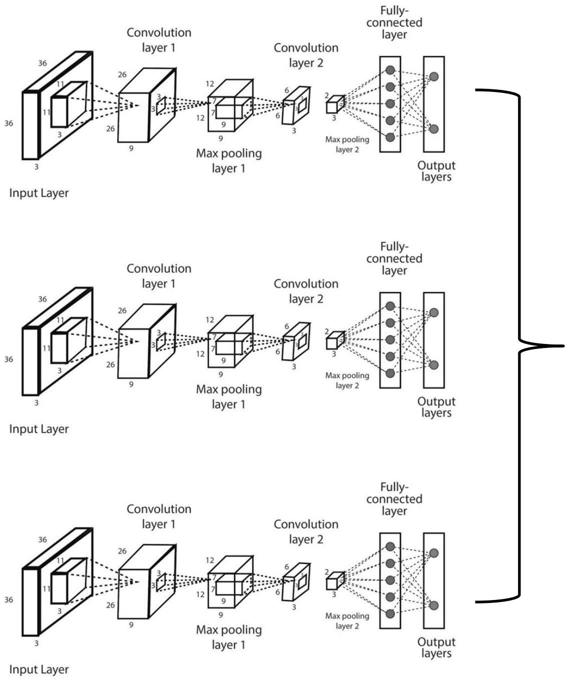
Overestimation Bias in Actor-Critic



Q-learning can overestimate when values are imperfect (even when unbiased)

Overestimation Bias in Actor-Critic – Ensemble Q

Learn two (or N) independent measures of Q, take the minimum
→ pessimistic on random variable



Independent updates

Critic

$$y_j = r(s,a) + \gamma \min_{i=1,\dots,N} Q_{\bar{\phi}_i}(s', \pi_\theta(s'))$$

$$\min_{\phi_j} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[(Q_{\phi_j}(s,a) - y_j)^2 \right]$$

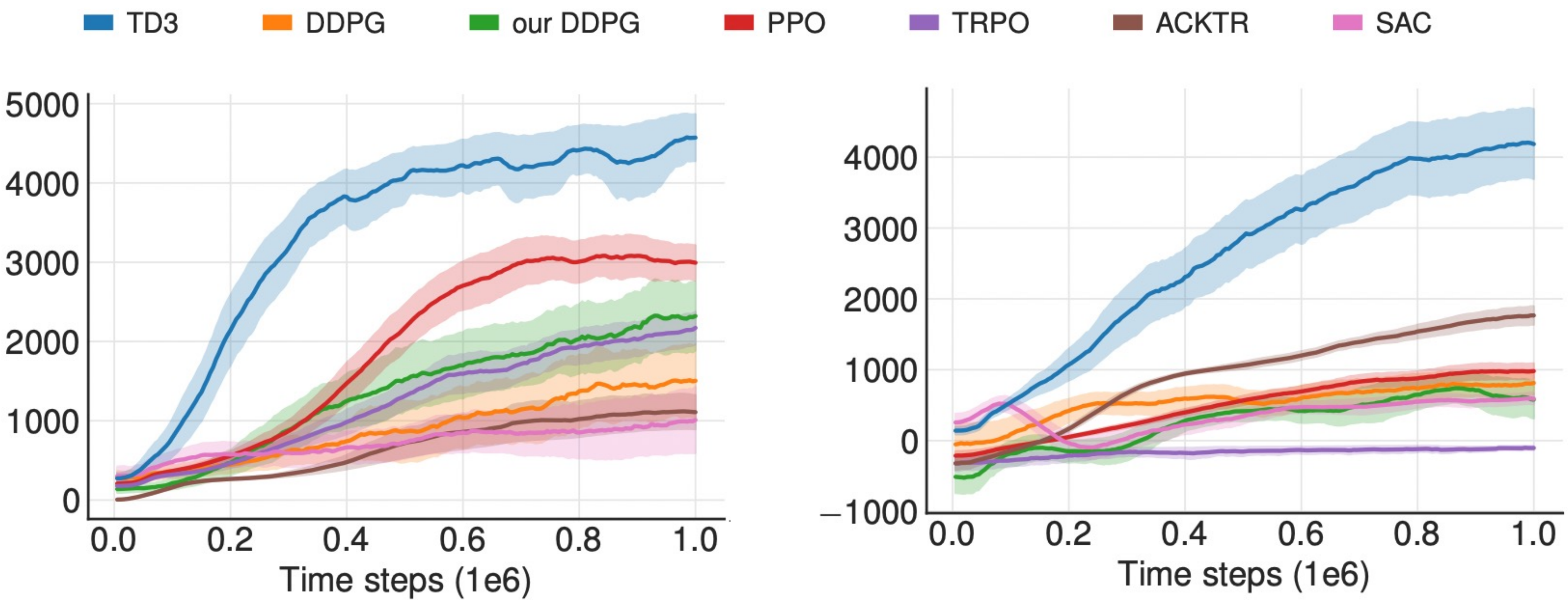
Actor

$$\max_{\theta_j} \mathbb{E}_{s\sim\mathcal{D}} \mathbb{E}_{a\sim\pi_{\theta_j}} \left[Q_{\phi_j}(s,a) \right]$$

Significantly improves overestimation and in turn sample efficiency!

Overestimation Bias in Actor-Critic

Significantly improves overestimation and in turn sample efficiency!



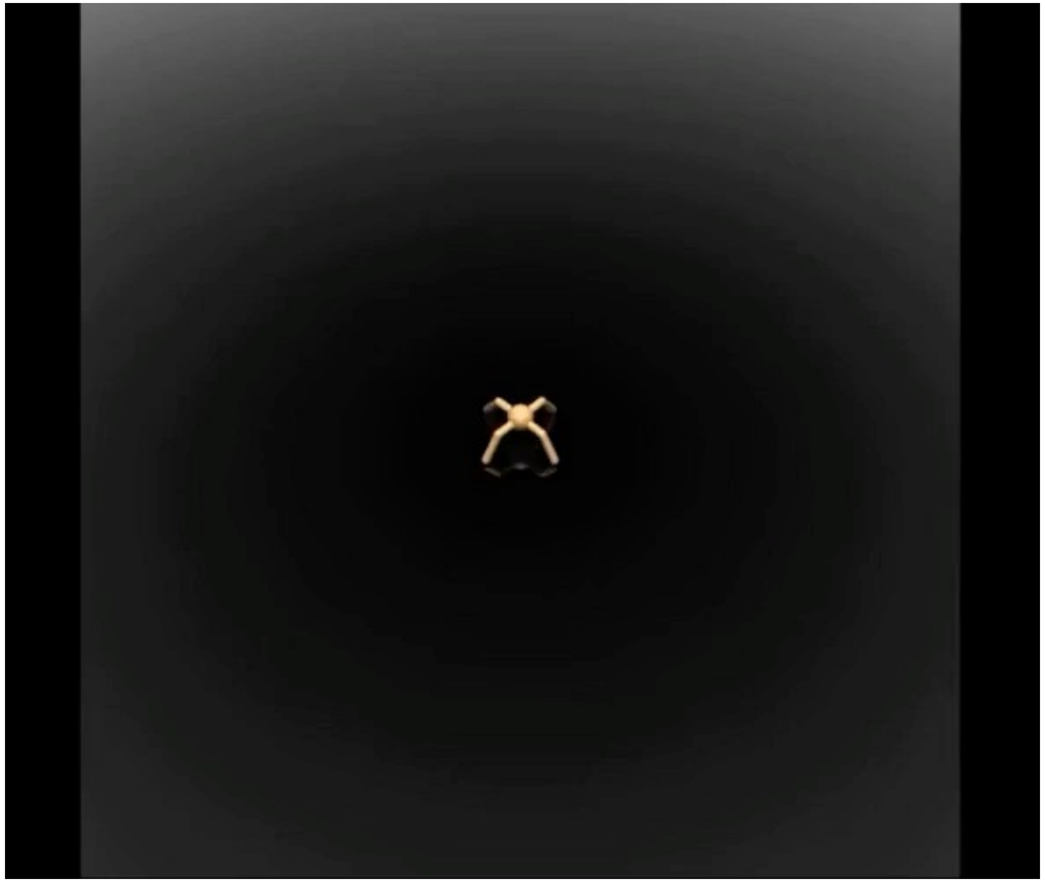
(c) Walker2d-v1

(d) Ant-v1

Double Actor Critic in Action

Double Actor Critic in Action

Where does this fail?



