Deep RL Foundations in 6 Lectures

Lecture 5: DDPG and SAC

Pieter Abbeel

Lecture Series

- Lecture 1: MDPs Foundations and Exact Solution Methods
- Lecture 2: Deep Q-Learning
- Lecture 3: Policy Gradients, Advantage Estimation
- Lecture 4: TRPO, PPO
- ***<u>Lecture 5: DDPG, SAC</u>***
- Lecture 6: Model-based RL

Outline for This Lecture

- ***<u>Deep Deterministic Policy Gradient (DDPG)</u>***
- Soft Actor Critic (SAC)

Deep Deterministic Policy Gradient (DDPG)

- for iter = 1, 2, …

  Roll-outs:
  Execute roll-outs under current policy (+some noise for exploration)

  Q function update:

  $$g \propto \nabla_\phi \sum_t (Q_\phi(s_t, u_t) - \hat{Q}(s_t, u_t))^2 \quad \text{with} \quad \hat{Q}(s_t, u_t) = r_t + \gamma Q_\phi(s_{t+1}, u_{t+1})$$

  Policy update:
  Backprop through Q to compute gradient estimates for all t:

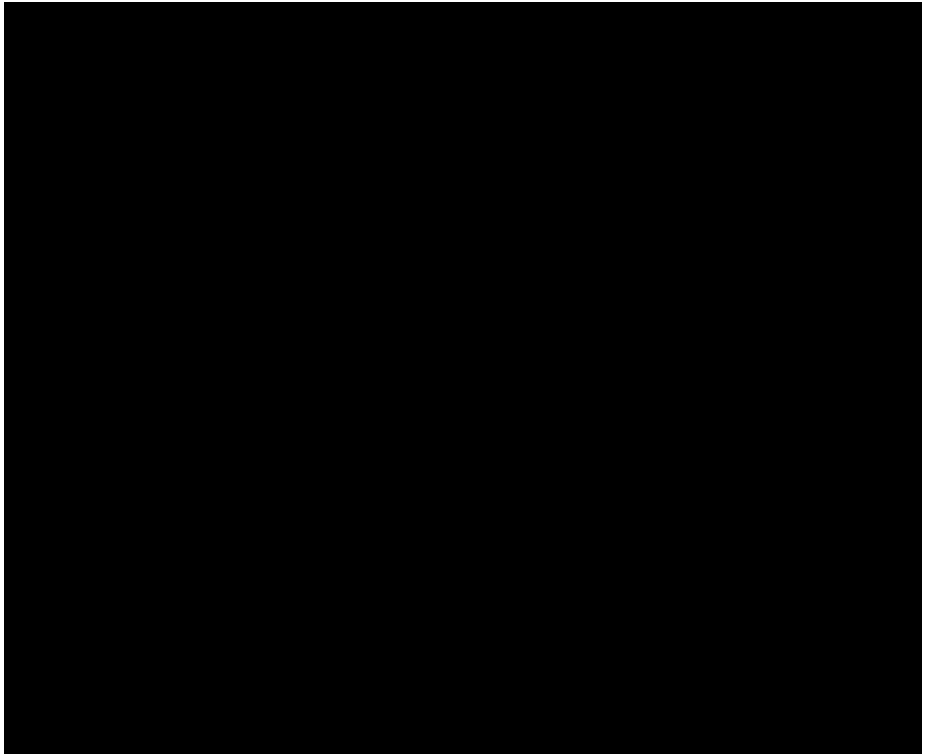
  $$g \propto \sum_t \nabla_\theta Q_\phi(s_t, \pi_\theta(s_t, v_t))$$

Deep Deterministic Policy Gradient (DDPG)

- Add noise for exploration
- Incorporate replay buffer and target network ideas from DQN for increased stability
- Use lagged (Polyak-averaging) version of Q_ϕ and π_θ for target values $\hat{Q}_t$

$$\hat{Q}_t = r_t + \gamma Q_{\phi'}(s_{t+1}, \pi_{\theta'}(s_{t+1}))$$

DDPG



DDPG

\+ very sample efficient thanks to off-policy updates

\- often unstable

→ Soft Actor Critic (SAC), which adds entropy of policy to the objective, ensuring better exploration and less overfitting of the policy to any quirks in the Q-function

