Robot Perception and Learning

Advanced Policy Gradient

Tsung-Wei Ke

Fall 2025

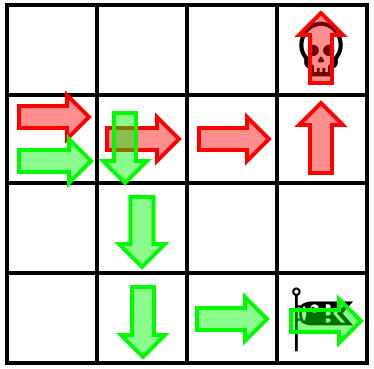
Disclaimer

- This lecture heavily borrows contents from:
  - RL-Trust Region Policy Optimization (TRPO) Explained by Jonathan Hui (link)
  - Reinforcement Learning by Ping-Chun Hsieh at National Yang Ming Chiao Tung University.

Recap: Policy Gradient Also Has Problems…

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



- The gradient estimator is unbiased, but requires a very large number of samples to accurately approximate the true gradient
- When the number of sample is small, the variance is high…

Recap: A More General Solution to Reduce Variance

- Subtract a baseline b

$$\nabla J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla \log p_\theta(\tau)\,[r(\tau) - b]$$

- The gradients don't change

$$E[\nabla \log p_\theta(\tau) b] = \int p_\theta(\tau) \nabla \log p_\theta(\tau)\, b d\tau = \int \nabla p_\theta(\tau) b d\tau = b \nabla \int p_\theta(\tau) b d\tau = 0$$

- Reduce the variance with carefully selected baseline:

$$\mathrm{Var}[x] = E[x^2] - E[x]^2$$

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)]$$

$$\mathrm{Var} = E_{\tau \sim p_\theta(\tau)}[(\nabla_\theta \log p_\theta(\tau)(r(\tau) - b))^2] - \underbrace{E_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)]^2}$$

this bit is just $E_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) r(\tau)]$
(baselines are unbiased in expectation)

Recap: Actor Critic

- Policy Gradient:

$$\nabla J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=0}^{T} \nabla \log \pi_\theta(a_t | s_t) \right) \left(\sum_{t=0}^{T} r(s_t, a_t) - b \right) \right]$$

- Re-write with action value $Q_\pi(s, a)$:

$$\nabla J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=0}^{T} \nabla \log \pi_\theta(a_t | s_t) \right) (Q^\pi(s_t, a_t) - b) \right]$$

Recap: Actor Critic

- Re-write with action value $Q_\pi(s,a)$:

$$\nabla J(\theta) = E_{\tau\sim p_\theta(\tau)}\left[\left(\sum_{t=0}^{T}\nabla\log\pi_\theta(a_t|s_t)\right)(Q^\pi(s_t,a_t)-b)\right]$$

$$\text{Let } b = V^\pi(s) \Rightarrow \nabla J(\theta) = E_{\tau\sim p_\theta(\tau)}\left[\left(\sum_{t=0}^{T}\nabla\log\pi_\theta(a_t|s_t)\right)(\underbrace{Q^\pi(s_t,a_t)-V^\pi(s)}_{\text{Advantage } A^\pi(s_t,a_t)})\right]$$

- Advantage $A^\pi(s_t,a_t)$ measures how much better action a_t is than other actions

- If we learn a value estimator $V_\phi^\pi(s)$ parameterized by ϕ, we obtain:
  - Estimated action value $Q^\pi(s_t,a_t) = R(s_t,a_t) + \gamma V_\phi^\pi(s_{t+1})$
  - Estimated advantage $A^\pi(s_t,a_t) = R(s_t,a_t) + \gamma V_\phi^\pi(s_{t+1}) - V_\phi^\pi(s_t)$

Wait! Policy Gradient Methods Still Have Issues

- Challenge 1: policy gradient calculates the steepest ascent direction for the rewards. First-order approximation assumes flat surface, leading to horrible updates if the surface has high curvature

$$\underbrace{\nabla J(\theta)}_{\text{Steepest direction to maximize rewards}} = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=0}^{T} \nabla \log \pi_\theta(a_t | s_t) \right) \mathrm{A}^{\pi_\theta}(s_t, a_t) \right]$$

$$\theta_{k+1} = \theta_k + \alpha \nabla J(\theta)$$

Image source

Wait! Policy Gradient Methods Still Have Issues

- Challenge 1: policy gradient calculates the steepest ascent direction for the rewards. First-order approximation assumes flat surface, leading to horrible updates if the surface has high curvature

$$\underbrace{\nabla J(\theta)}_{\text{Steepest direction to maximize rewards}} = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=0}^{T} \nabla \log \pi_\theta(a_t | s_t) \right) \mathrm{A}^{\pi_\theta}(s_t, a_t) \right]$$

$$\theta_{\mathrm{k+1}} = \theta_{\mathrm{k}} + \alpha \nabla J(\theta)$$

- Challenge 2: tuning learning rate is tricky. Low learning rate makes training slow; high learning rate breaks the policy.

Slide adapted from J. Hui

Idea: Trust Region Policy Optimization (TRPO)

Idea: TRPO iteratively updates the policy by solving the following:

In each iteration k, the policy is updated from π_{θ_k} to $\pi_{\theta_{k+1}}$

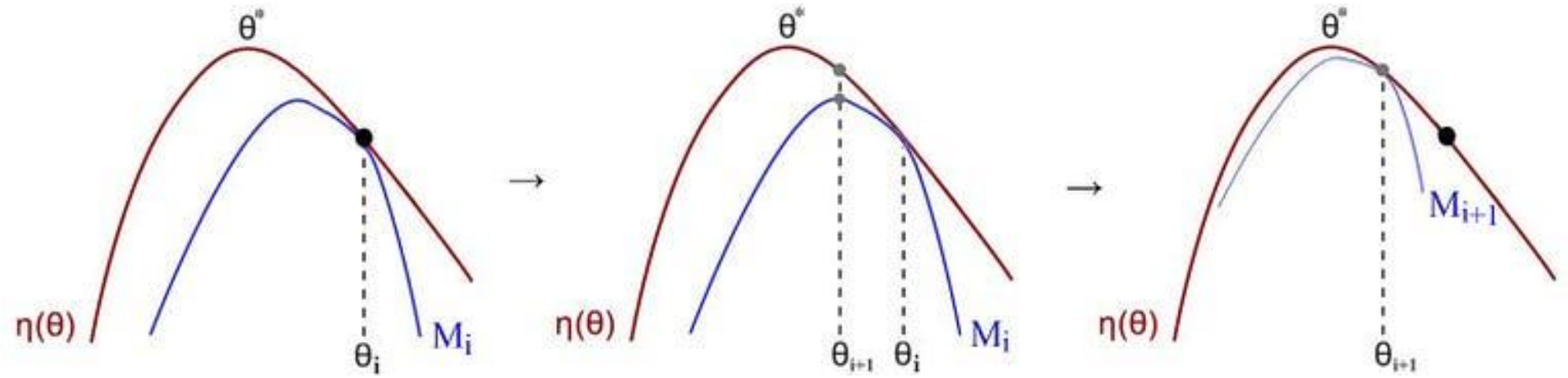
$$\pi_{\theta_{k+1}} = \arg\max_{\theta} \mathbb{E}_{s\sim d_{\mu}^{\pi_{\theta_k}}, a\sim\pi_{\theta_k}(\cdot|s)}\left[\frac{\pi_{\theta}(a\,|\,s)}{\pi_{\theta_k}(a\,|\,s)} A^{\pi_{\theta_k}}(s,a;)\right]$$

subject to $D(\pi_{\theta_k}, \pi_{\theta}) \leq \delta$ ← "trust region"

TRPO wants to achieve policy improvement via constrained optimization

Minorize-Maximization (MM) algorithm

- Can we guarantee that any policy updates always improve the expected rewards?
- The idea of MM algorithm: it may be difficult to directly improve the expected rewards η, but we can **iteratively** maximizing a lower bound function $\mathbf{M_i}$ approximating the expected reward locally



Trust Region

- Line search: determine the ascending / descending direction and take a step towards that direction (e.g. gradient descent)
- Trust region: determine the maximum step size that we want to explore and locate the optimal point within this trust region. We can adapt the step size (area of trust region) based on the goodness of approximative function $\mathbf{M_i}$