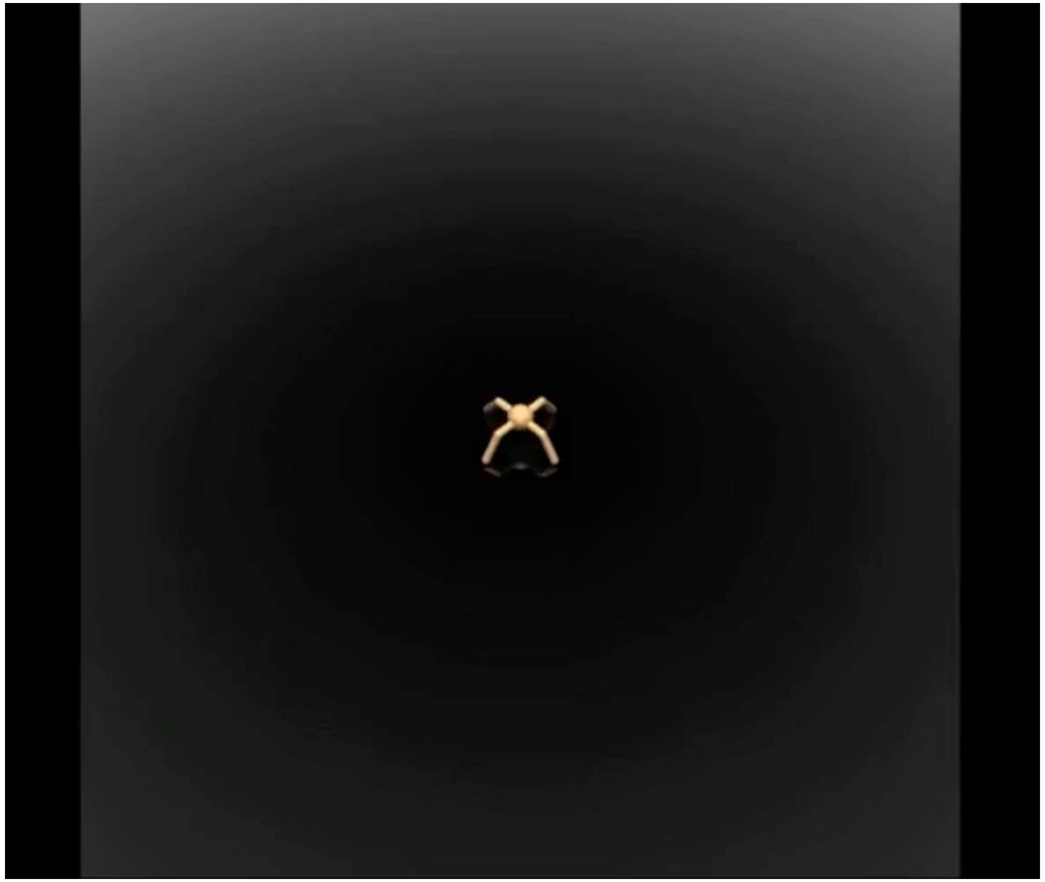
Some issues remain:

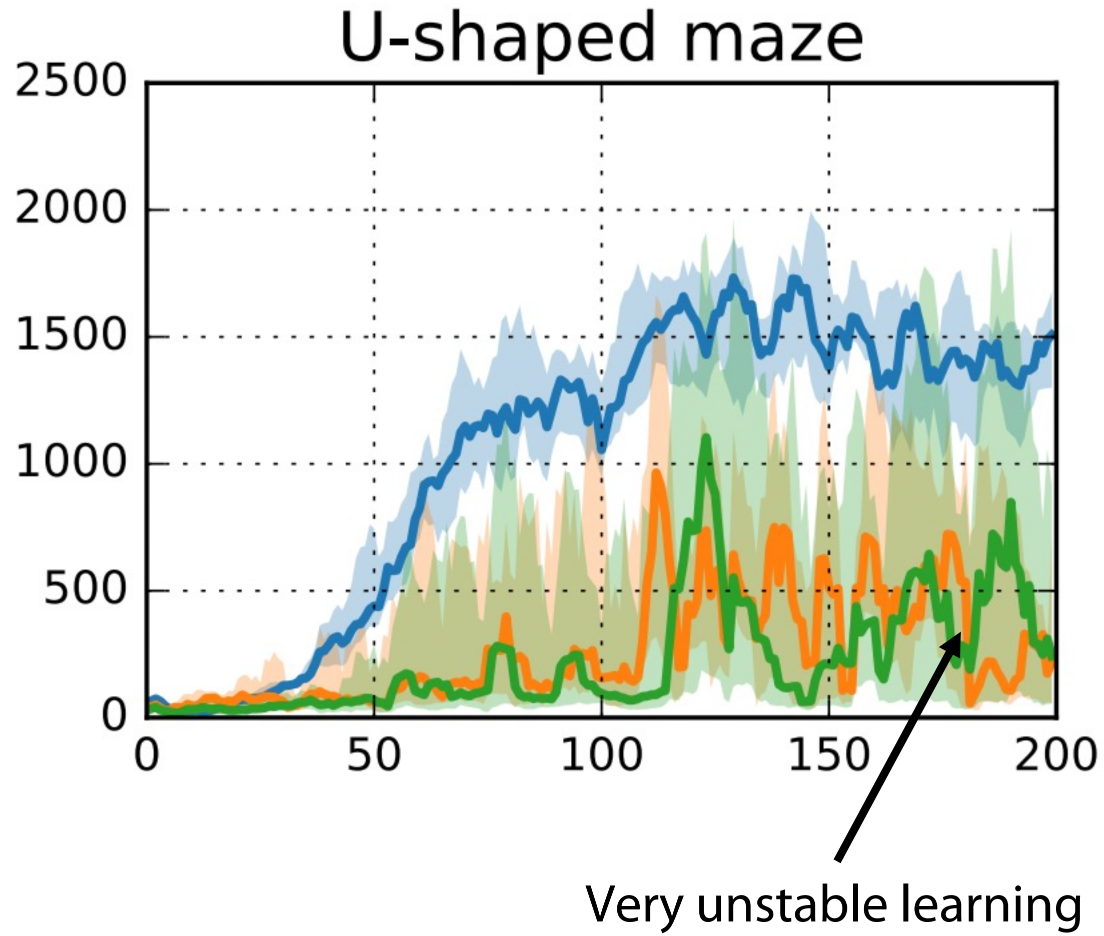
1. Overestimation bias
2. **Insufficient exploration**

Let’s try and understand these!