Outline for This Lecture

- Deep Deterministic Policy Gradient (DDPG)
- ***<u>Soft Actor Critic (SAC)</u>***

Soft Policy Iteration

1. Soft policy evaluation:

Fix policy, apply soft Bellman backup until converges:

$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \mathbb{E}_{\mathbf{s}'\sim p_{\mathbf{s}},\ \mathbf{a}'\sim\pi}\left[Q(\mathbf{s}',\mathbf{a}') - \log\pi(\mathbf{a}'|\mathbf{s}')\right]$$

This converges to Q^π.

2. Soft policy improvement:

Update the policy through information projection:

$$\pi_{\mathrm{new}} = \arg\min_{\pi'} \mathrm{D_{KL}}\left(\pi'(\,\cdot\,|\mathbf{s})\,\middle\|\,\frac{1}{Z}\exp Q^{\pi_{\mathrm{old}}}(\mathbf{s},\,\cdot\,)\right)$$

For the new policy, we have $Q^{\pi^{\mathrm{new}}} \geq Q^{\pi^{\mathrm{old}}}$.

3. Repeat until convergence

Soft Actor-Critic

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor.* ICML, 2018.

1. Take one stochastic gradient step to minimize soft Bellman residual
2. Take one stochastic gradient step to minimize the KL divergence
3. Execute one action in the environment and repeat

Soft Actor Critic

- Objective: $$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))\right]$$
- Iterate:
  - Perform roll-out from pi, add data in replay buffer
  - Learn

$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}}\left[\tfrac{1}{2}\left(V_\psi(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t\sim\pi_\phi}\left[Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right]\right)^2\right]$$

$$\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\,\mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V_{\bar{\psi}}(\mathbf{s}_{t+1})\right]$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi_\phi(\,\cdot\,|\mathbf{s}_t)\,\middle\|\,\frac{\exp\left(Q_\theta(\mathbf{s}_t,\,\cdot\,)\right)}{Z_\theta(\mathbf{s}_t)}\right)\right]$$

[see also: https://towardsdatascience.com/soft-actor-critic-demystified-b8427df61665]

Algorithm 1 Soft Actor-Critic

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$

Set target parameters equal to main parameters $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$

repeat

  Observe state s and select action $a \sim \pi_\theta(\cdot|s)$

  Execute a in the environment

  Observe next state s', reward r, and done signal d to indicate whether s' is terminal

  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$

  If s' is terminal, reset environment state.

  if it's time to update **then**

    for j in range(however many updates) **do**

      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$

      Compute targets for the Q functions:

$$y(r, s', d) = r + \gamma(1 - d)\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} \left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2 \qquad \text{for } i = 1, 2$$

      Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s)|\, s\right)\right),$$

      where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt θ via the reparametrization trick.