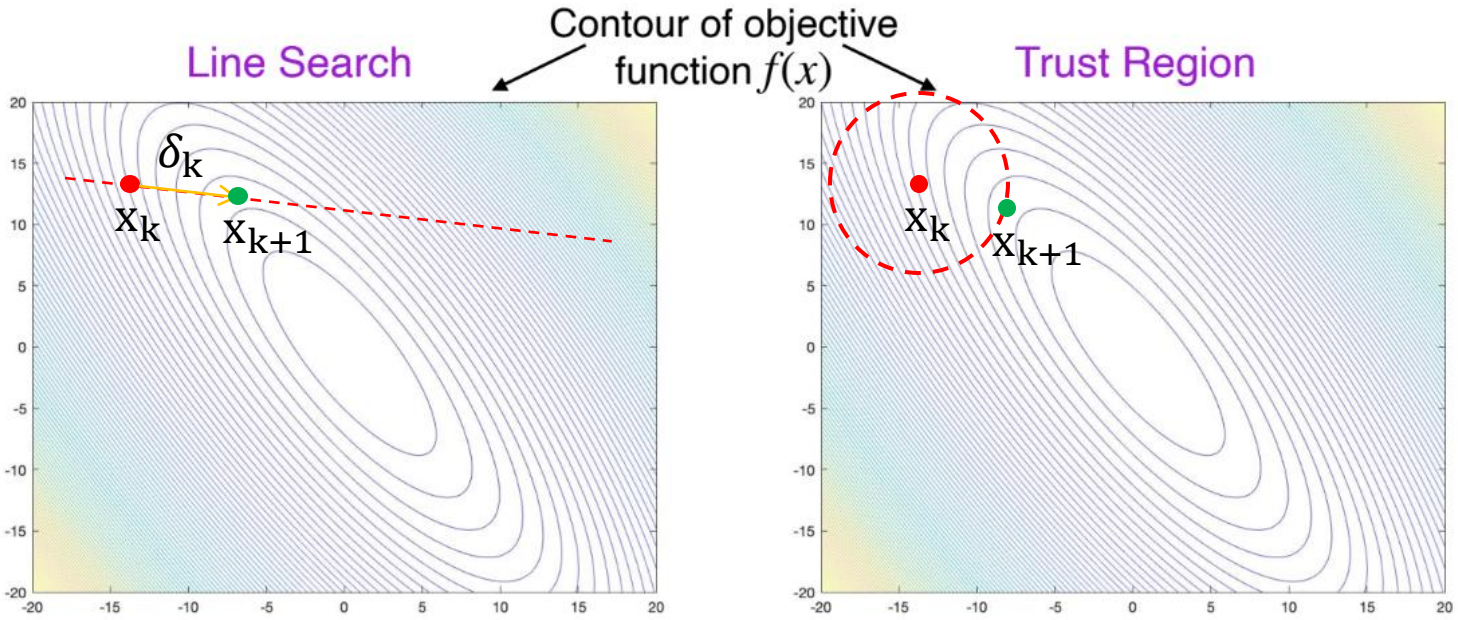


Image source Ping-Chun Hsieh

Trust Region

$$\max_{s} m_k(s)$$
$$s.t. \|s\| \leq \delta$$

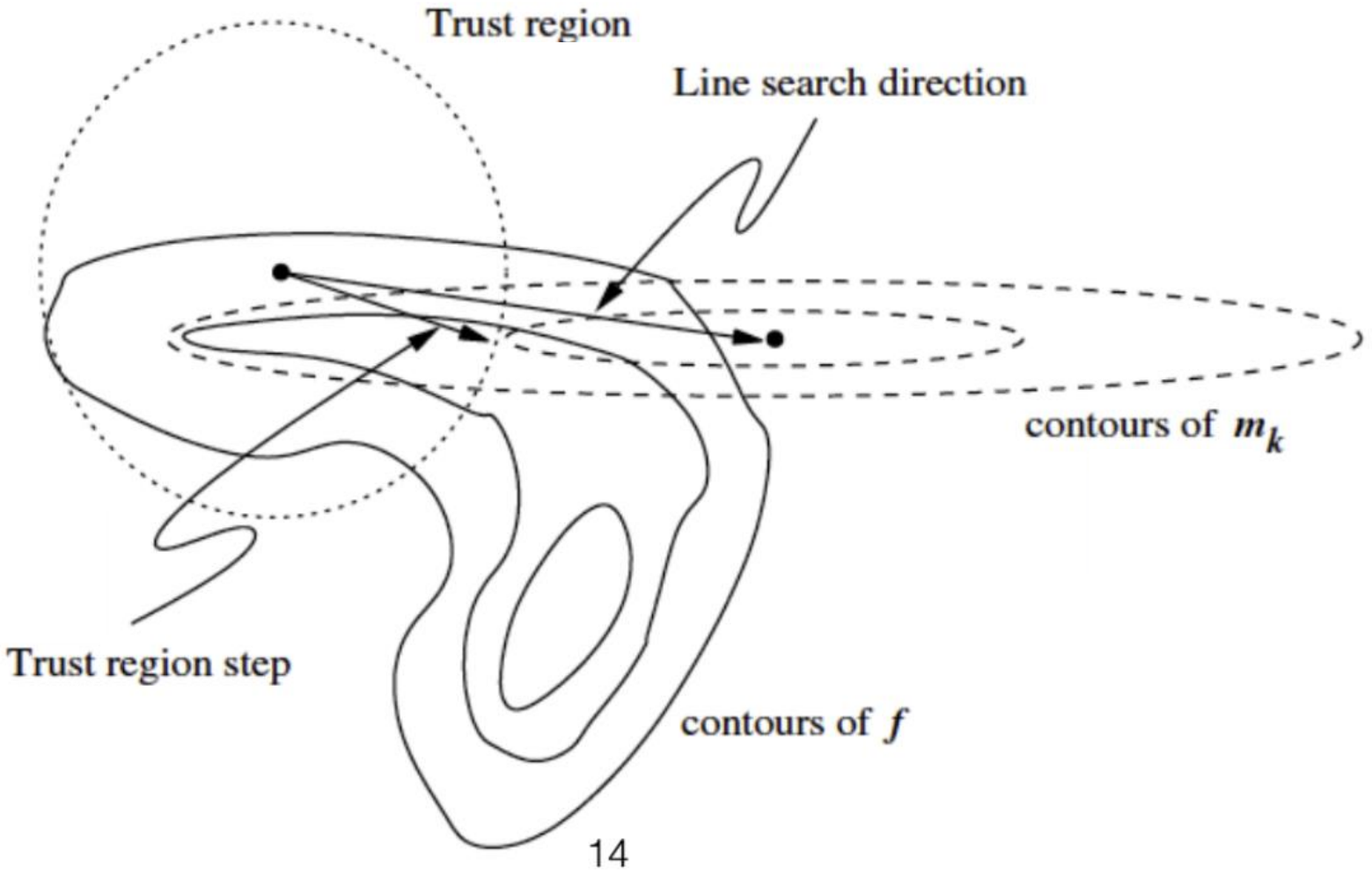


Image source Ping-Chun Hsieh

Off-policy Actor Critic with Importance Sampling

- Off-policy Actor-critic objective with importance sampling:

$$J(\theta) = \sum_{\tau \sim p_{\theta_{old}}(\tau)} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A^{\pi_{\theta_{old}}}(s_t, a_t)$$

- Policy gradients are:

Refine the current policy

$$\nabla J(\theta) = E_{\tau \sim p_{\pi_{\theta_{old}}}(\tau)} \left[\sum_{t=0}^{T} \nabla \log \pi_\theta(a_t|s_t) \left(\prod_{t'=0}^{t} \frac{\pi_\theta(a_{t'}|s_{t'})}{\pi_{\theta_{old}}(a_{t'}|s_{t'})} \right) A^{\pi_{\theta_{old}}}(s_t, a_t) \right]$$

Sample data from other policies

Importance Sampling Results in High Variance

Importance sampling is a technique for estimating expectations using samples drawn from a different distribution.

$$\mathop{\mathrm{E}}_{x\sim P}[f(x)] = \mathop{\mathrm{E}}_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right] \approx \frac{1}{|D|}\sum_{x\in D}\frac{P(x)}{Q(x)}f(x), \quad D \sim Q$$

The ratio $P(x)/Q(x)$ is the **importance sampling weight** for x.

What is the variance of an importance sampling estimator?

$$\begin{aligned}
\mathrm{var}(\hat{\mu}_Q) &= \frac{1}{N}\mathrm{var}\left(\frac{P(x)}{Q(x)}f(x)\right) \\
&= \frac{1}{N}\left(\mathop{\mathrm{E}}_{x\sim Q}\left[\left(\frac{P(x)}{Q(x)}f(x)\right)^2\right] - \mathop{\mathrm{E}}_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right]^2\right) \\
&= \frac{1}{N}\left(\mathop{\mathrm{E}}_{x\sim P}\left[\frac{P(x)}{Q(x)}f(x)^2\right] - \mathop{\mathrm{E}}_{x\sim P}[f(x)]^2\right)
\end{aligned}$$

The term in red is problematic—if $P(x)/Q(x)$ is large in the wrong places, the variance of the estimator explodes.

The variance may explode if the current policy is very different from the behavior policy

Slide credit Joshua Achiam

Objective Function using Importance Sampling

- Objective without importance sampling: $J(\theta) = \sum_{\tau \sim p_\theta(\tau)} A^{\pi_\theta}(s_t, a_t)$
- Objective with importance sampling: $\mathrm{J}^{\mathrm{IS}}(\theta) = \sum_{\tau \sim p_{\theta_{old}}(\tau)} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A^{\pi_{\theta_{old}}}(s_t, a_t)$
- Same derivatives if the current policy is close to the behavior policy

$$\nabla_\theta J^{IS}(\theta)\Big|_{\theta_{old}} = \sum_{\tau \sim p_{\theta_{old}}(\tau)} \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\Big|_{\theta_{old}} A^{\pi_{\theta_{old}}}(s_t, a_t) = \sum_{\tau \sim p_{\theta_{old}}(\tau)} \frac{\pi_\theta \nabla_\theta \log \pi_\theta}{\pi_{\theta_{old}}}\Big|_{\theta_{old}} A^{\pi_{\theta_{old}}}(s_t, a_t)$$

$$\approx \sum_{\tau \sim p_{\theta_{old}}(\tau)} \nabla_\theta \log \pi_\theta \Big|_{\theta_{old}} A^{\pi_{\theta_{old}}}(s_t, a_t) = \nabla_\theta J(\theta) \Big|_{\theta_{old}}$$

Slide adapted from J. Hui

Quick Summary

- Our goal: Optimize to achieve consistent policy improvement

  Reformulate the optimization problem:

$$\max_{\pi} \mathrm{J}(\pi) - \mathrm{J}(\pi_{\mathrm{old}})$$

Quick Summary

- Our goal: Optimize to achieve consistent policy improvement

  Reformulate the optimization problem:

$$\max_{\pi} \mathrm{J}(\pi) - \mathrm{J}(\pi_{\mathrm{old}})$$

  Solve this difficult optimization problem with the MM algorithm! What is the approximative function $\mathbf{M}$?

- Our observation: Off-policy actor critic allows sample data to be used efficiently, however, the current policy should not be very different from the behavior policy

Slide adapted from J. Hui

The Lower Bound Function

- Let's first define a function $\mathcal{L}$:

$$\mathcal{L}_{\pi_{old}}(\pi) = \frac{1}{1-\gamma} \underset{s\sim d^{\pi_{old}}, a\sim\pi_{old}}{E}\left[\frac{\pi(a|s)}{\pi_{old}(a|s)} A^{\pi_{old}}(\mathrm{s},\mathrm{a})\right] = \underset{\tau\sim\pi_{old}}{E}\left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi(a|s)}{\pi_{old}(a|s)} A^{\pi_{old}}(\mathrm{s},\mathrm{a})\right]$$

where $d^{\pi_{old}}$ is the state visit frequency under policy π_{old}

$$d^{\pi_{old}} = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi_{old})$$

Slide adapted from J. Hui

The Lower Bound Function

- Let's first define a function $\mathcal{L}$:

$$\mathcal{L}_{\pi_{old}}(\pi) = \frac{1}{1-\gamma} \underset{s\sim d^{\pi_{old}}, a\sim\pi_{old}}{E}\left[\frac{\pi(a|s)}{\pi_{old}(a|s)} A^{\pi_{old}}(\mathrm{s},\mathrm{a})\right] = \underset{\tau\sim\pi_{old}}{E}\left[\sum_{t=0}^{\infty}\gamma^t \frac{\pi(a|s)}{\pi_{old}(a|s)} A^{\pi_{old}}(\mathrm{s},\mathrm{a})\right]$$

where $d^{\pi_{old}}$ is the discounted state visit frequency under policy π_{old}

$$d^{\pi_{old}} = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi_{old})$$

- One can show that (check TRPO paper)

$$\left|\mathrm{J}(\pi) - (\mathrm{J}(\pi_{\mathrm{old}}) + \mathcal{L}_{\pi_{old}}(\pi))\right| \le \mathrm{C}\sqrt{\underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[\mathrm{s}]]}$$

$$\Rightarrow \mathrm{J}(\pi) - \mathrm{J}(\pi_{\mathrm{old}}) \ge \mathcal{L}_{\pi_{old}}(\pi) - \mathrm{C}\sqrt{\underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[\mathrm{s}]]}$$

→ Lower bound function

Slide adapted from J. Hui

Guaranteed Monotonic Improvement

- How can we show that this objective function guarantee policy improvement?

$$\mathrm{J}(\pi) - \mathrm{J}(\pi_{\mathrm{old}}) \geq \mathcal{L}_{\pi_{old}}(\pi) - \mathrm{C}\sqrt{\underset{\mathrm{s}\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[\mathrm{s}]]}$$

- Intuitively, $\pi = \pi_{\mathrm{old}}$ is a feasible point:
  - $\sqrt{\underset{\mathrm{s}\sim d^{\pi_{old}}}{E}[D_{KL}(\pi_{old}|\pi_{old})[\mathrm{s}]]} = 0$
  - $\mathcal{L}_{\pi_{old}}(\pi_{\mathrm{old}}) = \underset{\tau\sim\pi_{old}}{E}[\sum_{t=0}^{\infty}\gamma^t A^{\pi_{old}}(\mathrm{s},\mathrm{a})] = 0$

  as a result, $\mathrm{J}(\pi) - \mathrm{J}(\pi_{\mathrm{old}}) \geq 0$

Slide adapted from J. Hui

Dual Problem of the Lower Bound Function

- Maximization of the lower bound function

  KL penalized formulation

$$\max_{\pi} \mathcal{L}_{\pi_{old}}(\pi) - C\sqrt{\underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[s]]}$$

