Soft Actor-Critic

B. Ravindran

Soft Actor-Critic (SAC)

- v1 : "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", Haarnoja et al

- v2 : "Soft Actor-Critic: Algorithms and Applications", Haarnoja et al

Soft Actor-Critic (SAC)

- v1 : "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", Haarnoja et al

- v2 : "Soft Actor-Critic: Algorithms and Applications", Haarnoja et al

- Off-policy, model-free RL algorithm
- Uses stochastic policies
- Uses maximum entropy formulation to encourage stability and exploration

Soft Actor-Critic (SAC)

- v1 : "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", Haarnoja et al

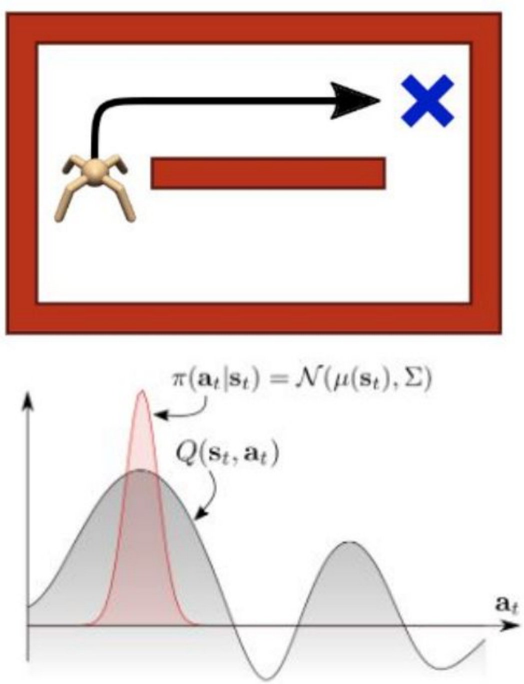
- v2 : "Soft Actor-Critic: Algorithms and Applications", Haarnoja et al

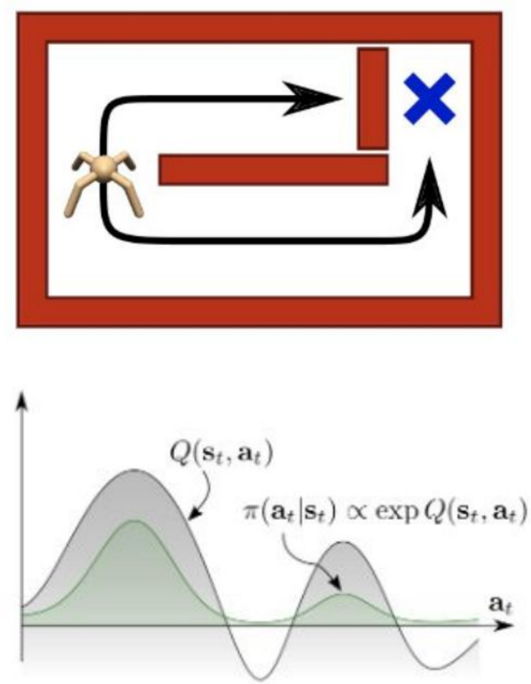
- Off-policy, model-free RL algorithm
- Uses stochastic policies
- Uses maximum entropy formulation to encourage stability and exploration

- Sample-efficient
- Scales to high-dimensional observation/action spaces
- Robust to random seeds, noise etc.

Maximum Entropy RL

- Maximize expected return while acting as randomly as possible
- Agent can capture different modes of optimality to improve robustness against environmental changes





Maximum Entropy?

- Given many distributions that satisfy certain constraints, pick the one that has maximum entropy.

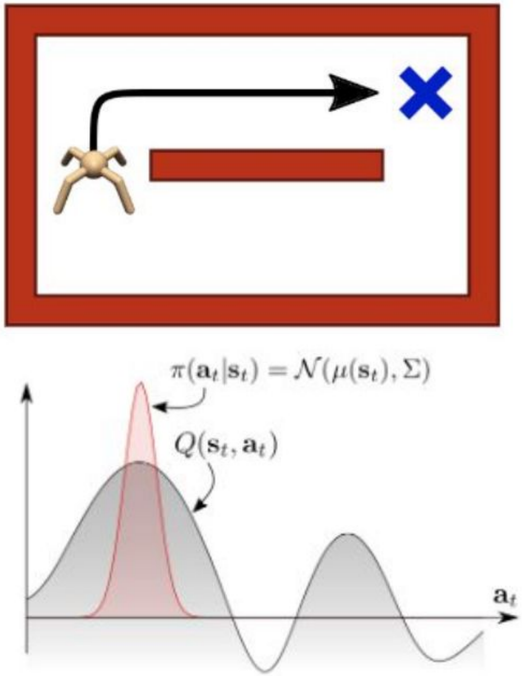
b \ a	0	1	p_b
0	p_{11}	p_{12}	
1	p_{21}	p_{22}	
p_a	0.6	0.4	1