      Update target networks with

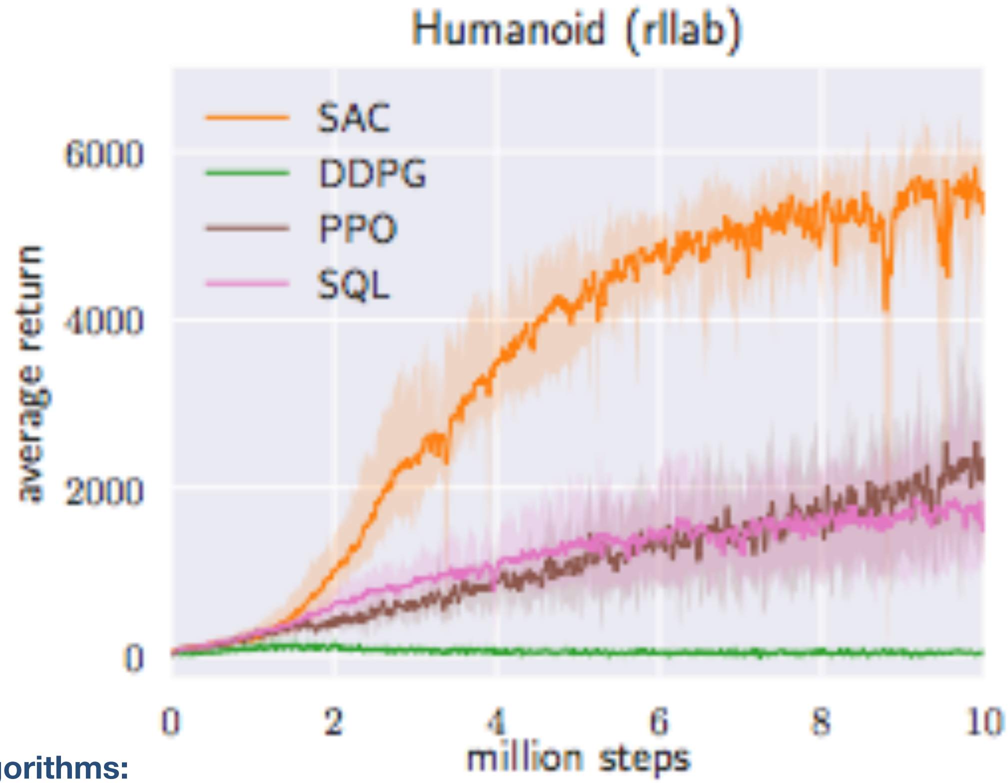
$$\phi_{\text{targ},i} \leftarrow \rho\phi_{\text{targ},i} + (1 - \rho)\phi_i \qquad \text{for } i = 1, 2$$