Collapse of Exploration in Off-Policy RL

Deep RL policies will often converge prematurely or explore insufficiently





Addressing Policy Collapse in Off-Policy RL

Adding entropy to the RL objective can help significantly

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$$

$$\Downarrow$$

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(.|s_t))\right]$$

Simple change in on-policy RL

$$\mathbb{E}_{\pi}\left[\sum_{t=0}^{T}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} \gamma^{t'-t}(r(s_t, a_t) + \alpha \mathcal{H}(\pi(.|s_t))\right] + \alpha \nabla_\theta \mathcal{H}(\pi_\theta(.|s_t))\right]$$ (via chain rule)

Max-Ent Off-Policy RL

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(.|s_t))\right]$$

Work through the recursion, same as with the regular Bellman

Critic – Policy Evaluation

$$\min_{\phi} \mathbb{E}_{\substack{(s_t, a_t, s_{t+1}) \sim \mathcal{D} \\ a_{t+1} \sim \pi(\cdot|s_{t+1})}} \left[(Q^{\pi}_{\phi}(s_t, a_t) - (r(s_t, a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1}, a_{t+1})) - \alpha \log \pi(a_{t+1}|s_{t+1}))^2 \right]$$

Actor – Policy Improvement

$$\max_{\pi} \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi(\cdot|s)} \left[Q^{\pi}_{\phi}(s, a) - \alpha \log \pi(a|s) \right] \right]$$

Soft Bellman Equation from Max-Ent RL

Optimize a "soft" Bellman equation

$$Q(s_t, a_t) \leftarrow r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s}\left[V(s_{t+1})\right]$$

$$V(s_t) \leftarrow \max_a Q(s_t, a)$$

$$\pi(a|s_t) \leftarrow \arg\max_a Q(s_t, a)$$

$$Q_{\text{soft}}(s_t, a_t) \leftarrow r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s}\left[V_{\text{soft}}(s_{t+1})\right]$$

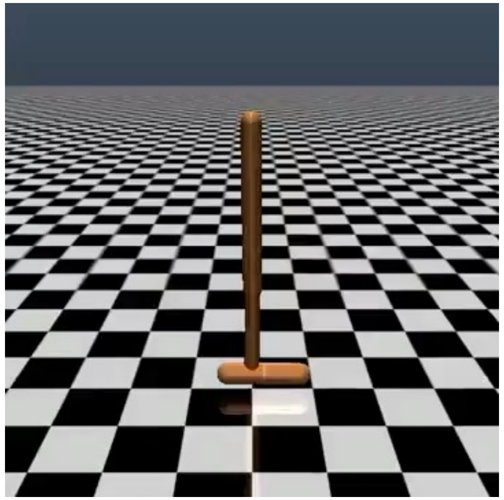
$$V_{\text{soft}}(s_t) \leftarrow \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{\text{soft}}(s_t, a')\right) da'$$

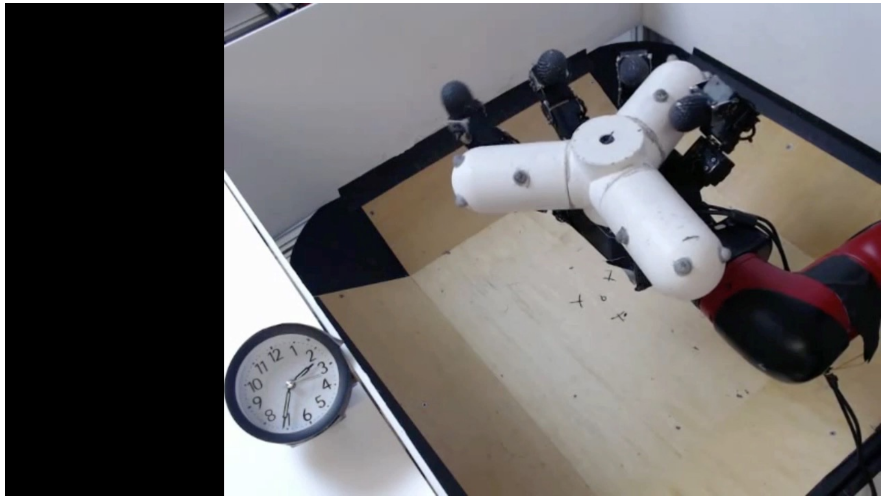
$$\pi_{\text{soft}}(a|s_t) = \exp\left(\frac{1}{\alpha}\left(Q_{\text{soft}}(s_t, a) - V_{\text{soft}}(s_t)\right)\right)$$

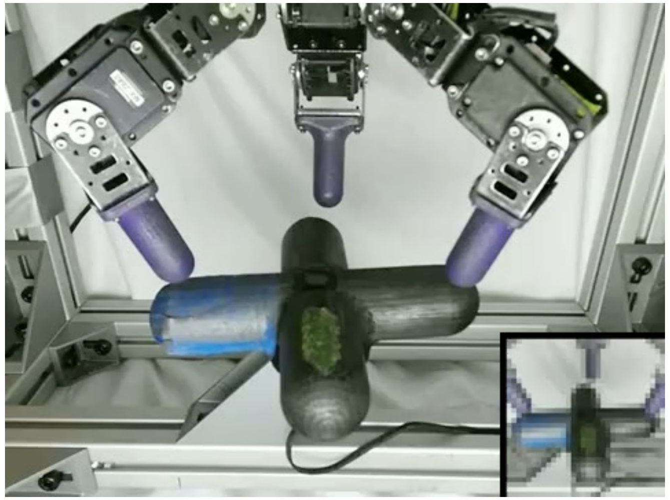
Go from max to "softmax" (imagine if α goes to 0, it becomes a max)

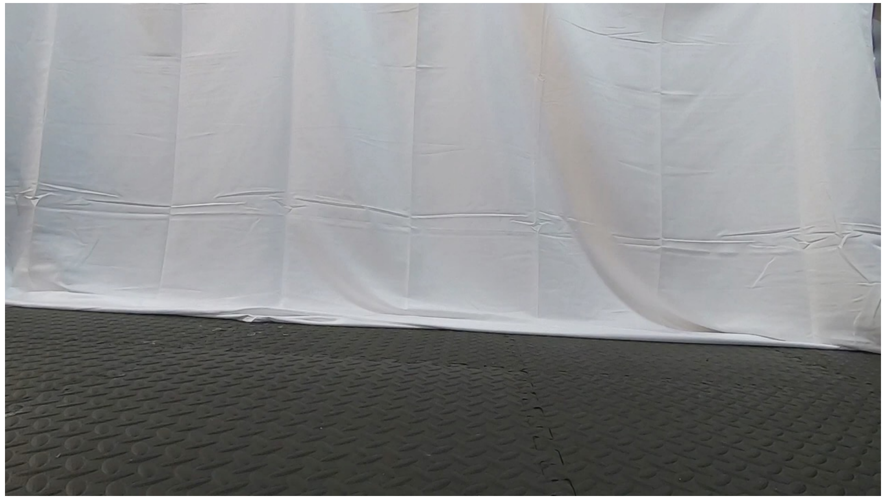
Prevents premature collapse of exploration while smoothing out optimization landscape!

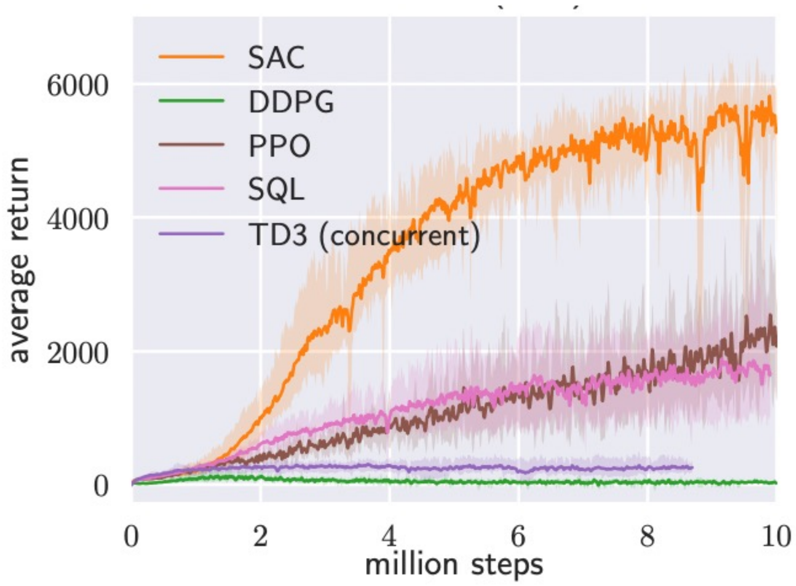
Maximum Entropy Actor-Critic Algorithms in Action











(f) Humanoid (rllab)

Lecture outline

Kronecker Factorization (K-FAC)

↓

Frontiers of Policy Gradients

↓

Going from Monte Carlo Returns to Critic Estimation

↓

Getting Actor Critic to Work in Practice

Ok, so are off-policy algorithms perfect?