- It is mathematically equivalent to the optimization problem

  KL constrained formulation

$$\max_{\pi} \mathcal{L}_{\pi_{old}}(\pi) \quad \text{s.t.} \underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[s]] \leq \delta$$

Slide adapted from J. Hui

Natural Policy Gradient Solves the KL Constrained Problem

- KL-constrained optimization problem:

$$\max_{\pi} \mathcal{L}_{\pi_{old}}(\pi) \quad \text{s.t.} \underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[s]] \leq \delta$$

- Let's do Taylor's expansion:

π is parameterized by θ →

Let $\underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\cdot\,|\,\cdot)] = \bar{D}_{KL}(\cdot\,|\,\cdot)$ →

$$\mathcal{L}_{\theta_{old}}(\theta) \approx \mathcal{L}_{\theta_{old}}(\theta_{old}) + \nabla_\theta \mathcal{L}_{\theta_{old}}(\theta)\Big|_{\theta_{old}}^{\top}(\theta-\theta_{old}) + \cdots$$

$$\bar{D}_{KL}(\theta|\theta_{old}) \approx \bar{D}_{KL}(\theta_{old}|\theta_{old}) + \nabla_\theta \bar{D}_{KL}(\theta|\theta_{old})\Big|_{\theta_{old}}(\theta-\theta_{old})$$

$$+\frac{1}{2}(\theta-\theta_{old})^{\top}\nabla_\theta^2 \bar{D}_{KL}(\theta|\theta_{old})\Big|_{\theta_{old}}(\theta-\theta_{old})$$

Natural Policy Gradient Solves the KL Constrained Problem

- KL-constrained optimization problem:

$$\max_{\pi} \mathcal{L}_{\pi_{old}}(\pi) \quad \text{s.t.} \underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[s]] \leq \delta$$

- Let's do Taylor's expansion:

π is parameterized by θ →

Let $\underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\cdot\,|\,\cdot)] = \bar{D}_{KL}(\cdot\,|\,\cdot)$ →

$$\mathcal{L}_{\theta_{old}}(\theta) \approx \mathcal{L}_{\theta_{old}}(\theta_{old}) + \boxed{\nabla_\theta \mathcal{L}_{\theta_{old}}(\theta)\Big|_{\theta_{old}}}^{\top}(\theta-\theta_{old}) + \cdots \quad (g)$$

$$\bar{D}_{KL}(\theta|\theta_{old}) \approx \bar{D}_{KL}(\theta_{old}|\theta_{old}) + \nabla_\theta \bar{D}_{KL}(\theta|\theta_{old})\Big|_{\theta_{old}}(\theta-\theta_{old})$$

$$+\frac{1}{2}(\theta-\theta_{old})^{\top}\boxed{\nabla^2_\theta \bar{D}_{KL}(\theta|\theta_{old})\Big|_{\theta_{old}}}(\theta-\theta_{old}) \quad (H)$$

Natural Policy Gradient Solves the KL Constrained Problem

- KL-constrained optimization problem:

$$\max_{\pi} \mathcal{L}_{\pi_{old}}(\pi) \quad \text{s.t.} \underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[s]] \leq \delta$$

- Let's rewrite the KL-constrained optimization problem:

$$\theta_{k+1} = \arg\max_{\theta} g^{\top}(\theta - \theta_k) \quad \text{s.t.} \quad \frac{1}{2}(\theta - \theta_k)^{\top} H(\theta - \theta_k) \leq \delta$$

Slide adapted from J. Hui

Natural Policy Gradient Solves the KL Constrained Problem

- KL-constrained optimization problem:

$$\max_{\pi} \mathcal{L}_{\pi_{old}}(\pi) \quad \text{s.t.} \underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[s]] \leq \delta$$

- Let's rewrite the KL-constrained optimization problem:

$$\theta_{k+1} = \arg\max_{\theta} g^{\top}(\theta - \theta_k) \quad \text{s.t.} \quad \frac{1}{2}(\theta - \theta_k)^{\top} H(\theta - \theta_k) \leq \delta$$

which can be solve analytically

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^{\top}H^{-1}g}} H^{-1}g$$

Natural policy gradient is a second order optimization that is more accurate and works regardless of how the model is parameterized

Slide adapted from J. Hui

Natural Policy Gradient Has Problems

- NPG has problems:
  1. Might not be robust to trust region size δ; at some iterations δ may be too large and performance can degrade
  2. Because of quadratic approximation, KL-divergence constraint may be violated
- Solution?
  1. Require improvement in surrogate (make sure that $\mathcal{L}_{\pi_{old}}(\pi) \geq 0$)
  2. Enforce KL-constraint

Algorithm 2 Line Search for TRPO

Compute proposed policy step $\Delta_k = \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{H}_k^{-1} \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$

for $j = 0, 1, 2, ..., L$ **do**

  Compute proposed update $\theta = \theta_k + \alpha^j \Delta_k$

  if $\mathcal{L}_{\theta_k}(\theta) \geq 0$ and $\bar{D}_{KL}(\theta || \theta_k) \leq \delta$ **then**

    accept the update and set $\theta_{k+1} = \theta_k + \alpha^j \Delta_k$

    break

  end if

end for

Slide credit Joshua Achiam

Natural Policy Gradient Has Problems

- KL-constrained optimization problem:

$$\max_{\pi} \mathcal{L}_{\pi_{old}}(\pi) \quad \text{s.t.} \underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[s]] \leq \delta$$

- Let's rewrite the KL-constrained optimization problem:

$$\theta_{k+1} = \arg\max_{\theta} g^{\top}(\theta - \theta_k) \quad \text{s.t.} \quad \frac{1}{2}(\theta - \theta_k)^{\top} H (\theta - \theta_k) \leq \delta$$

which can be solve analytically

Extremely expensive to calculate…

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^{\top}H^{-1}g}} H^{-1} g$$

Natural policy gradient is a second order optimization that is more accurate and works regardless of how the model is parameterized

Slide adapted from J. Hui

Truncated Natural Policy Gradient

- Let's take a look at:

$$\Delta\theta_k = H^{-1}\mathrm{g} \Rightarrow \underbrace{H\Delta\theta_k - \mathrm{g}}_{\mathrm{f}'(\Delta\theta_k)} = 0$$

which is equivalent to

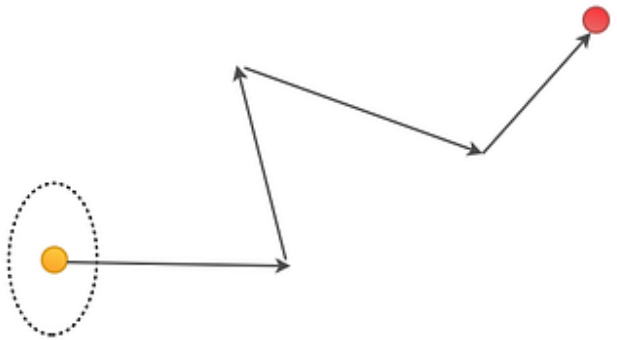
$$\min_{\Delta\theta_k} \underbrace{\frac{1}{2}\Delta\theta_k{}^{\top}\mathrm{H}\Delta\theta_k - g^{\top}\Delta\theta_k}_{\mathrm{f}(\Delta\theta_k)}$$

TRPO uses conjugate gradient method to solve the minimization problem

Slide adapted from J. Hui

Conjugate Gradient Method

- Gradient ascent: update along the steepest direction. Previous update may be undo
- Conjugate gradient: a line search method that finds update directions without undoing all previous updates. In other words, the remaining error after taking this step should be orthogonal to the current direction



Gradient ascent

Conjugate gradient

Trust Region Policy Optimization (TRPO)

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$

Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$

Perform backtracking line search with exponential decay to obtain final update

$$\theta_{k+1} = \theta_k + \alpha^j \Delta_k$$

end for

Slide credit Joshua Achiam

Proximal Policy Optimization (PPO)

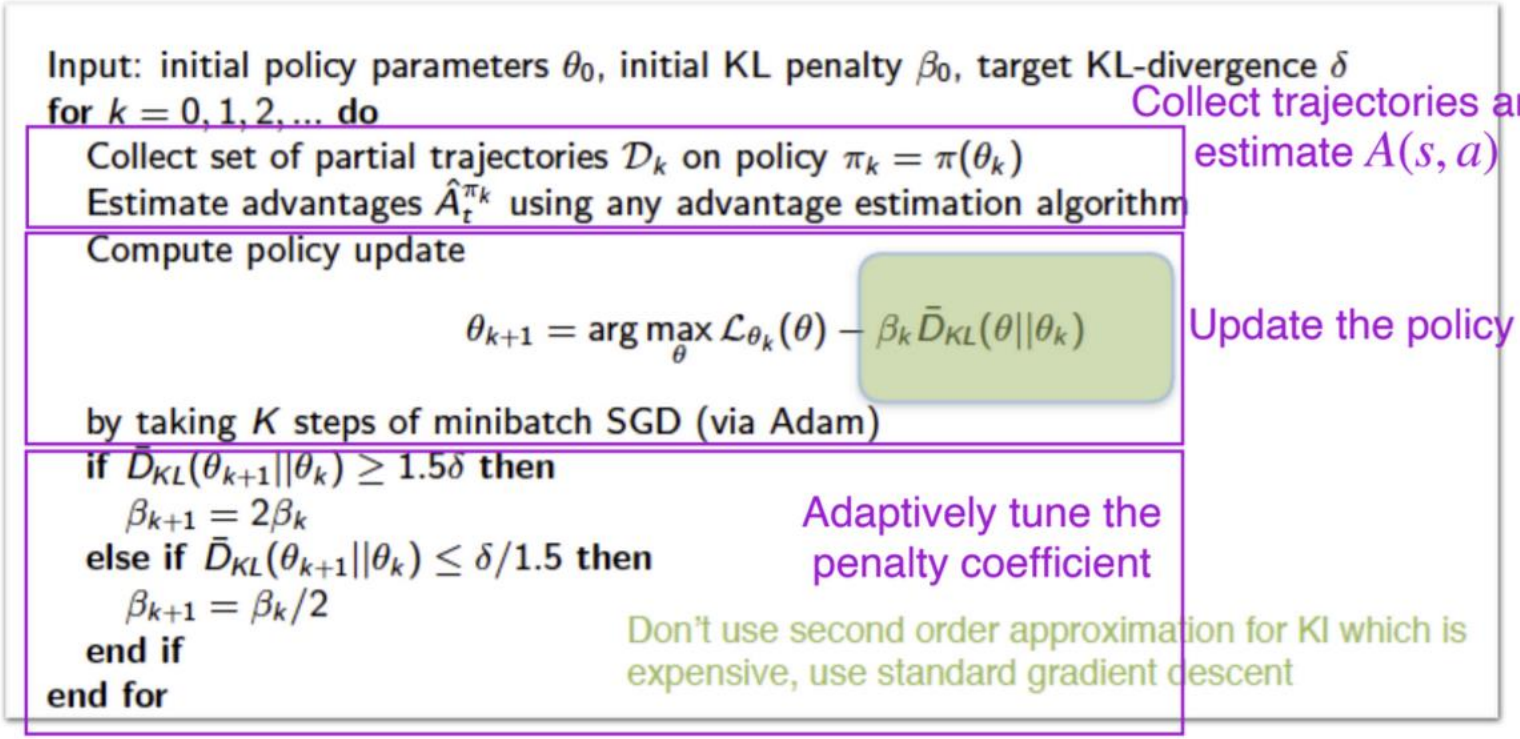
- Can we avoid calculation of Hessian matrix $\mathbf{H}$?
  - ➢ Yes, let's solve the KL-penalty optimization problem instead
- Maximization of the lower bound function