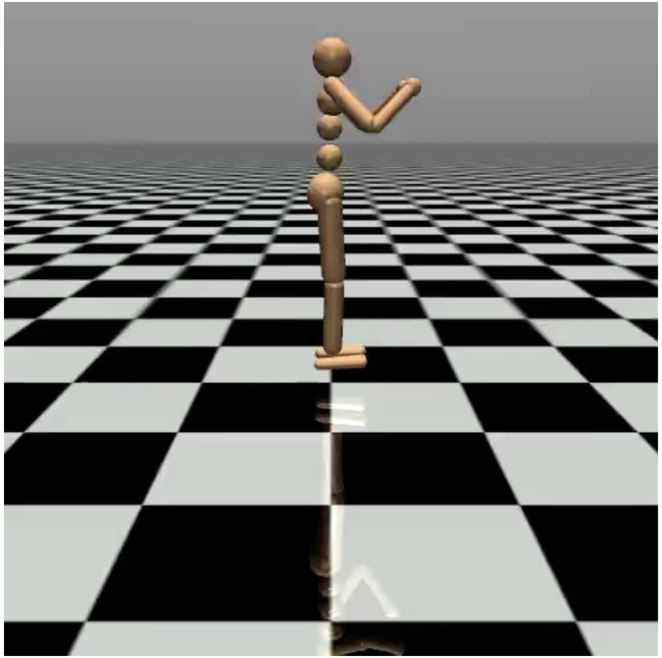
    end for

  end if

until convergence

Source: OpenAI Spinning Up

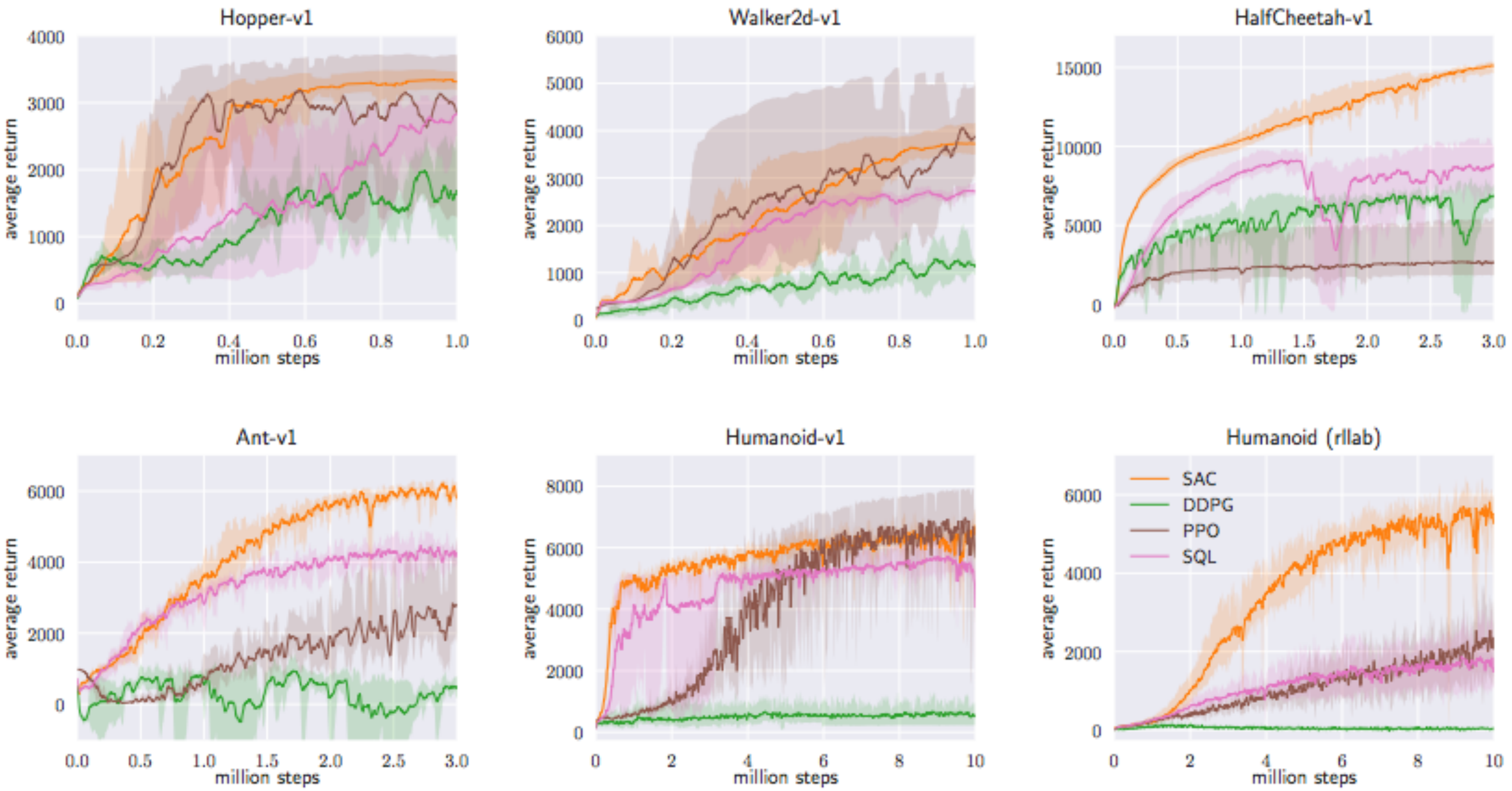