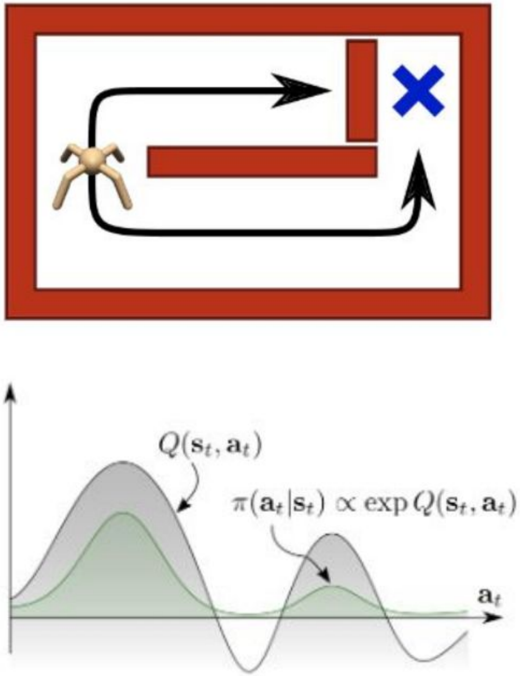
b \ a	0	1	p_b
0	0.5	0.1	
1	0.1	0.3	
p_a	0.6	0.4	1

b \ a	0	1	p_b
0	0.3	0.2	
1	0.3	0.2	
p_a	0.6	0.4	1

Maximum Entropy RL

- Maximize expected return while acting as randomly as possible
- Agent can capture different modes of optimality to improve robustness against environmental changes





Maximum Entropy RL

- Entropy of a random variable x

$$H(P) = \underset{x \sim P}{\mathrm{E}} \left[-\log P(x) \right].$$

- Maximum Entropy RL objective

$$\pi^* = \arg\max_{\pi} \underset{\tau \sim \pi}{\mathrm{E}} \left[\sum_{t=0}^{\infty} \gamma^t \Big(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right) \Big) \right]$$

- Here α > 0, is the weightage given to the entropy term in the objective. α is commonly referred to as the "temperature"

Maximum Entropy RL

- Define the value function to include the entropy bonuses from every timestep:

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t \bigg(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right) \bigg) \bigg| s_0 = s \right]$$

Maximum Entropy RL

- Define the value function to include the entropy bonuses from every timestep:

$$V^{\pi}(s) = \underset{\tau \sim \pi}{\mathrm{E}}\left[\sum_{t=0}^{\infty} \gamma^t \bigg(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right) \bigg) \bigg| s_0 = s \right]$$

- Define the action-value function to include the entropy bonuses from every timestep *except the first*:

$$Q^{\pi}(s,a) = \underset{\tau \sim \pi}{\mathrm{E}}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \sum_{t=1}^{\infty} \gamma^t H\left(\pi(\cdot|s_t)\right) \bigg| s_0 = s, a_0 = a \right]$$

Maximum Entropy RL

- Thus

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{a \sim \pi}\left[Q^{\pi}(s,a)\right] + \alpha H\left(\pi(\cdot|s)\right)$$

Maximum Entropy RL

- Thus

$$V^{\pi}(s) = \underset{a \sim \pi}{\mathrm{E}}\left[Q^{\pi}(s,a)\right] + \alpha H\left(\pi(\cdot|s)\right)$$

- Bellman equation

$$\begin{aligned} Q^{\pi}(s,a) &= \underset{\substack{s' \sim P \\ a' \sim \pi}}{\mathrm{E}}\left[R(s,a,s') + \gamma\left(Q^{\pi}(s',a') + \alpha H\left(\pi(\cdot|s')\right)\right)\right] \\ &= \underset{s' \sim P}{\mathrm{E}}\left[R(s,a,s') + \gamma V^{\pi}(s')\right]. \end{aligned}$$