  KL penalized formulation

$$\max_{\pi} \mathcal{L}_{\pi_{old}}(\pi) - \mathrm{C}\sqrt{\underset{s\sim d^{\pi_{old}}}{E}[D_{KL}(\pi|\pi_{old})[s]]}$$

Adaptive KL Penalty

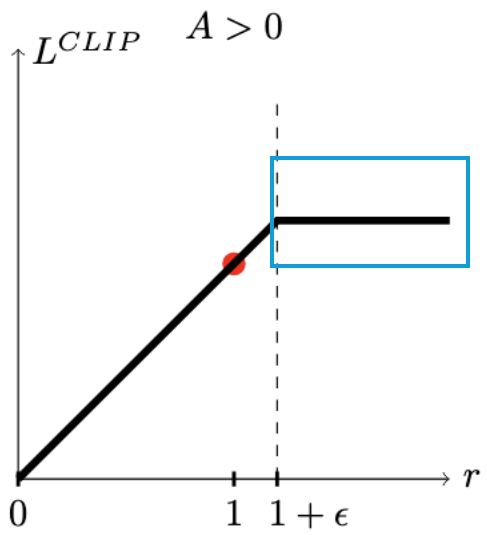
- KL penalized formulation: $\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \bar{D}_{KL}(\theta|\theta_k)$
- Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint

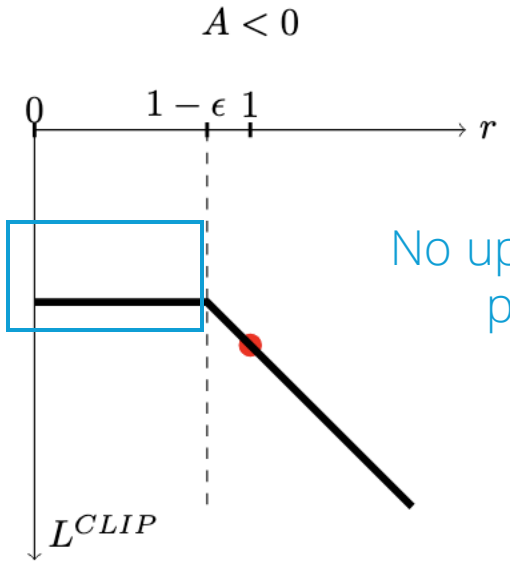


Slide credit Joshua Achiam and Ping-Chun Hsieh

Clipped Objective

- Let $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$ be the ratio of the current policy π_θ and old policy π_{θ_k}
- New objective: $\mathcal{L}^{CLIP}_{\theta_k}(\theta) = \underset{\tau\sim\pi_k}{E}\left[\min\left(r_t(\theta)\widehat{A}^{\pi_k}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}^{\pi_k}_t\right)\right]$
- Policy update: $\theta_{k+1} = \arg\max_\theta \mathcal{L}^{CLIP}_{\theta_k}(\theta)$





No update if the current and old policy are very different

Slide credit Joshua Achiam

Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ

Collect trajectories and estimate $A(s, a)$

for $k = 0, 1, 2, \ldots$ **do**

Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

by taking K steps of minibatch SGD (via Adam), where

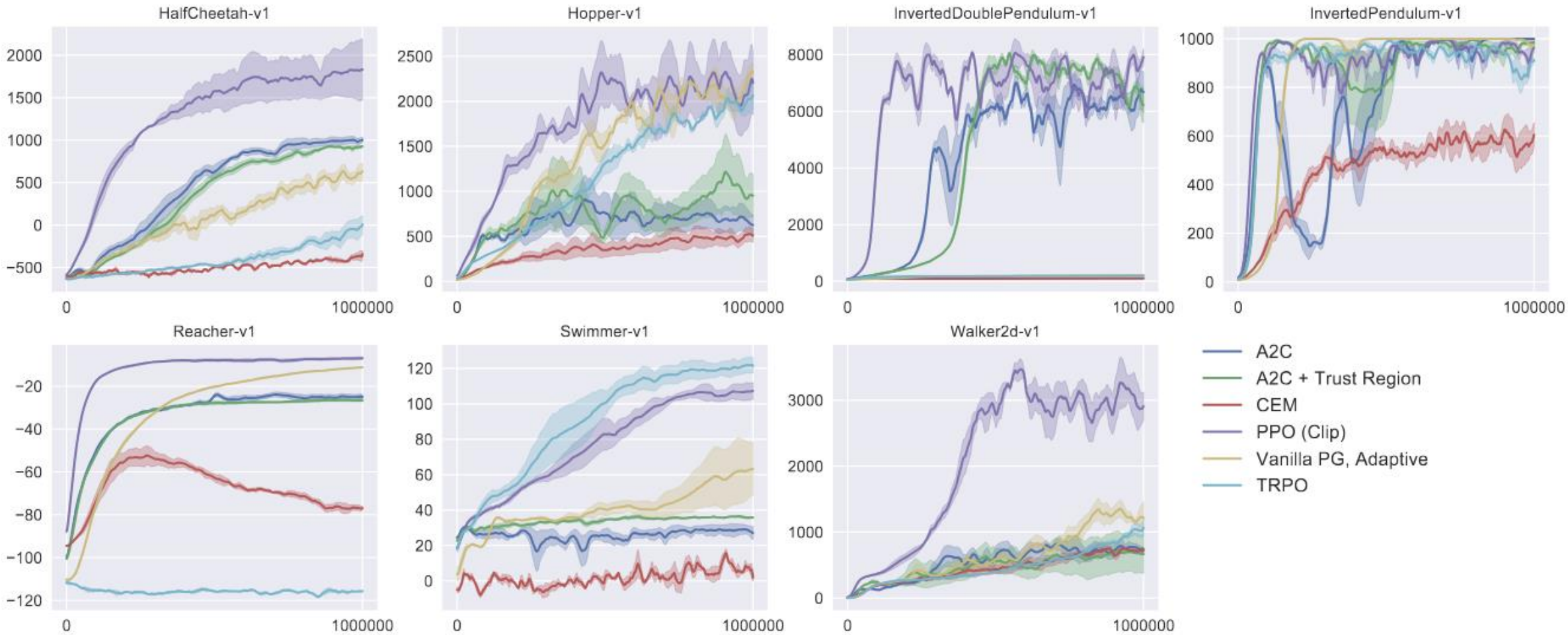
$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k} \left[\sum_{t=0}^{T} \left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k}) \right] \right]$$

end for

PPO does not use 2nd-order derivatives

Slide credit Ping-Chun Hsieh

PPO is a Strong Baseline for Many RL Tasks



Reinforcement Learning vs. Imitation Learning

- Imitation learning:
  - Pros: IL is sample efficient
    1. It bootstraps from human knowledge
    2. It produces an ok-ish model from a **relatively** small dataset
  - Cons: IL fails to generalize
    1. It struggles with distributional drift

- Reinforcement learning:
  - Pros: RL is able to generalize
    1. It automatically discovers feasible strategies by trial and error
  - Cons: RL is sample inefficient
    1. It requires a lot of sample data before discovering the solution

IL is Incapable of Generalization

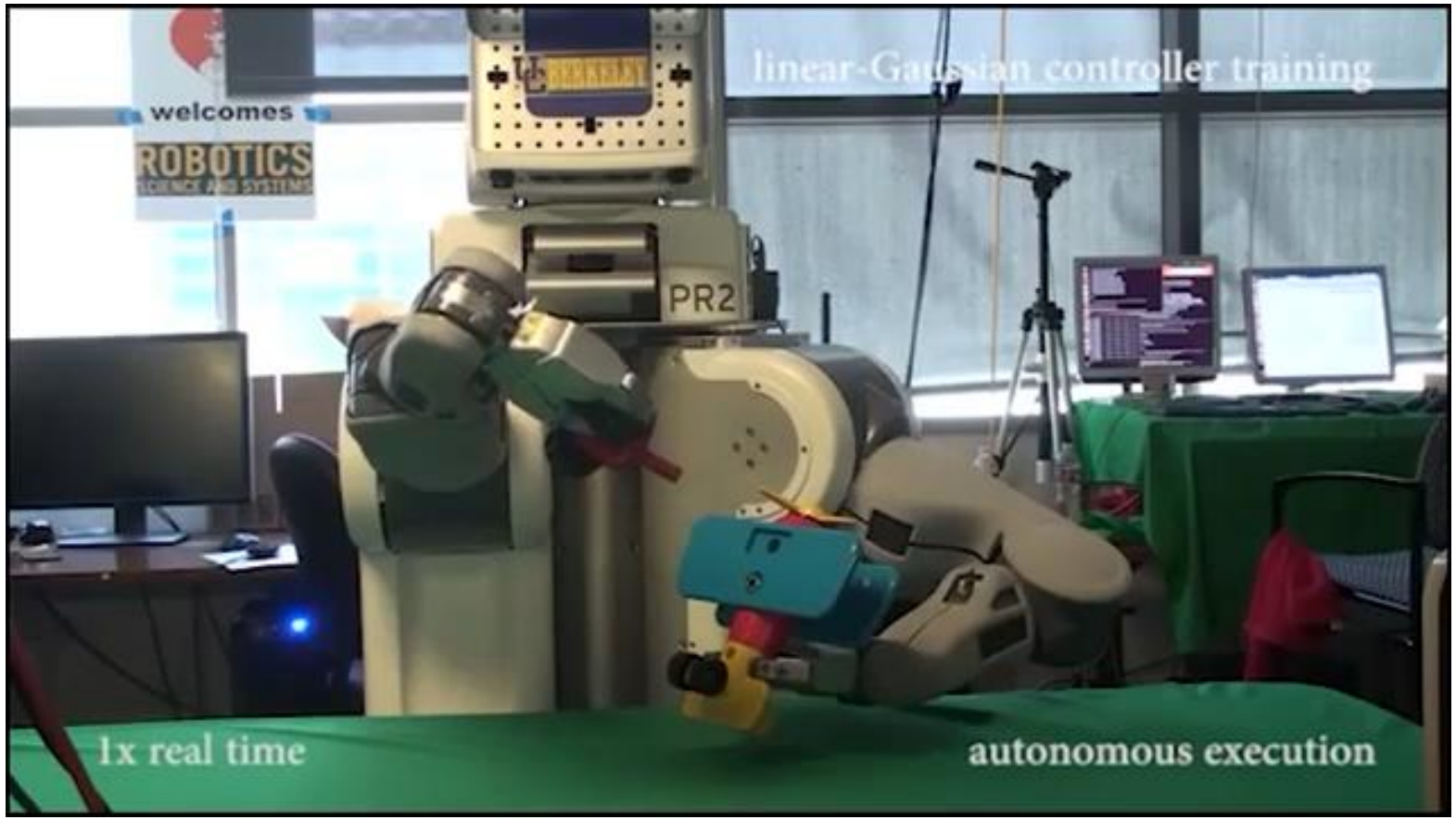
Succeed in the real world

Fail in the digital twin

OpenVLA: An Open-Source Vision-Language-Action Model

Idea: Enhance Generalization of IL Policy with RL

https://www.youtube.com/watch?v=kFPcoclV5Vo



https://www.youtube.com/watch?v=JeVppkoloXs&t=11s

Step 1: Implement the Actor and Critic Network

PPO algorithm

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, \ldots$ do
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta||\theta_k)$$