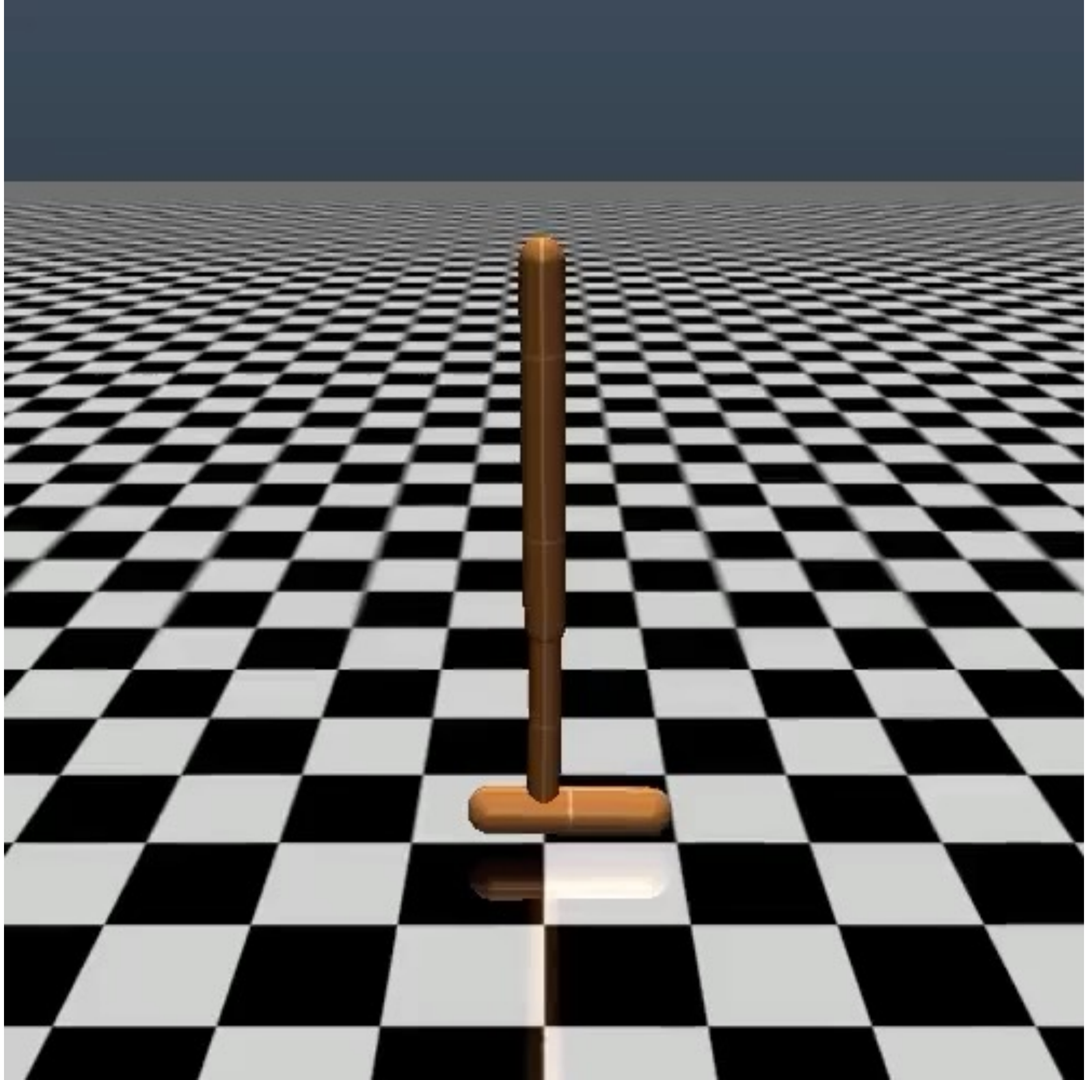
Algorithms:

Soft Actor-Critic (SAC)
Deep Deterministic Policy Gradient (DDPG)
Proximal Policy Optimization (PPO)
Soft Q-Learning (SQL)