SAC

- Policy π_θ
- Two Q functions Q_{ϕ_1}, Q_{ϕ_2}
- Two target Q functions
- SAC v1 : Temperature α is a hyperparameter
- SAC v2 : Temperature α is learnt

Learning the Q functions

- Both Q-functions are learned with MSBE minimization, by regressing to a single shared target.

$$L(\phi_i, \mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d)\sim\mathcal{D}} \left[\left(Q_{\phi_i}(s,a) - y(r,s',d) \right)^2 \right]$$

- The shared target is computed using target Q-networks, which are obtained by polyak averaging the Q-network parameters.
- The shared target makes use of the **clipped double-Q** trick.

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

- The next-state actions used in the target come from the **current policy** instead of a target policy.

$$\tilde{a}' \sim \pi(\cdot|s')$$

Learning the policy

- Maximize

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{a \sim \pi} [Q^{\pi}(s,a)] + \alpha H(\pi(\cdot|s))$$
$$= \mathop{\mathrm{E}}_{a \sim \pi} [Q^{\pi}(s,a) - \alpha \log \pi(a|s)].$$

- Policy is stochastic, therefore actions are sampled.
- To be able to backprop through sampled actions, we use the **reparameterization trick**
  - Policy outputs mean and variance of a Gaussian distribution
  - Add noise
  - Use tanh to squash to [-1,1]

$$\tilde{a}_{\theta}(s,\xi) = \tanh(\mu_{\theta}(s) + \sigma_{\theta}(s) \odot \xi), \qquad \xi \sim \mathcal{N}(0, I).$$

- Thus we can rewrite the expectation over actions into an expectation over noise,

$$\underset{a \sim \pi_\theta}{\mathrm{E}}\left[Q^{\pi_\theta}(s,a) - \alpha \log \pi_\theta(a|s)\right] = \underset{\xi \sim \mathcal{N}}{\mathrm{E}}\left[Q^{\pi_\theta}(s, \tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right]$$

- Thus we can rewrite the expectation over actions into an expectation over noise,

$$\underset{a\sim\pi_\theta}{\mathrm{E}}\left[Q^{\pi_\theta}(s,a)-\alpha\log\pi_\theta(a|s)\right]=\underset{\xi\sim\mathcal{N}}{\mathrm{E}}\left[Q^{\pi_\theta}(s,\tilde{a}_\theta(s,\xi))-\alpha\log\pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right]$$

- Thus the final objective becomes

$$\max_\theta \underset{\substack{s\sim\mathcal{D}\\ \xi\sim\mathcal{N}}}{\mathrm{E}}\left[\min_{j=1,2} Q_{\phi_j}(s,\tilde{a}_\theta(s,\xi))-\alpha\log\pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right]$$