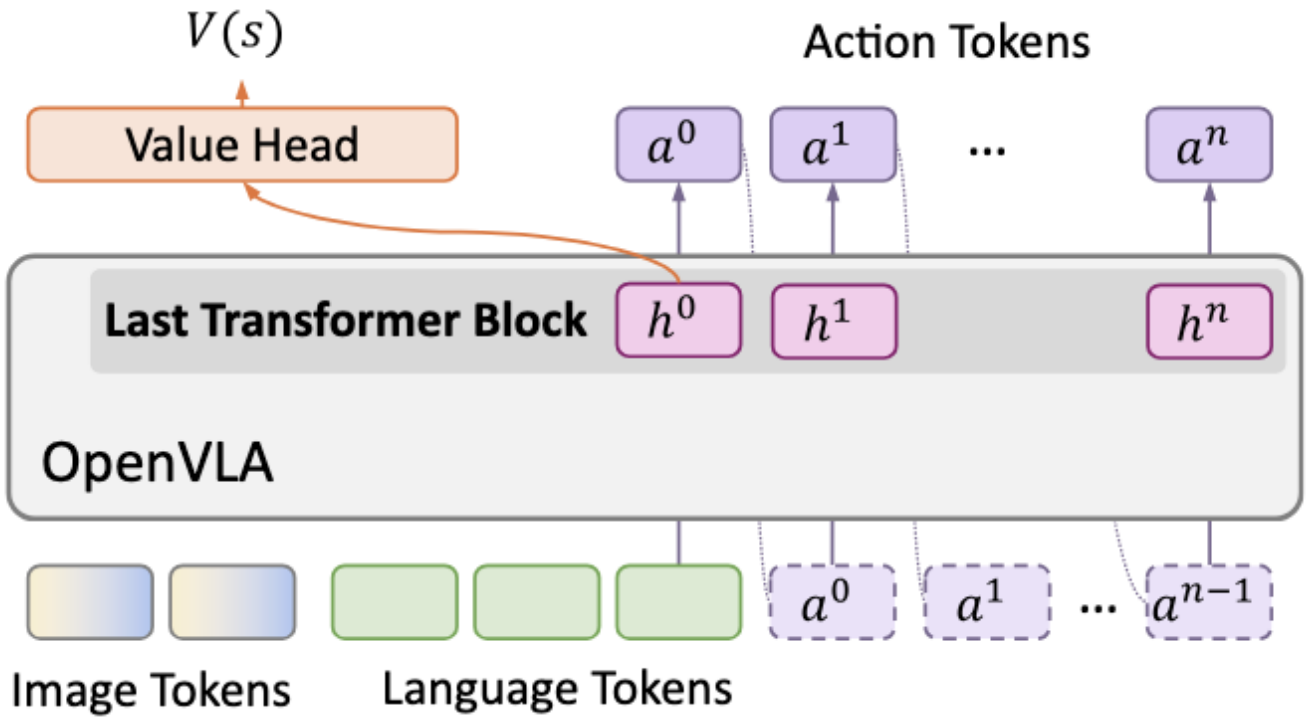
  by taking K steps of minibatch SGD (via Adam)
  if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \geq 1.5\delta$ then
    $\beta_{k+1} = 2\beta_k$
  else if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \leq \delta/1.5$ then
    $\beta_{k+1} = \beta_k/2$
  end if
end for

Collect trajectories and estimate $A(s, a)$

Update the policy

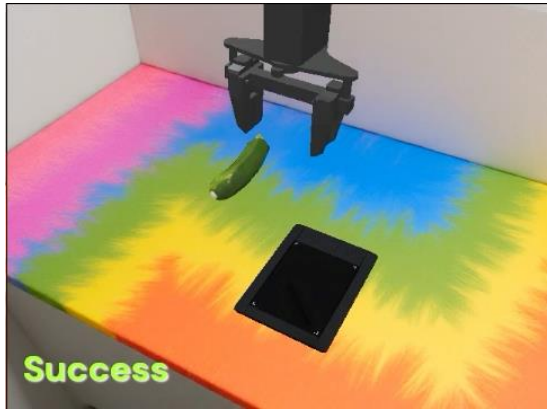
Adaptively tune the penalty coefficient

Don't use second order approximation for Kl which is expensive, use standard gradient descent

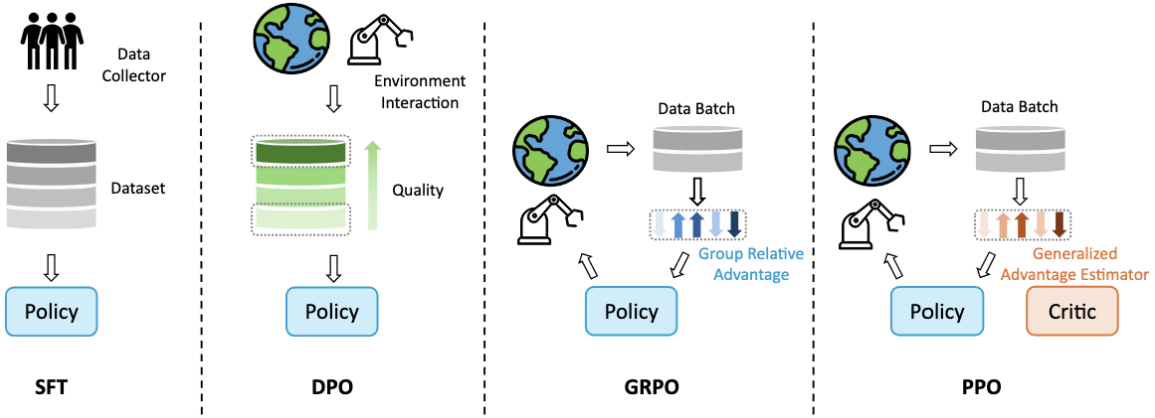


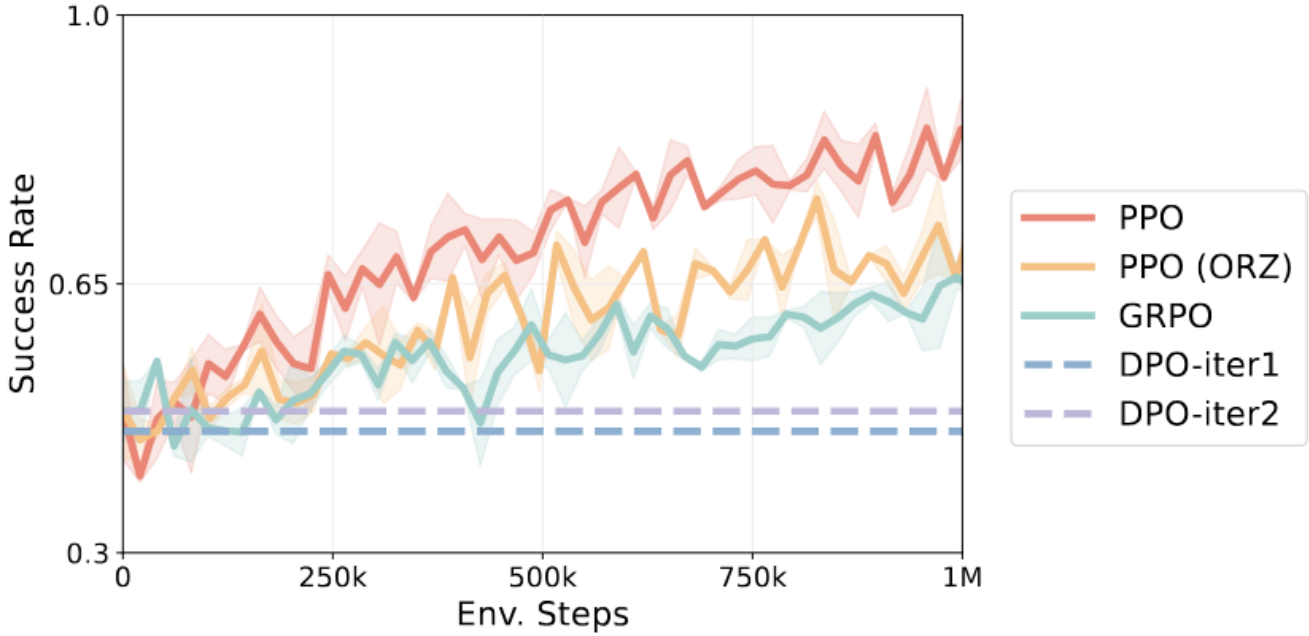
What Can RL Bring to VLA Generalization? An Empirical Study. Liu et al.