What makes off-policy RL hard?

Deadly triad:

1. Function Approximation
2. Bootstrapping
3. Off-policy learning

$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q^{\pi}_{\phi}(s_t, a_t) - \left(r(s_t, a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1})\right]\right) \right]^2 \right]$$

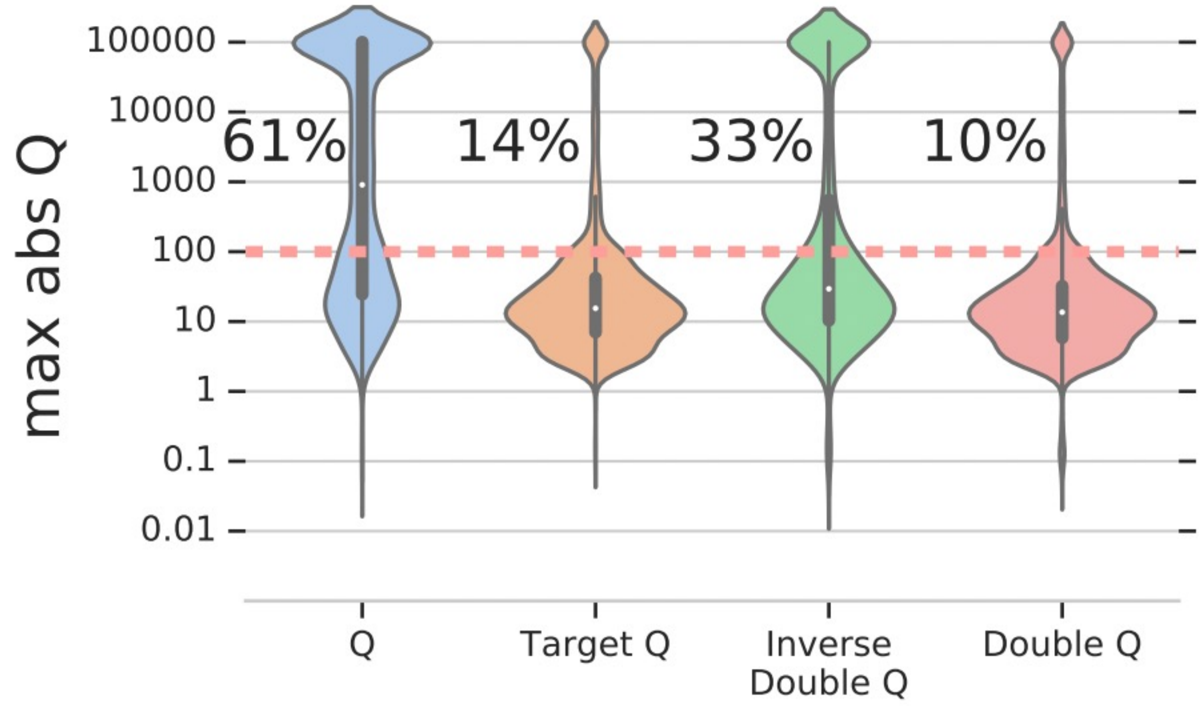
These in combination lead to many of the difficulties in stabilizing off-policy RL with function approximation

Zooming out – what makes off-policy RL hard?

Deadly triad:

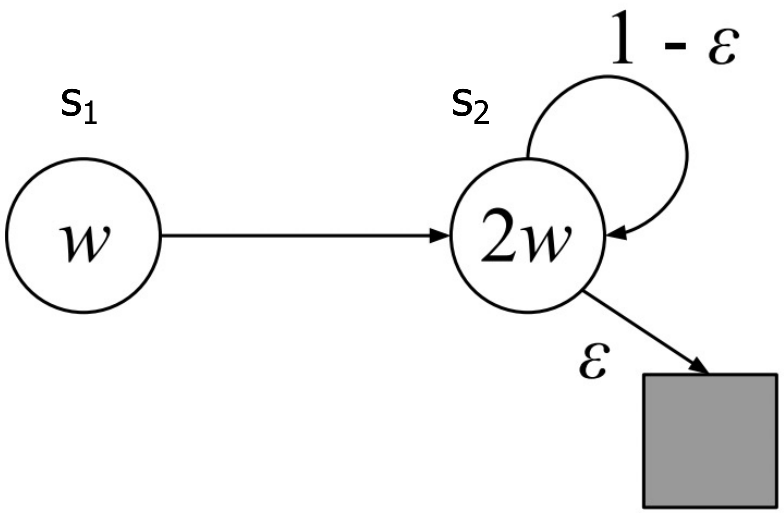
1. Function Approximation
2. Bootstrapping
3. Off-policy learning

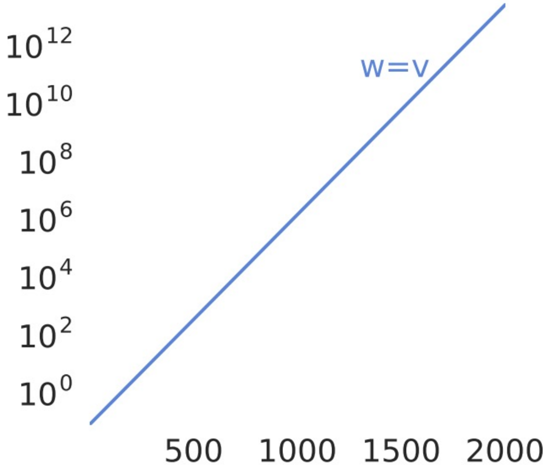
61% of runs show divergence of Q-values



Diverges even with linear function approximation, when off-policy + bootstrapping

Zooming out – what makes off-policy RL hard?





(b) $v(s) = w\phi(s)$ diverges.

Let's go to the whiteboard!

What should I work on?

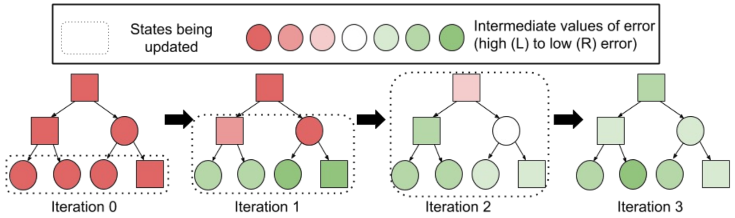
Where does the frontier of off-policy RL lie?

Off-policy is an extremely promising tool, but not quite plug and play like PG methods

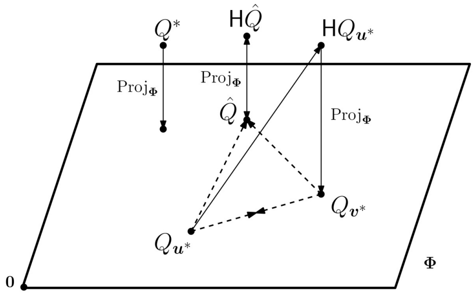
- Low variance, off-policy, avoids reconstruction, performs dynamic programming
- Has the potential to be **performant** and **sample efficient**

But in practice is often unstable, inefficient with high dimensional observations