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

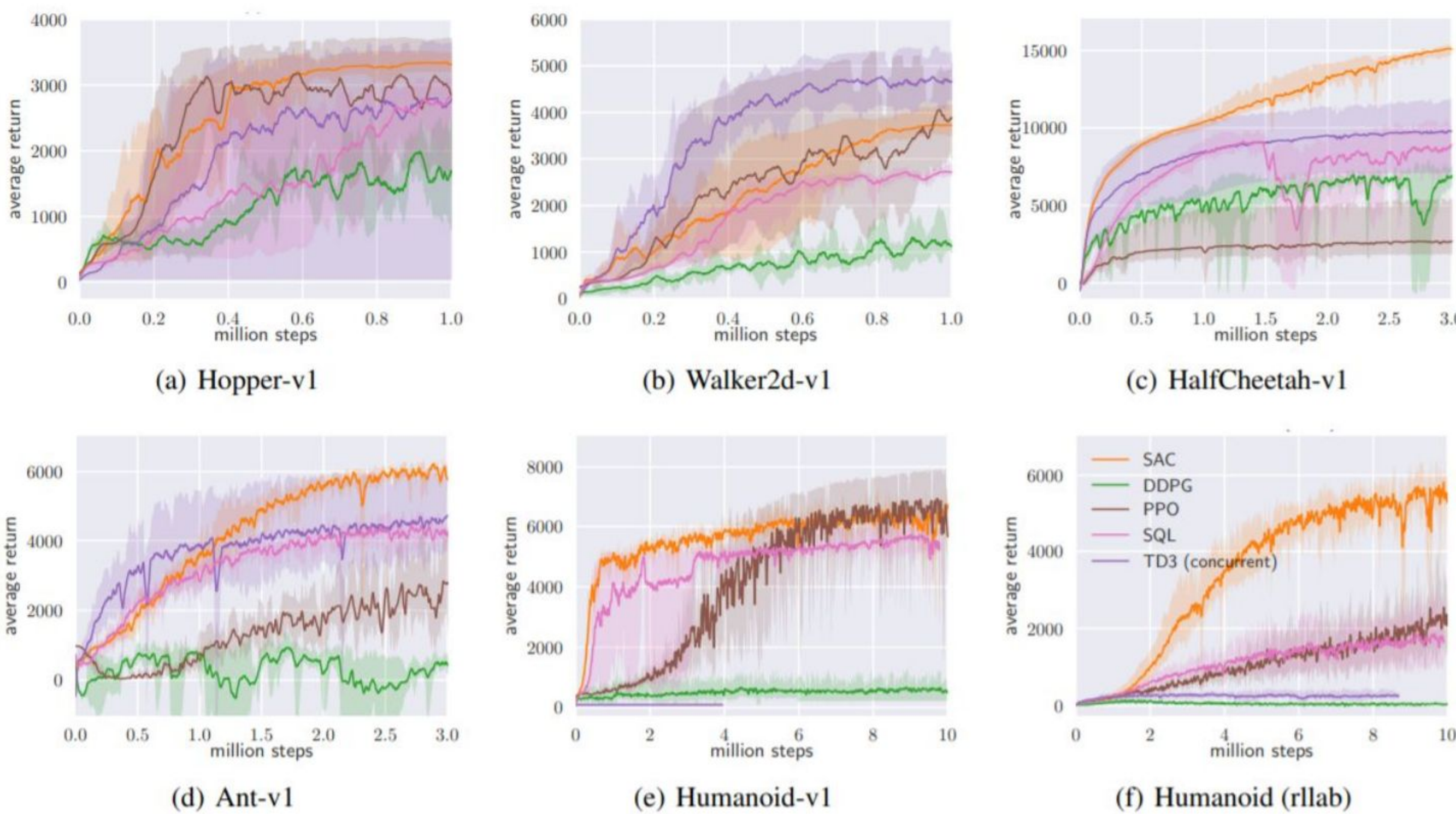
  end if

until convergence

SAC v1

- "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", Haarnoja et al
- Temperature α is a hyperparameter
- Tasks
  - A range of continuous control tasks from the OpenAI gym benchmark suite
  - The easier tasks can be solved by a wide range of different algorithms. The more complex benchmarks, such as the 21-dimensional Humanoid (rllab) are exceptionally difficult to solve with off-policy algorithms.
- Baselines:
  - DDPG, SQL, PPO, TD3 (concurrent)
  - TD3 is an extension to DDPG that first applied the double Q-learning trick to continuous control along with other improvements.

SAC v1 : Experimental Results



(a) Hopper-v1

(b) Walker2d-v1

(c) HalfCheetah-v1

(d) Ant-v1

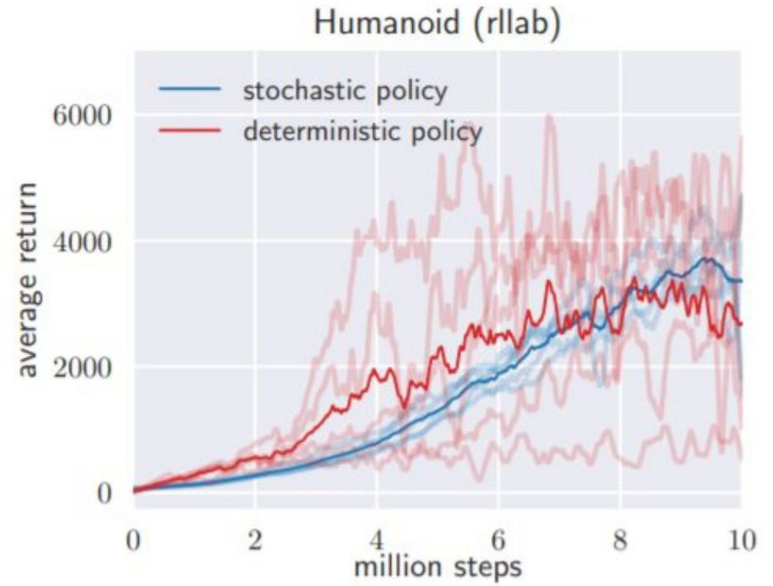
(e) Humanoid-v1

(f) Humanoid (rllab)