sites.google.com/view/soft-actor-critic

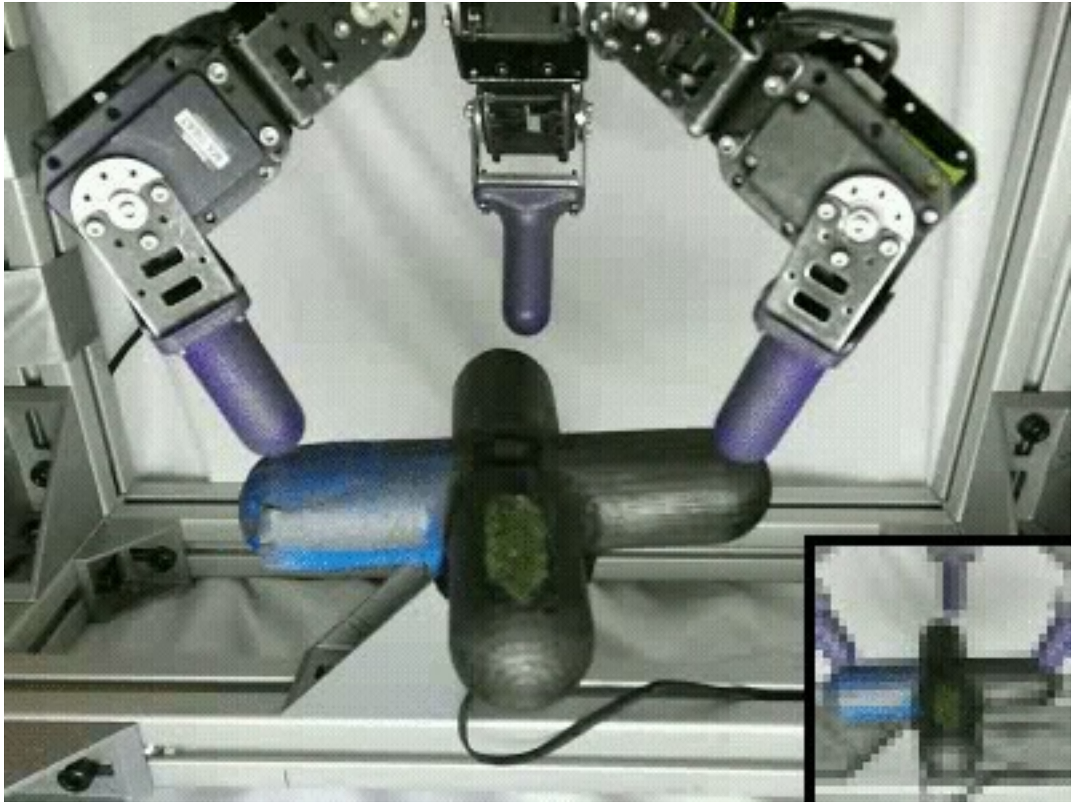


sites.google.com/view/soft-actor-critic

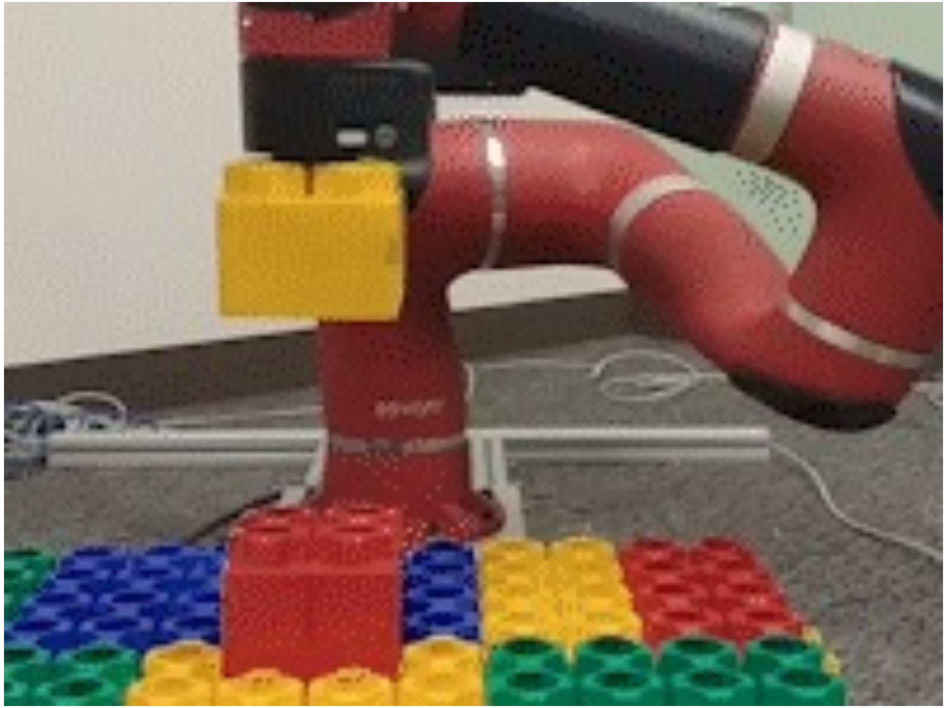


Real Robot Results

Real Robot Results



Real Robot Results



Summary of This Lecture

- Deep Deterministic Policy Gradient (DDPG)
- Soft Actor Critic (SAC)