Sampling



Theory



Exploration

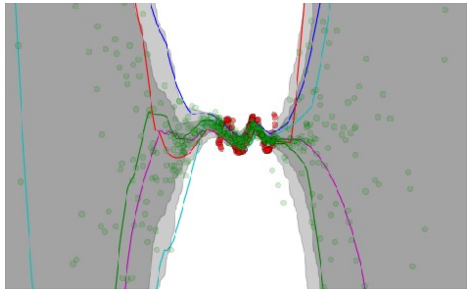
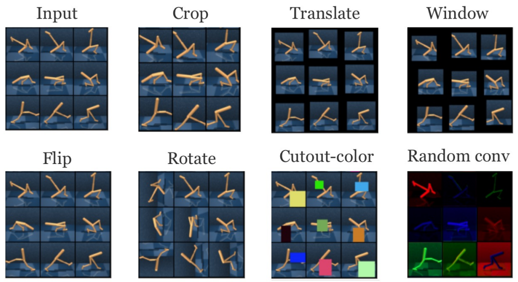
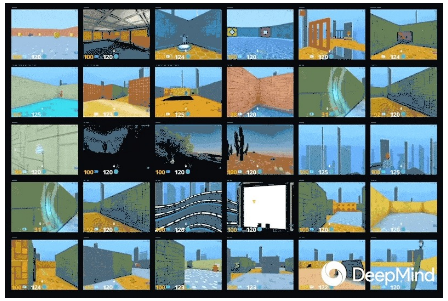


Image-based RL

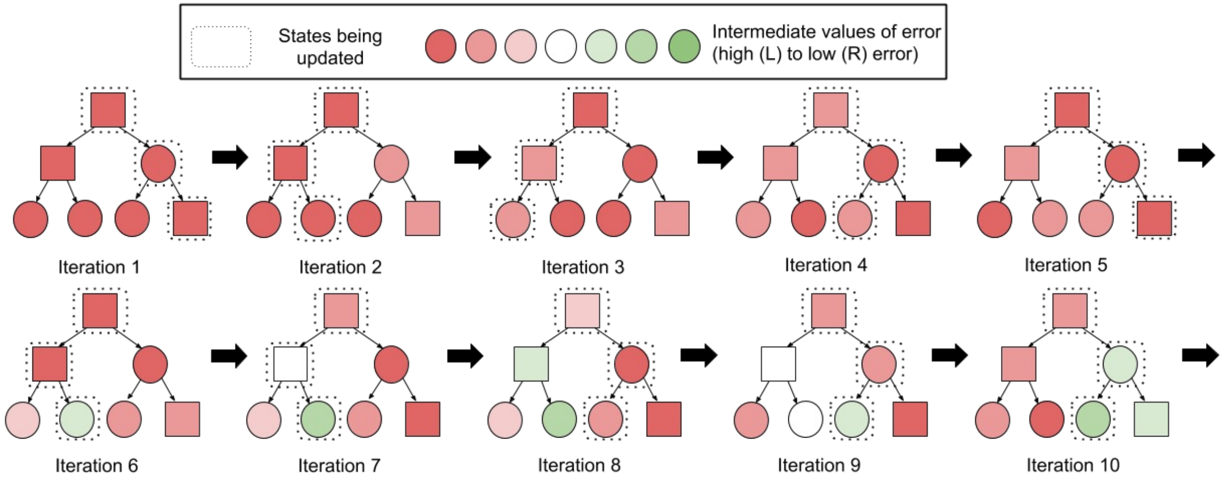


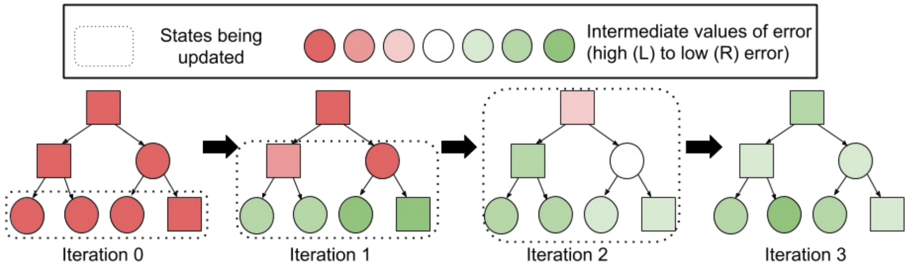
Partial Observability

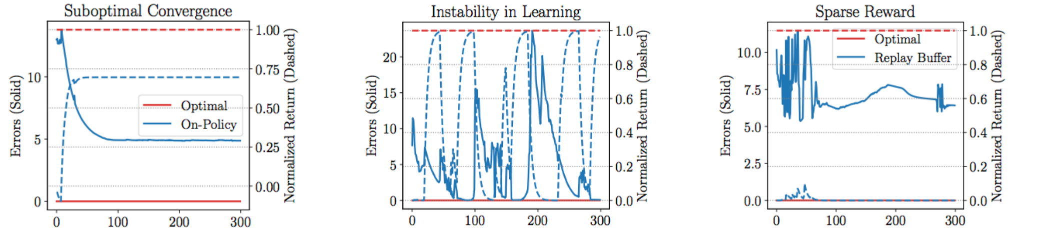


Prioritizing Experience

Performing uniform buffer TD updates can be catastrophically bad







(a) Sub-optimal convergence for on-policy distributions: return (dashed) and value error (solid). Note that value error decreases rapidly at the start and finally converges to a nonzero value, leading to sub-optimal return.

(b) Instability for replay buffer distributions: return (dashed) and value error (solid) over training iterations. Note the rapid increase in value error at multiple points, which co-occurs with instabilities in returns.

(c) Error (left) and returns (right) for sparse reward MDP with replay buffer distributions. Note the inability to learn, low return, and highly unstable value error $\mathcal{E}_k$, often increasing sharply, destabilizing the learning process.

Need to prioritize updates to propagate good values

Theory/Convergence with Function Approximation

Significant body of work on learning dynamics with function approximation

Delusional Bias

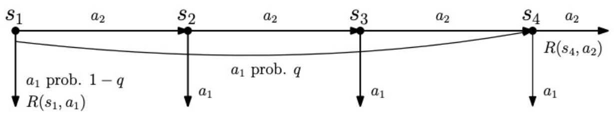


Figure 1: A simple MDP that illustrates delusional bias (see text for details).

Implicit regularization

$$\overline{\mathcal{R}}_{\exp}(\theta) = \sum_{i \in \mathcal{D}} \phi(\mathbf{s}_i, \mathbf{a}_i)^\top \phi(\mathbf{s}'_i, \mathbf{a}'_i).$$

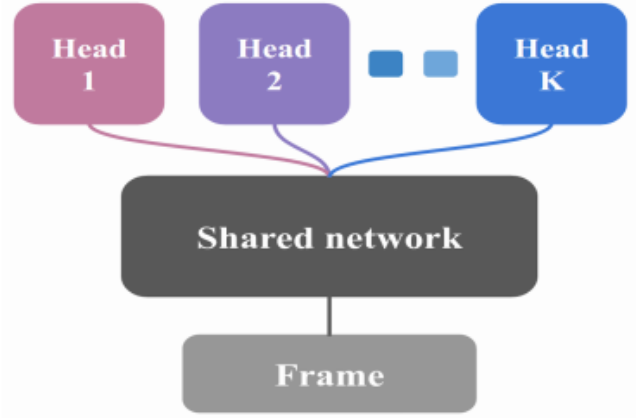
Bilinear classes

Framework	B-Rank	B-Complete	W-Rank	Bilinear Class (this work)
B-Rank	✓	✗	✓	✓
B-Complete	✗	✓	✗	✓
W-Rank	✗	✗	✓	✓
Bilinear Class (this work)	✗	✗	✗	✓