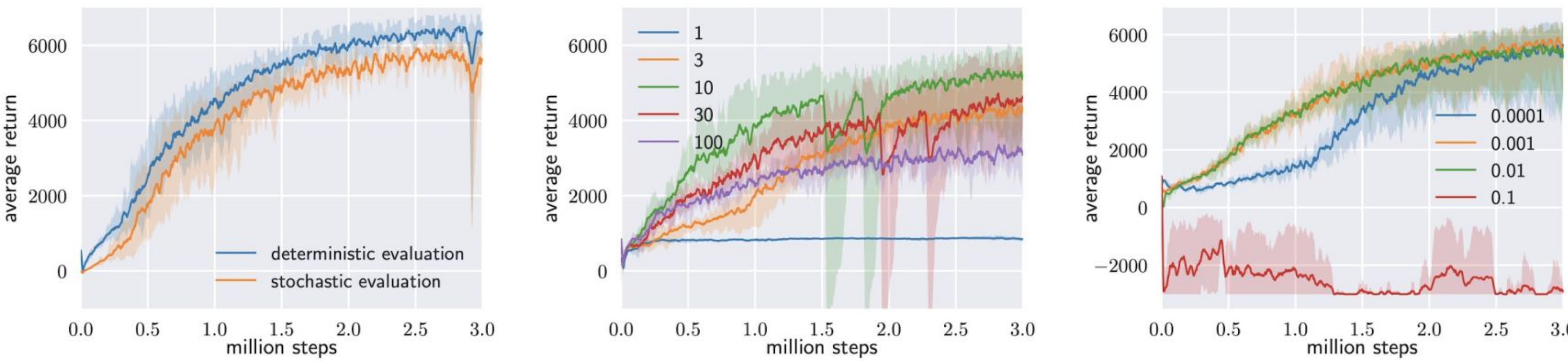
SAC v1 : Ablation Study

- How does the stochasticity of the policy and entropy maximization affect the performance?

- Comparison with a deterministic variant of SAC that does not maximize the entropy and that closely resembles DDPG



SAC v1 : Hyperparameter Sensitivity



(a) Evaluation (b) Reward Scale (c) Target Smoothing Coefficient (τ)

Limitation of SAC v1

- SAC v1 is brittle to the choice of hyperparameter α (temperature) that controls exploration
  - Solution --> Automatic temperature tuning!

SAC v2

- "Soft Actor-Critic: Algorithms and Applications", Haarnoja et al
- Temperature α is learnt
- Shows results on simulated tasks from OpenAI gym, RL Lab as well as real-world tasks such as locomotion for a quadrupedal robot and robotic manipulation with a dexterous hand