Step 2: Create Diverse Environments

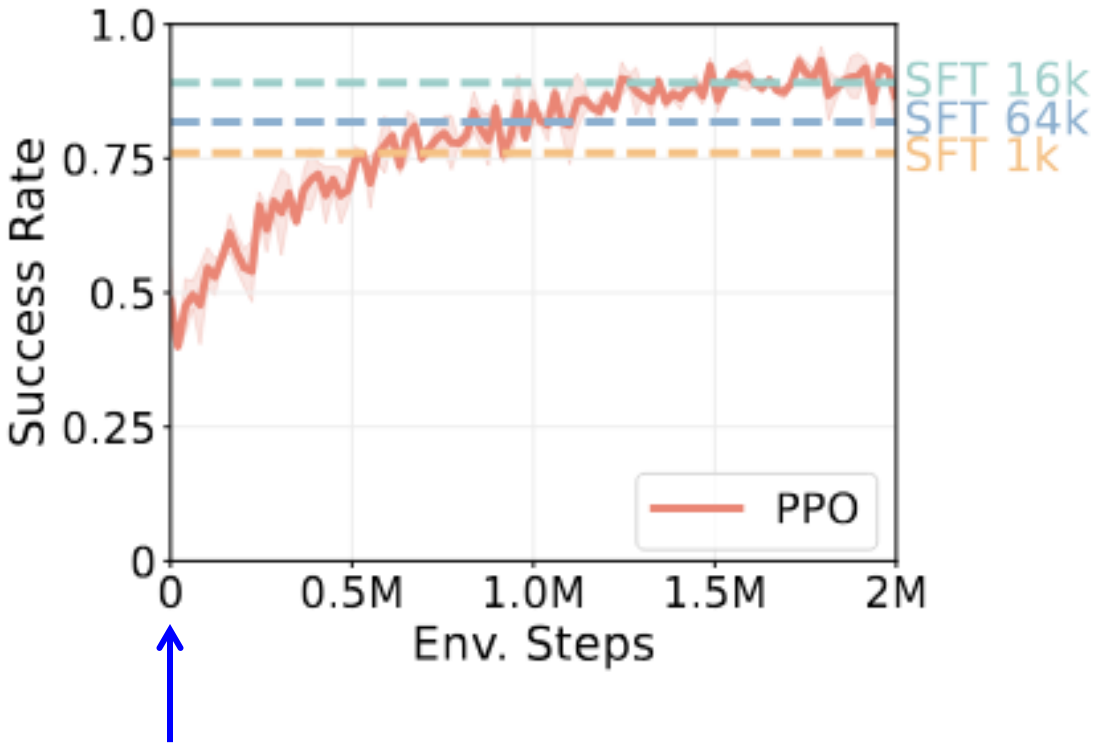


Step 3: Fine-tune IL Policy with PPO

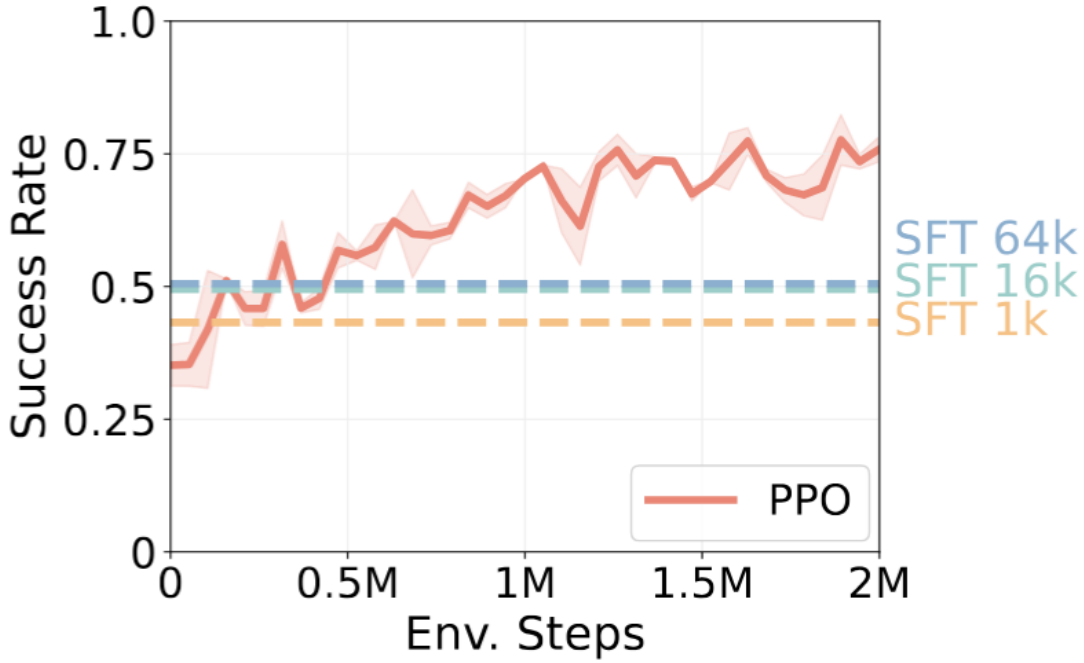




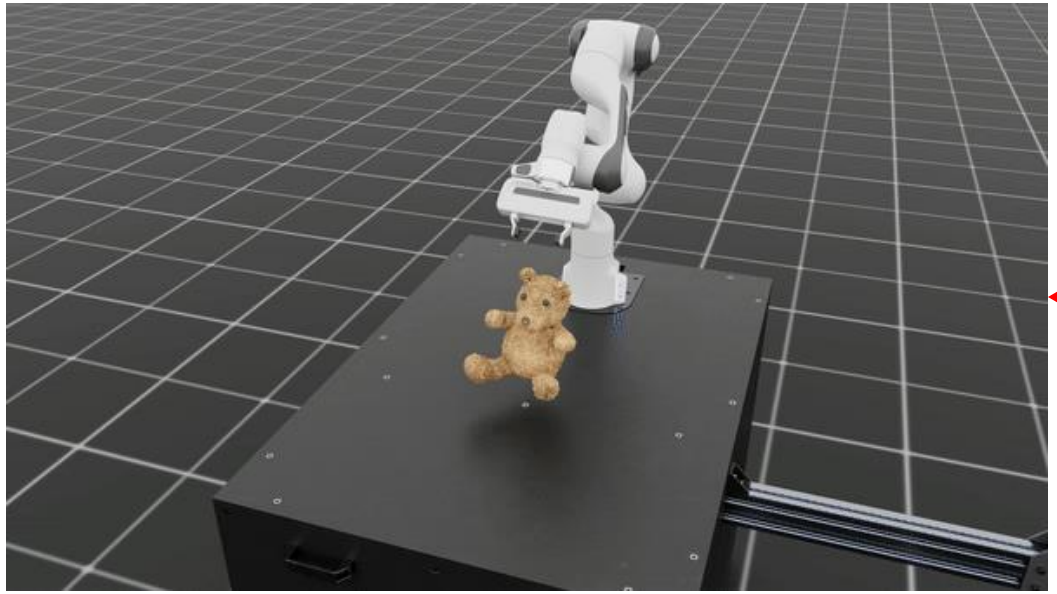
Results: Policy Generalizes to the Current Simulation Environment



Before RL finetuning

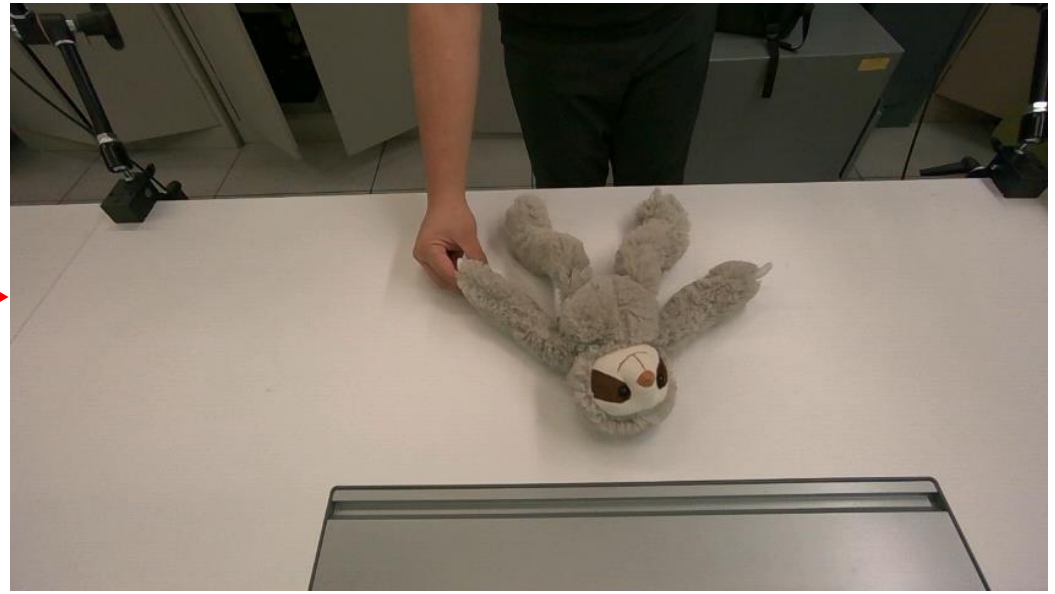


However, This Approach is Only Demonstrated in Simulation Environment. We need Real-world RL…



Video source: Isaac Lab

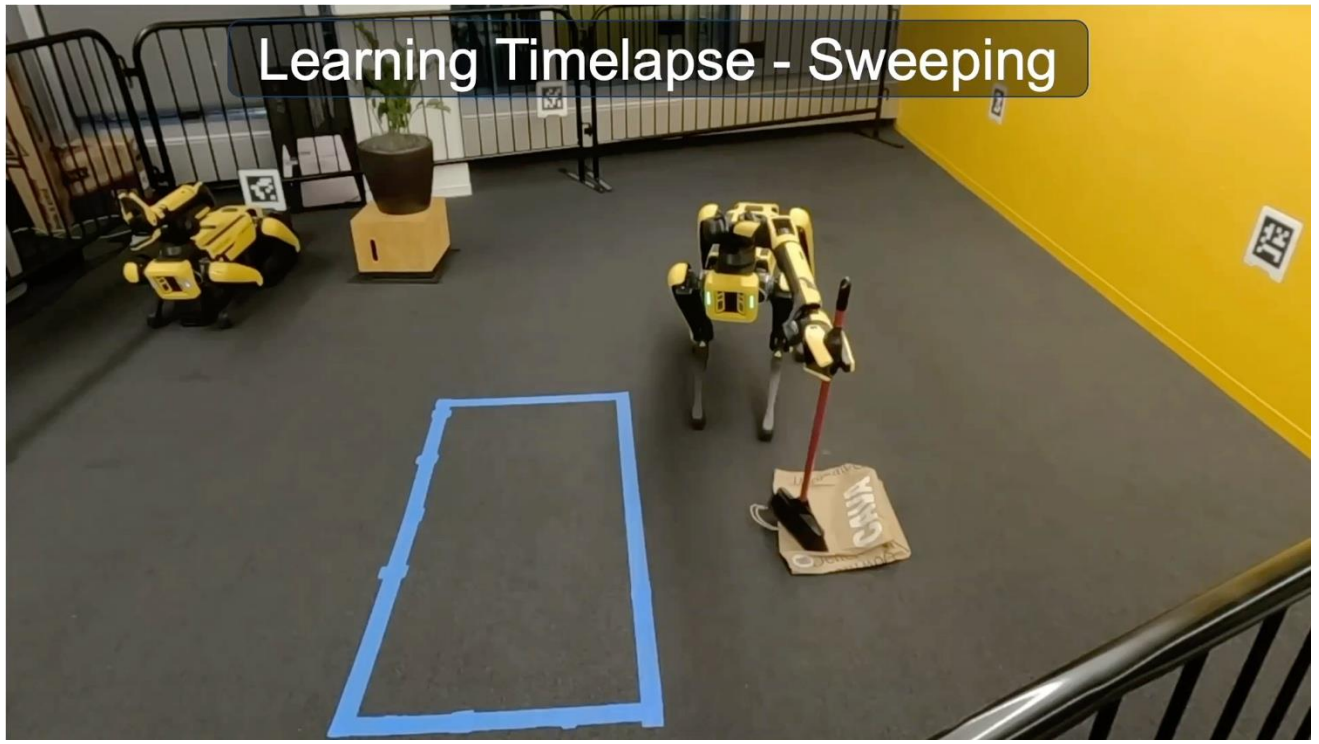
Sim2real gap



Video source: PhysTwin

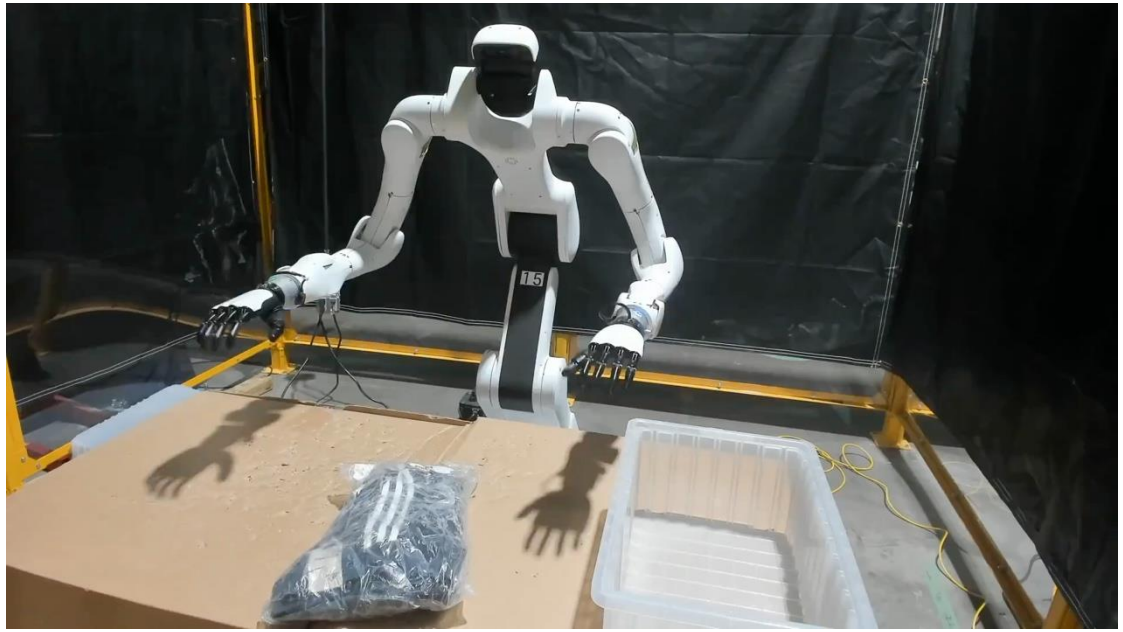
Can We Do RL in the Real World?