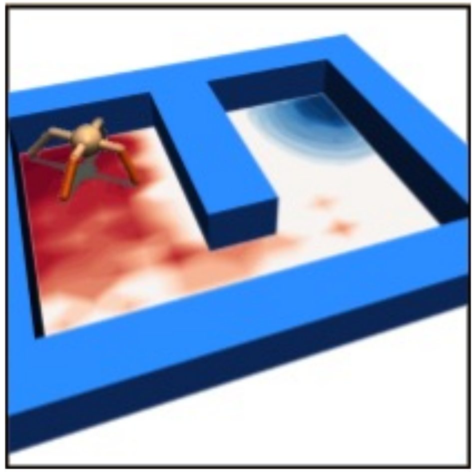
Exploration in Off-Policy RL

Better exploration methods

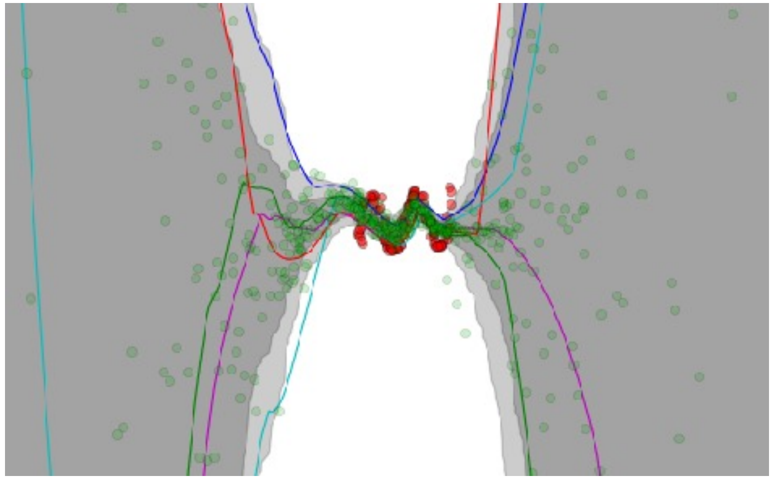
Uncertainty based methods



Count-based methods



Information gain methods

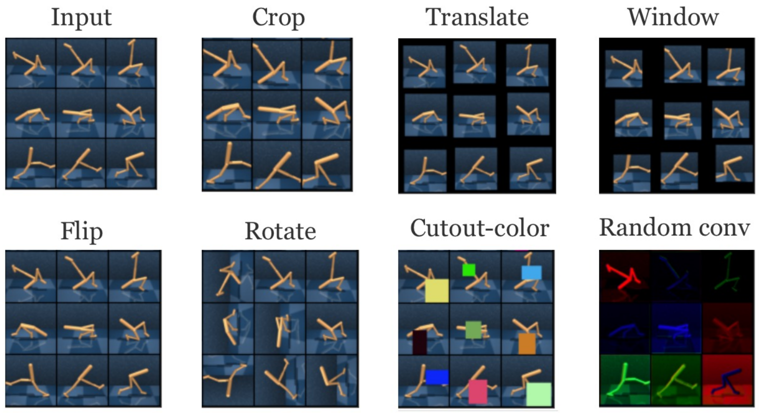


Often critical for getting algorithms to work!