Experimental Results: RL Lab

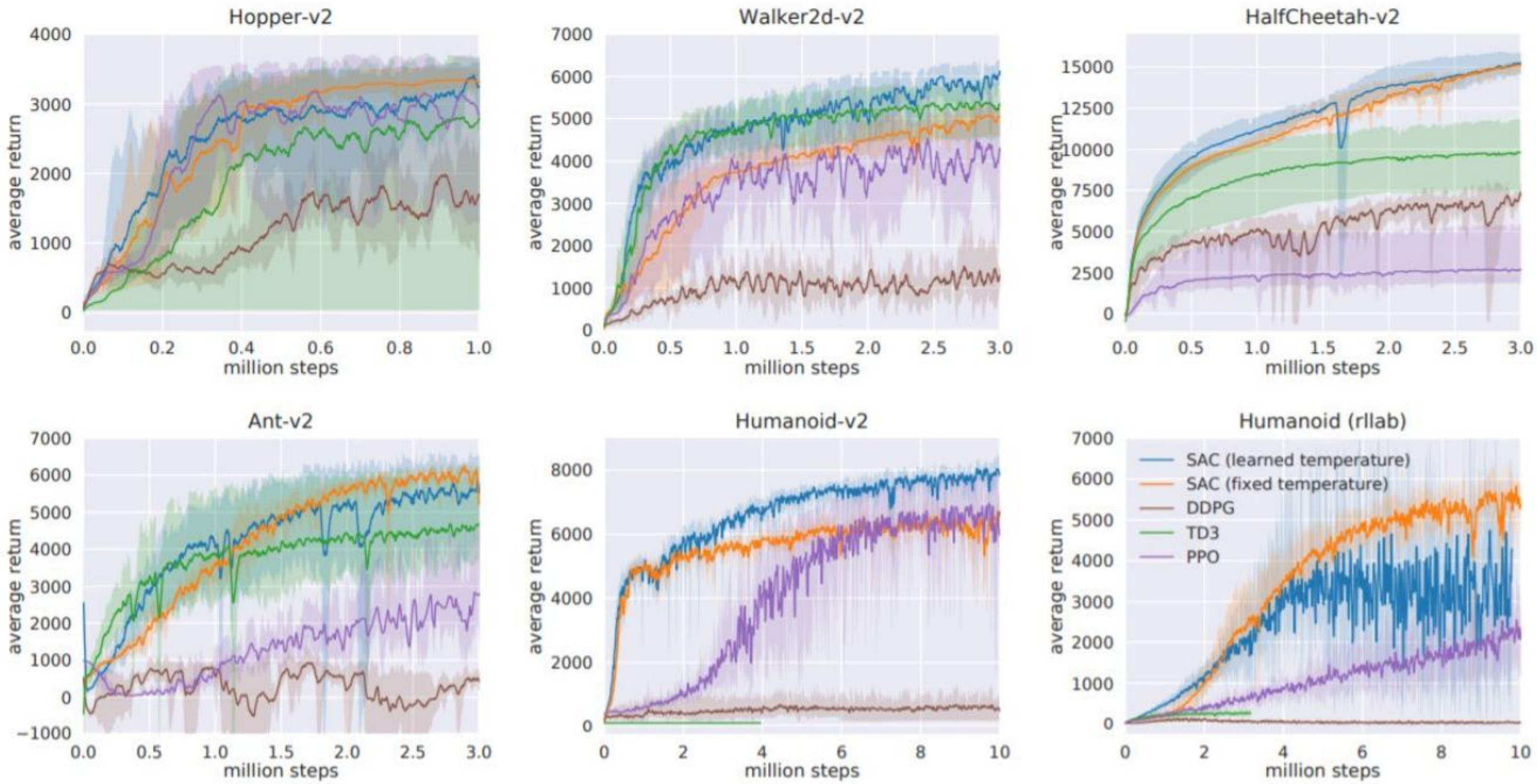
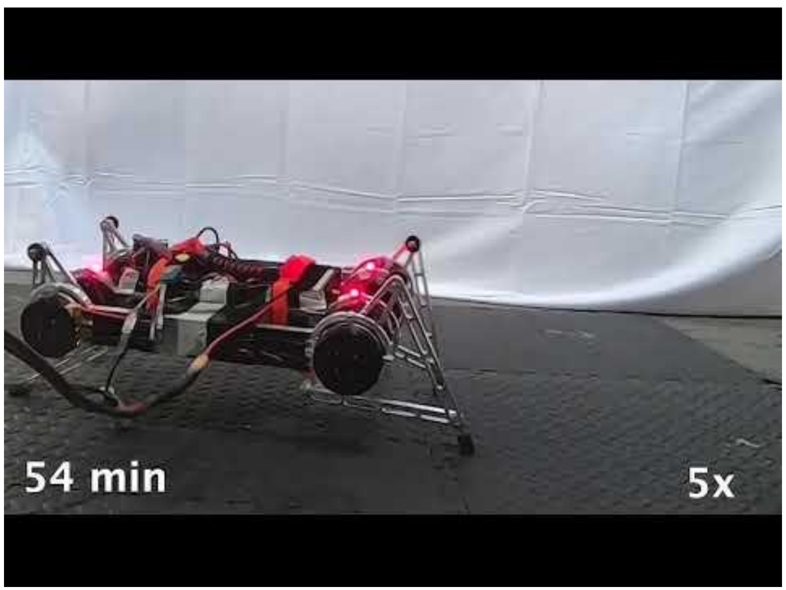
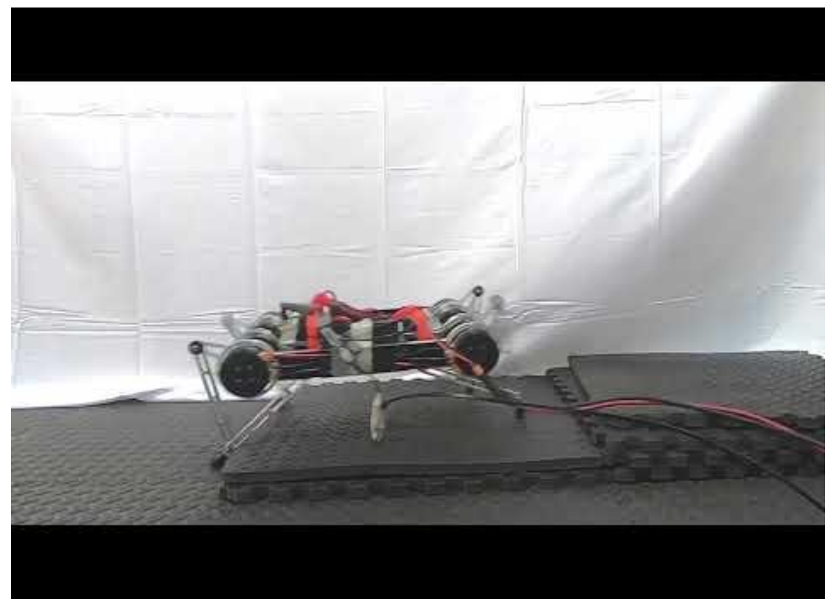