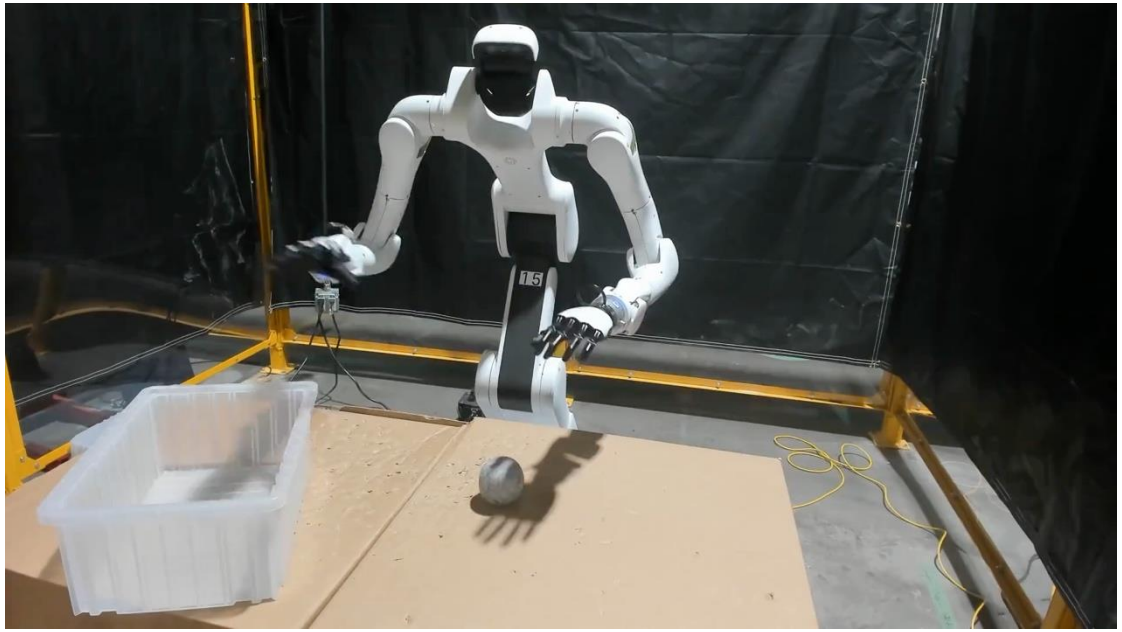
MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale. Kalashnikov et al.



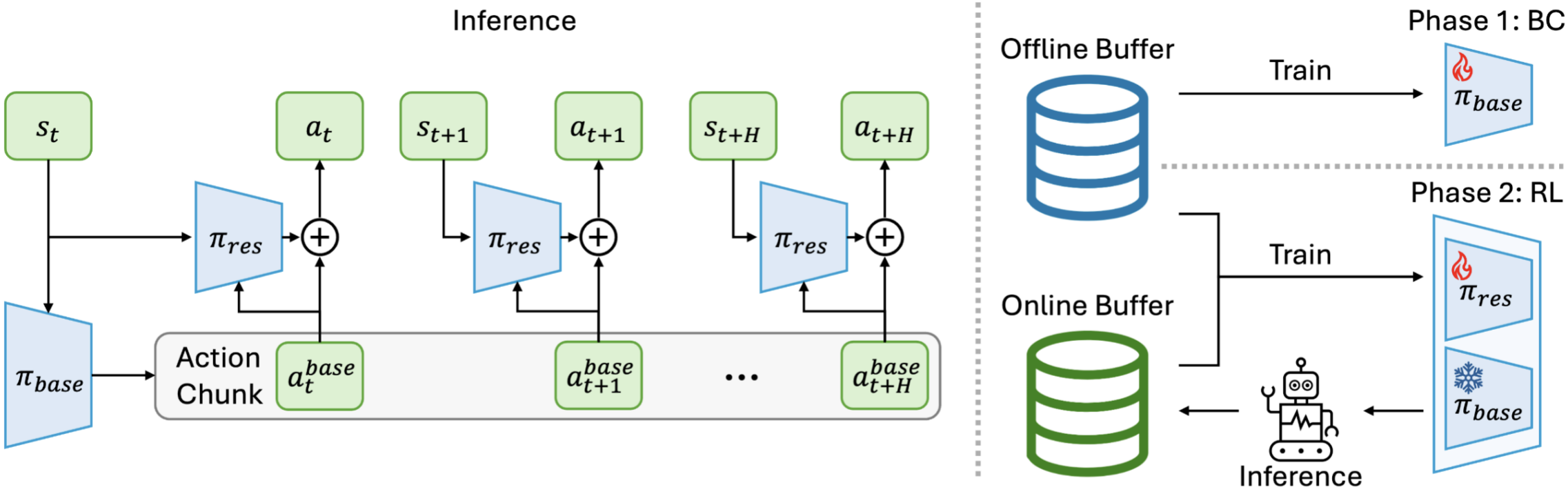
Continuously Improving Mobile Manipulation with Autonomous Real-World RL. Mendonca et al.

Supervised Fine-tuning Sometimes Doesn't Work Out-of-Box





Adapt the Policy with Real-world Residual RL



Fine-tuning with Off-policy Residual RL

Algorithm 1 ResFiT: Residual fine-tuning w/ Off-Policy RL

Input: pre-trained base π_{b}, dataset of demos $\mathcal{D}_{\text{offline}}$
Initialize: residual policy π_θ, Q ensemble $Q_{\phi_1}, \dots Q_{\phi_N}$, vision encoder f_ω, empty buffer $\mathcal{D}_{\text{online}}$
Initialize: set target networks $\theta' \leftarrow \theta$, $\phi'_i \leftarrow \phi_i$

for step $= 1, 2, \dots,$ `buffer_warmup_steps` **do**
  Sample noise $\epsilon_t \sim \mathcal{U}(-\text{noise\_scale}, \text{noise\_scale})$
  Step env with $a_t = \epsilon_t + a_t^{\text{b}}$ where $a_t^{\text{b}} \sim \pi_{\text{b}}(s_t)$
  Observe next state s_{t+1}, reward r_t, done flag d_t
  Add transition $(s_t, a_t^{\text{b}}, a_t, s_{t+1}, a_{t+1}^{b}, r_t, d_t)$ to $\mathcal{D}_{\text{online}}$
end for
repeat
  $a_t^{\text{b}} \sim \pi_{\text{b}}(s_t)$ and $a_t^{\text{r}} \sim \pi_\theta(s_t, a_t^{\text{b}})$
  Step env with $a_t = a_t^{\text{b}} + a_t^{\text{r}}$
  Add $(s_t, a_t^{\text{b}}, a_t, s_{t+1}, a_{t+1}^{b}, r_t, d_t)$ to $\mathcal{D}_{\text{online}}$
  Repeat UTD times:
    Sample batch B evenly from $\mathcal{D}_{\text{offline}}$, $\mathcal{D}_{\text{online}}$
    Update critic using B:
      $a_{t+1}^{\text{r}} \sim \pi_{\theta'}(s_{t+1}, a_{t+1}^{\text{b}})$
      Compute $a_{t+1} = a_{t+1}^{\text{b}} + a_{t+1}^{\text{r}}$
      $y = r_t + (1 - d_t) * \gamma * \min_{\text{subset(i)}} Q_{\phi'_i}(s_{t+1}, a_{t+1})$
      Update ϕ_i, ω to minimize $\text{MSE}(Q_{\phi_i}(s_t, a_t), y)$
      Update critic targets $\phi'_i \leftarrow \rho\phi'_i + (1 - \rho)\phi_i$
  Update actor using latest B:
    Update θ to maximize:
      $\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_i}(s_t, \pi_\theta(s_t, a_t^{\text{b}}) + a_t^{\text{b}})$
    Update actor target $\theta' \leftarrow \rho\theta' + (1 - \rho)\theta$
until convergence