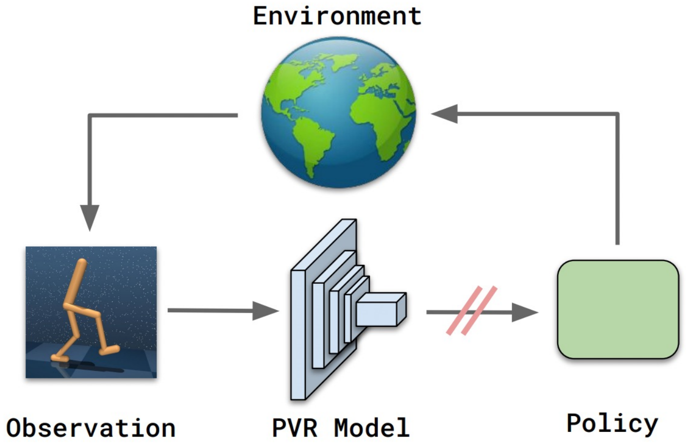
Image-based Off-Policy RL

Learning from high dimensional observations is unstable – images/point clouds

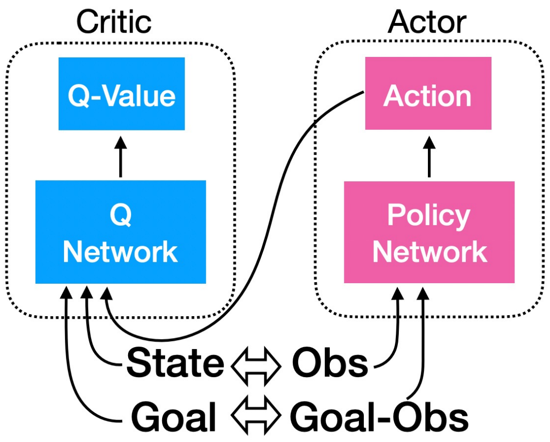
Data augmentations



Pre-trained representations



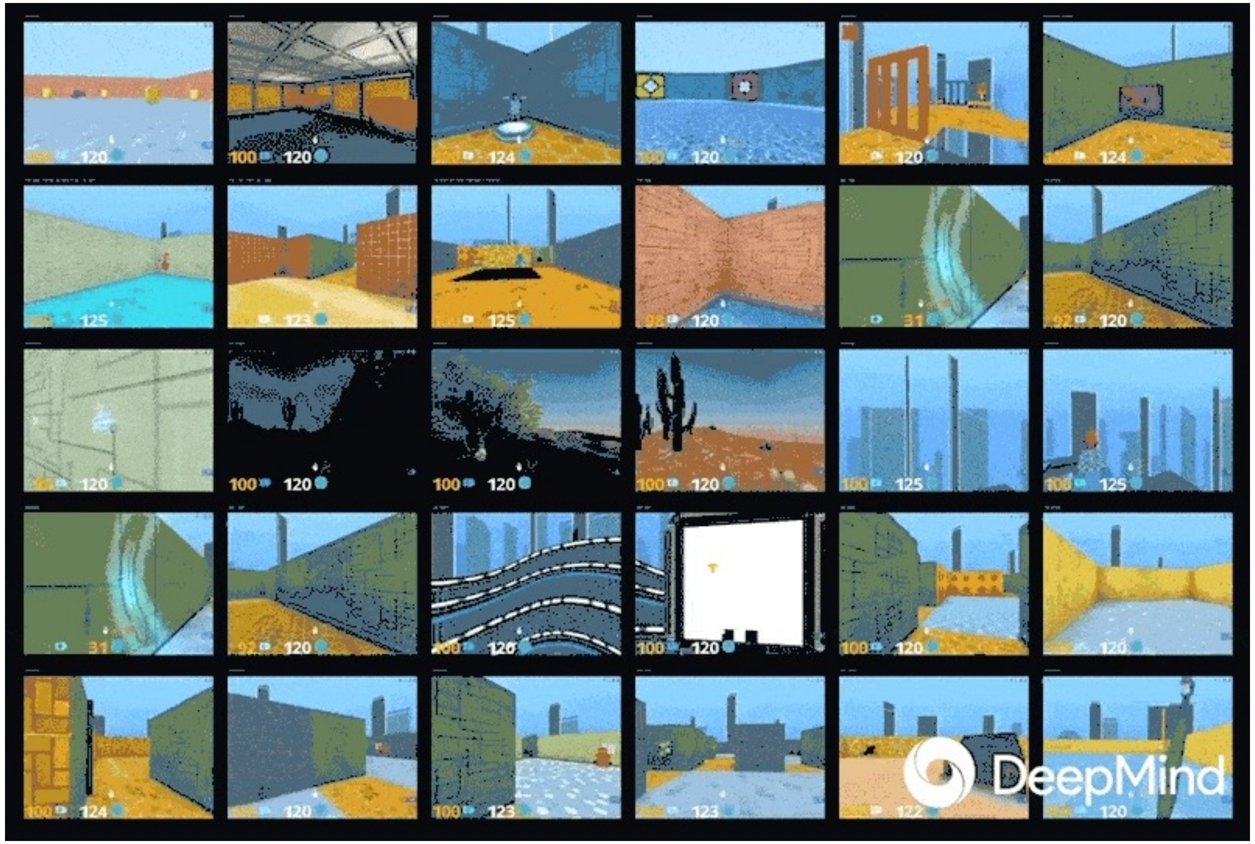
Student-teacher

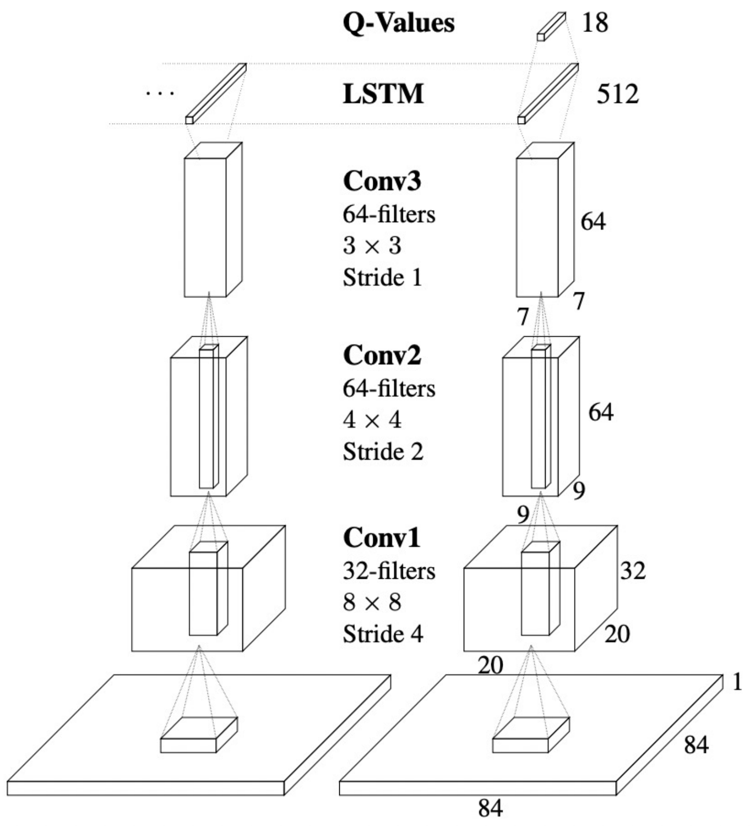


Still very unstable, lot of open research problems!

Partial Observability in Off-Policy RL

Off-policy methods critically depend on the Markov assumption



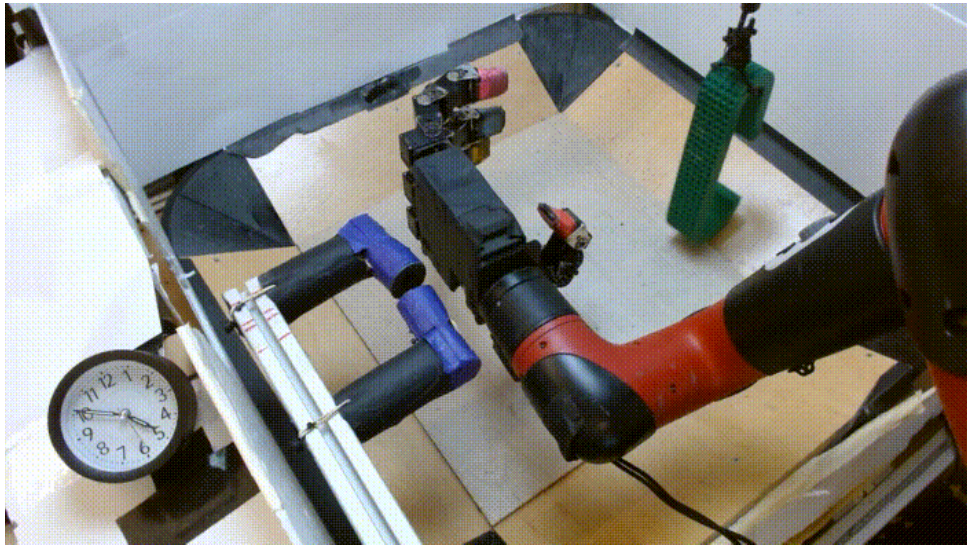


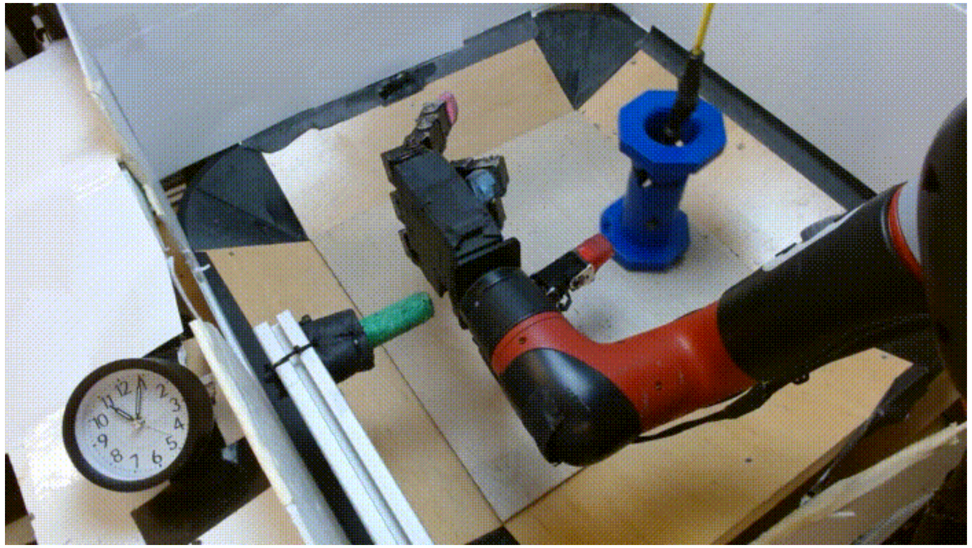
Learning history conditioned/recurrent Q-functions is an open area!

How has off-policy RL manifested in robotics?

Small changes – larger number of ensembles, more minibatch steps allow for training in < 20 mins

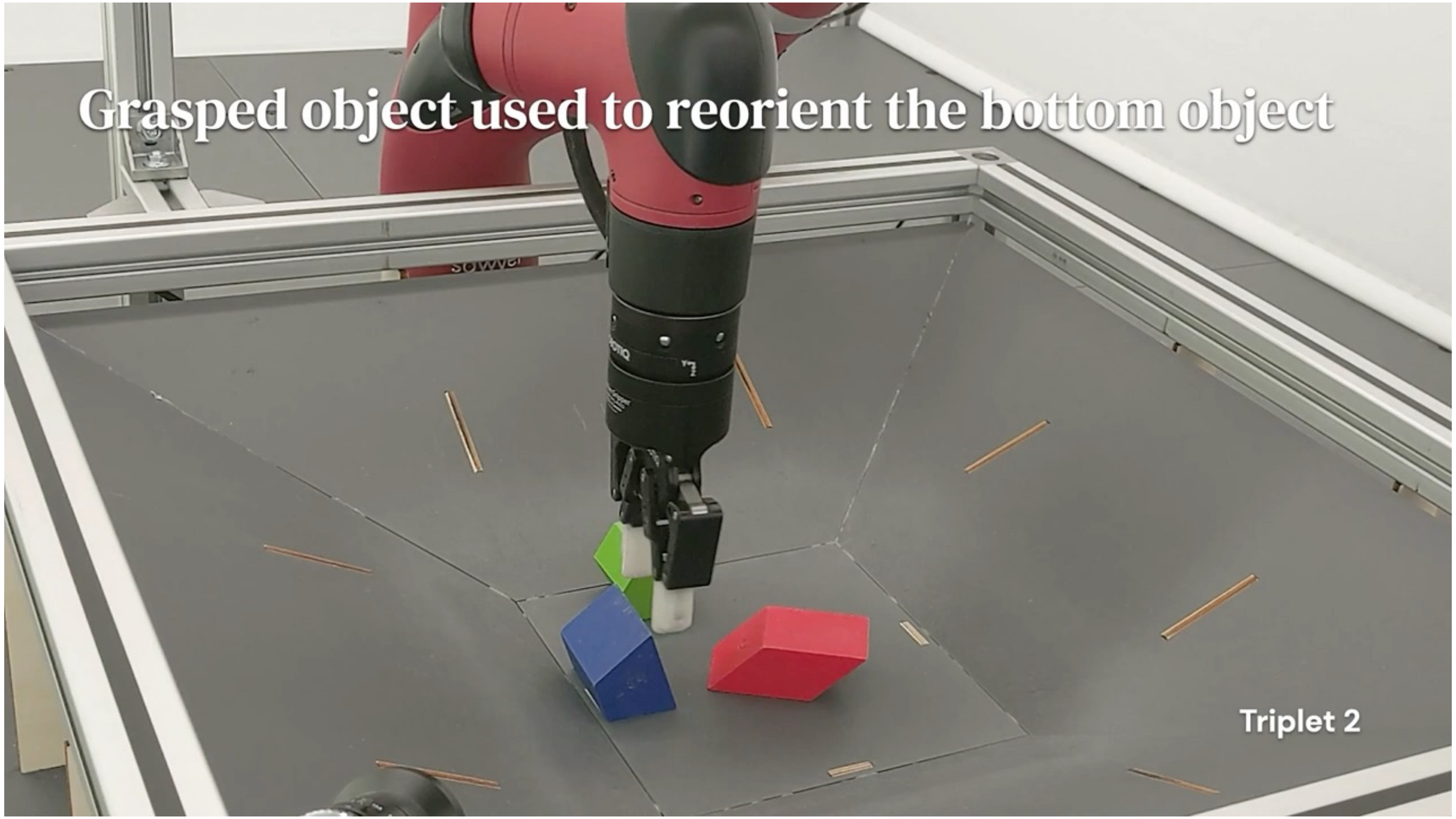
How has off-policy RL manifested in robotics?