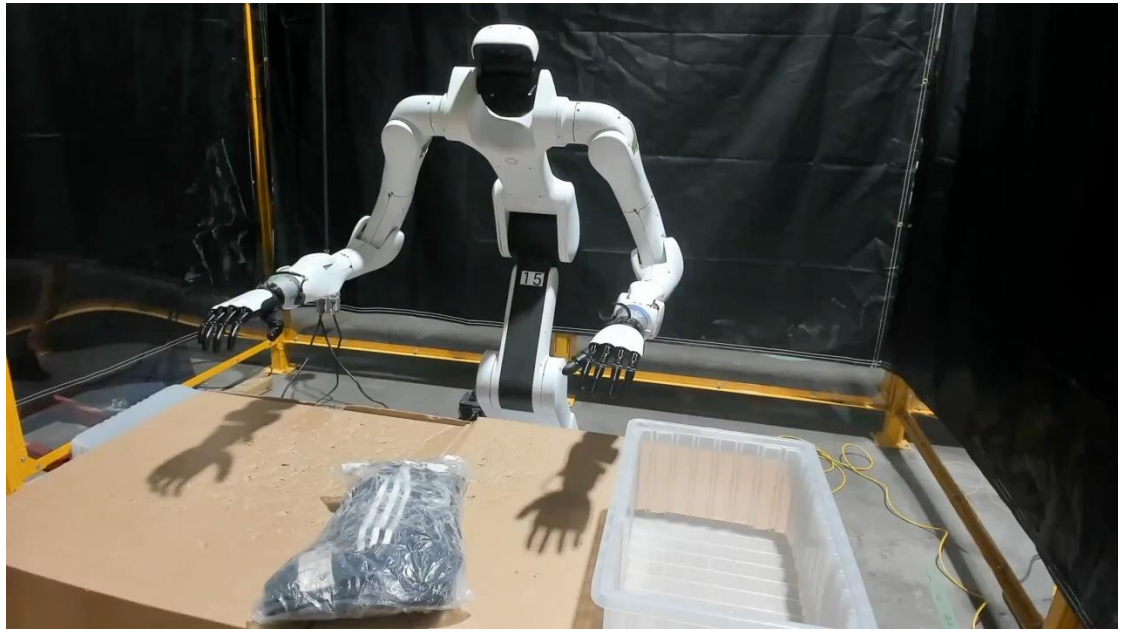
Warm-up replay buffer

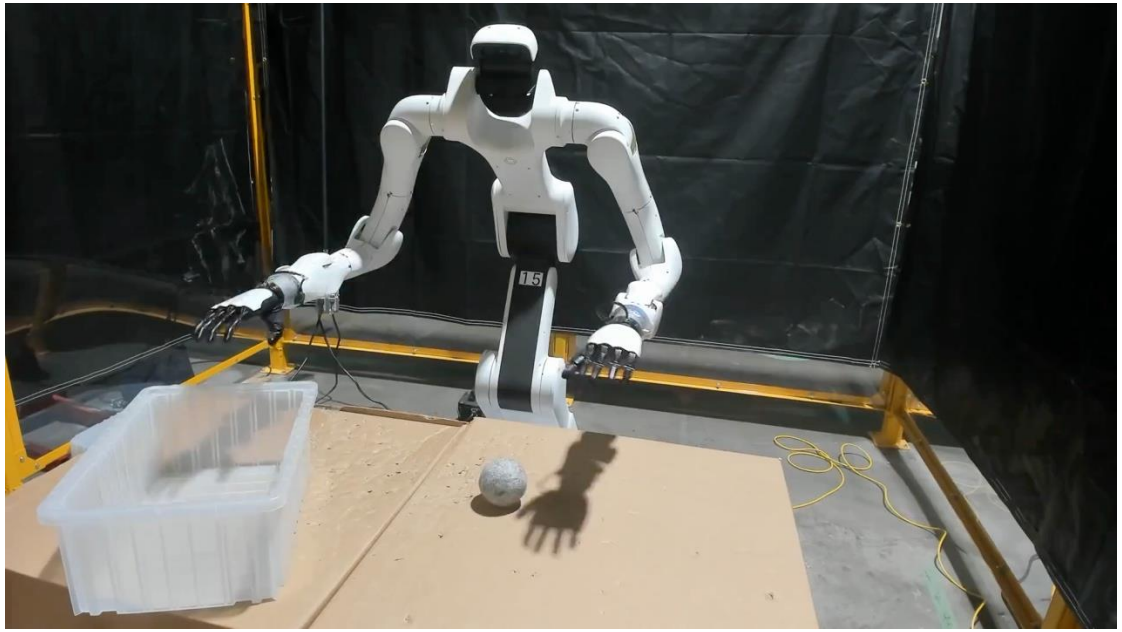
Fit the value network

Fit the policy network

Residual Off-Policy RL for Finetuning Behavior Cloning Policies. Ankile et al.

Real-world RL Enhances the Policy

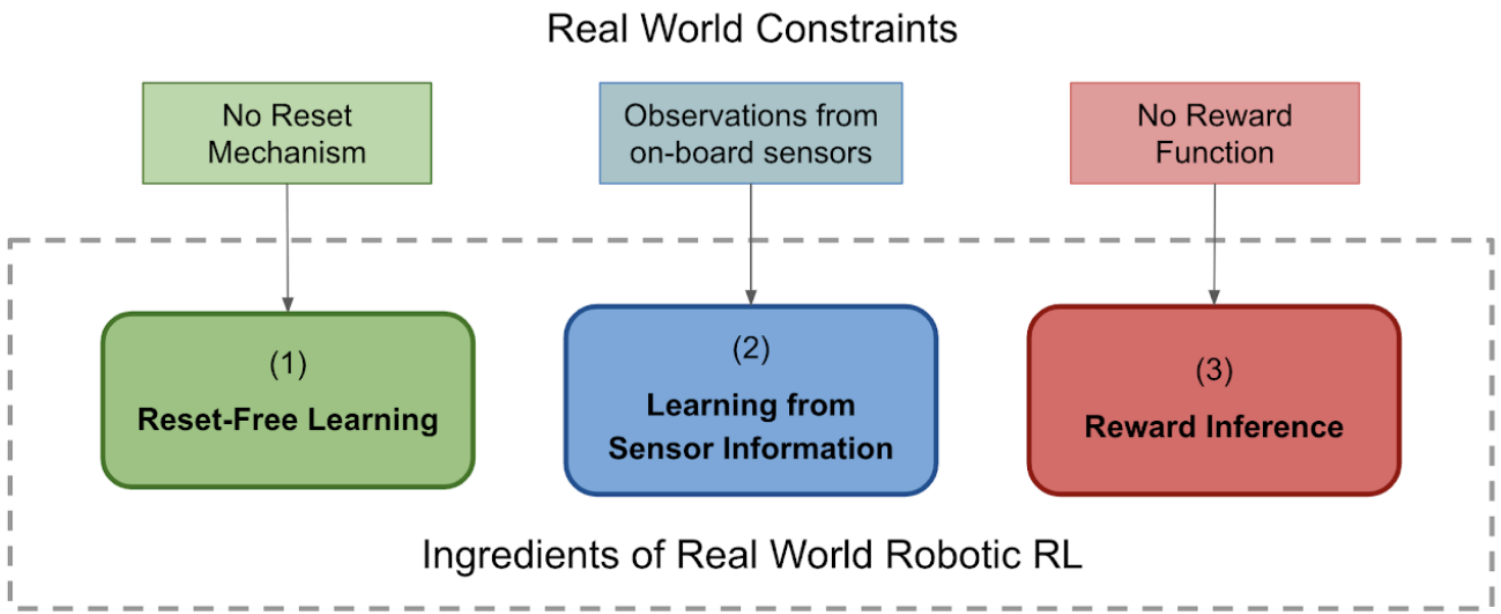




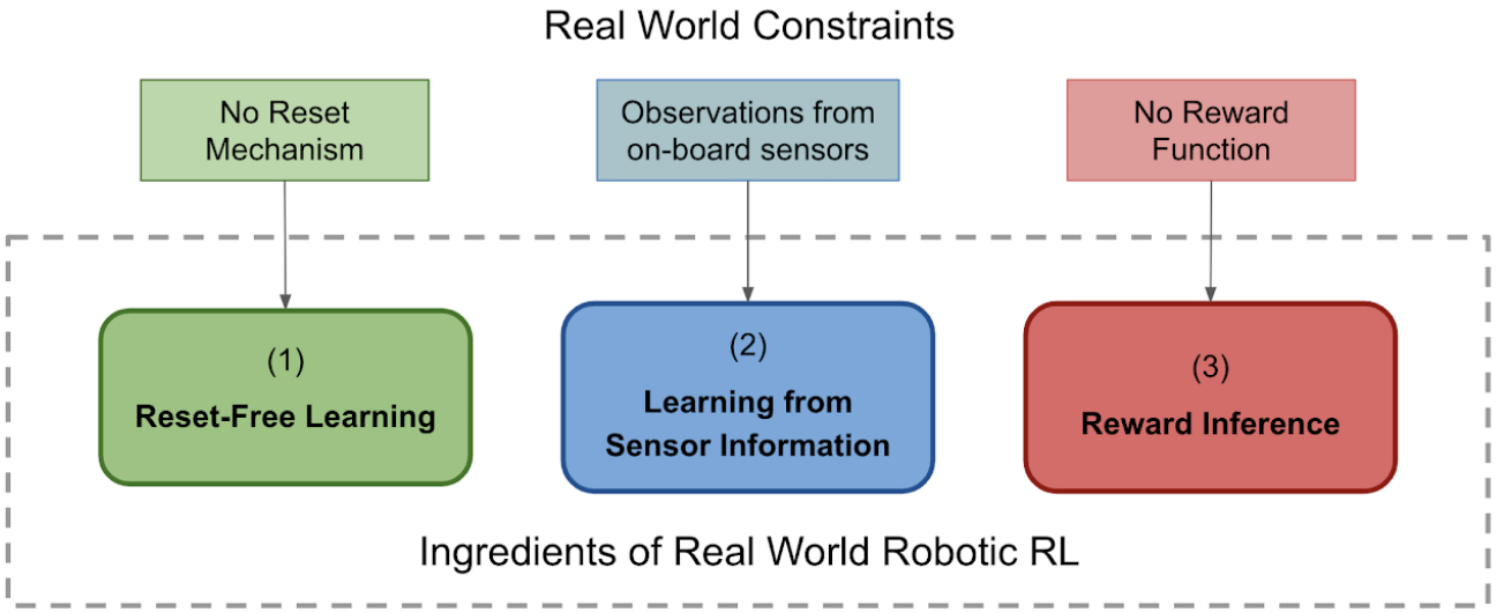
Question: What are the Challenges of Real-World RL?

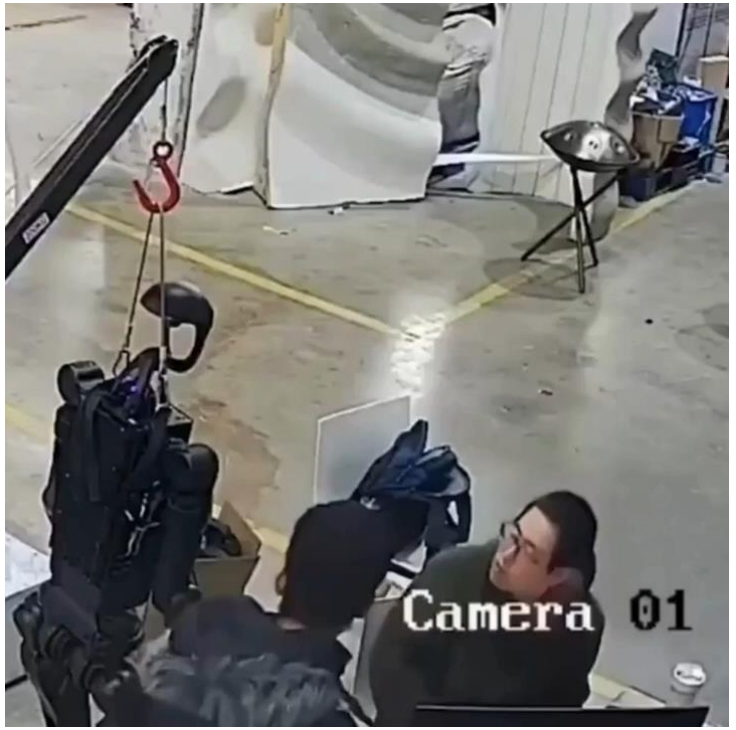
THE INGREDIENTS OF REAL-WORLD ROBOTIC REINFORCEMENT LEARNING

Henry Zhu[*1], **Justin Yu**[*1], **Abhishek Gupta**[*1], **Dhruv Shah**[1],
Kristian Hartikainen[2], **Avi Singh**[1], **Vikash Kumar**[3], **Sergey Levine**[1]
[1] University of California, Berkeley [2] University of Oxford [3] University of Washington



Don’t Forget Safety! But How?





https://youtube.com/shorts/RKUeRVTxzjY?si=BpQ7Fez72DkKfUDf