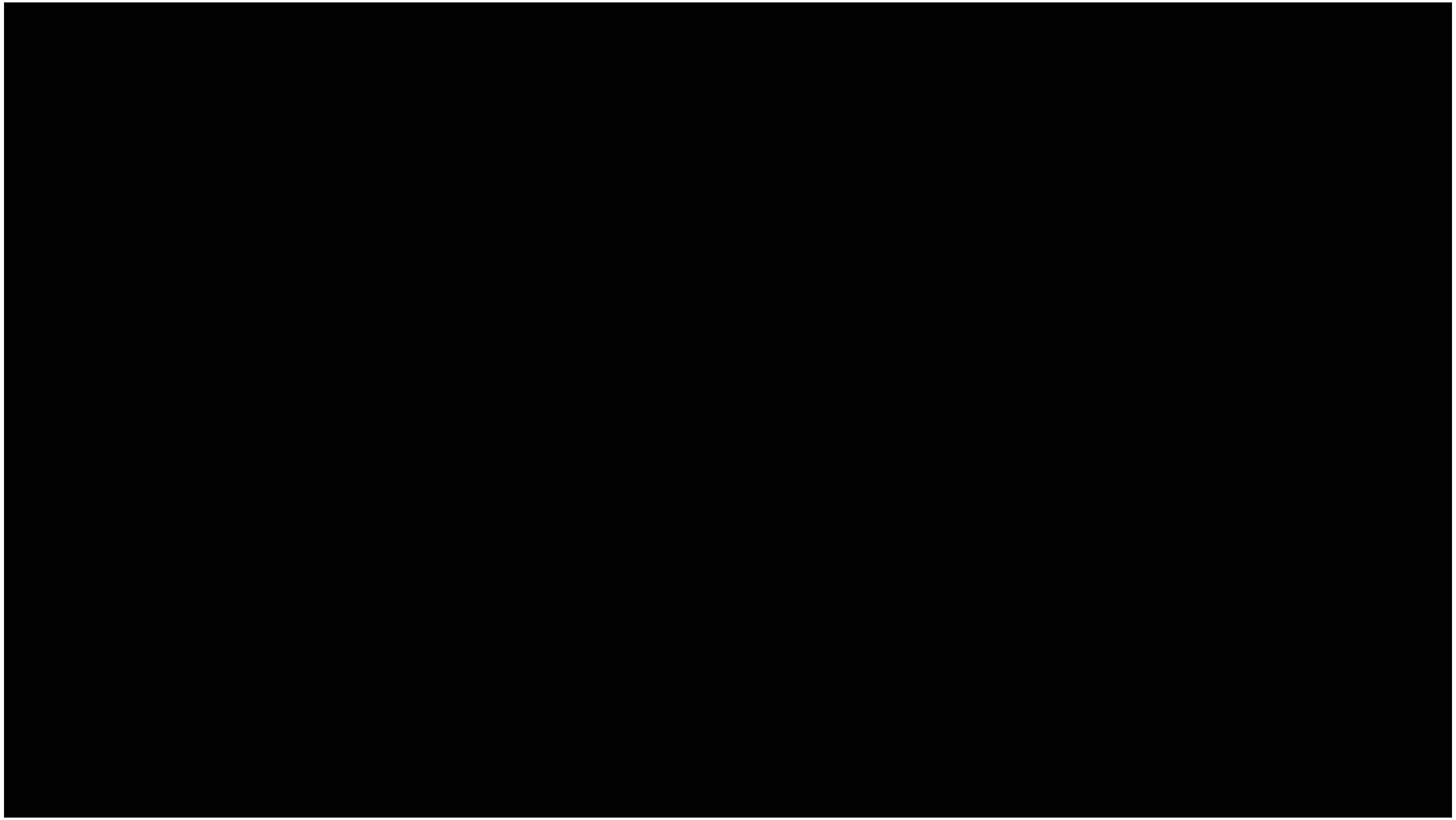
How has off-policy RL manifested in robotics?

Uses MPO – a variant of actor critic with a supervised learning style actor update

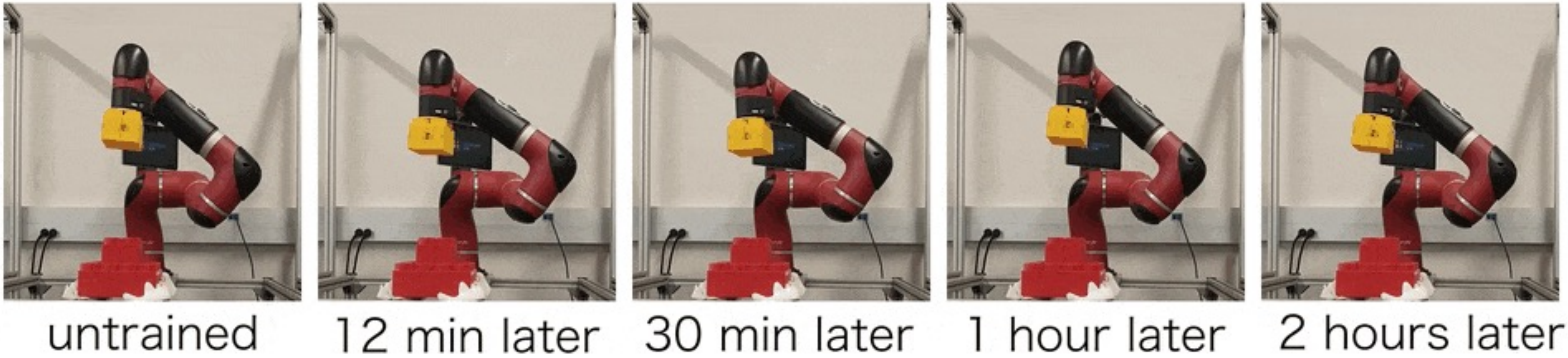


How has off-policy RL manifested in robotics?

Bootstrapped with a few demonstrations



How has off-policy RL manifested in robotics?



Pros/Cons of Off-Policy Methods in Robotics

Pros:

1. Sample-efficient enough for real world
2. Can learn from images with suitable design choices
3. Off-policy, can incorporate prior data

Cons

1. Often unstable
2. Can achieve lower asymptotic performance
3. Requires significant storage

Fin.