Figure 1: Training curves on continuous control benchmarks. Soft actor-critic (blue and yellow) performs consistently across all tasks and outperforming both on-policy and off-policy methods in the most challenging tasks.

Real World Robots

Quadrupedal Robot Locomotion



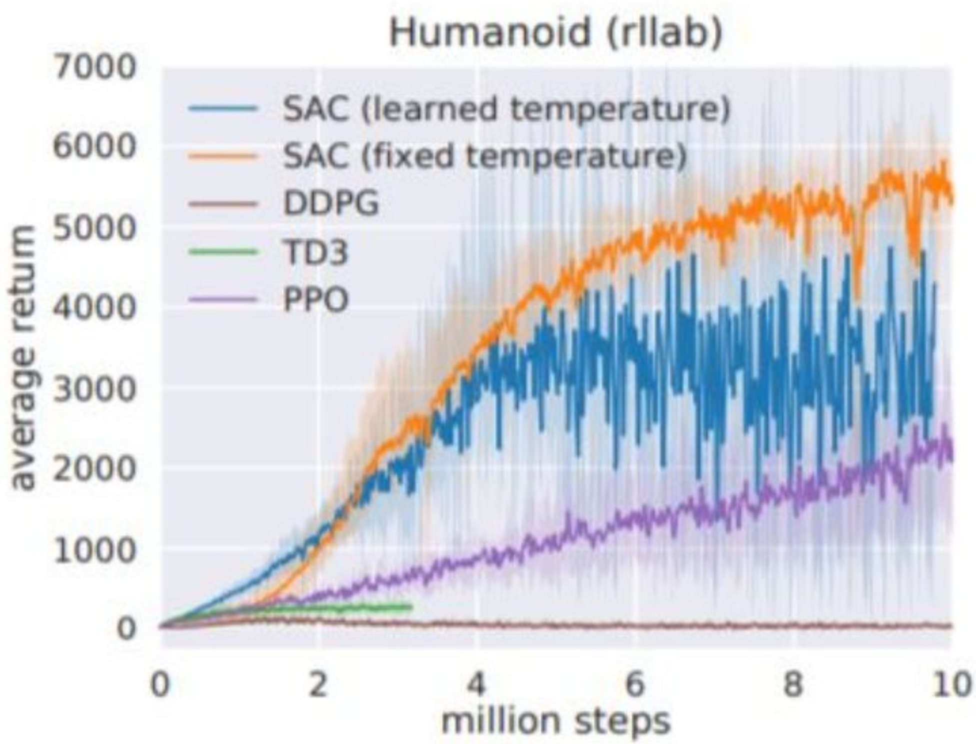


Real World Robots

- Dexterous hand manipulation
  - 20 hour end-to-end learning
  - Valve position as input: SAC 3 hours vs. PPO 7.4 hours

Limitations/Open Issues

- Lack of experiments on hard-exploration problems
- High-variance due to automatic temperature tuning



Recap: SAC

- An off-policy, model-free maximum entropy deep RL algorithm
  - Sample-efficient
  - Scales to high-dimensional observation/action spaces
  - Robust to random seeds, noise etc.
- SAC outperforms SOTA model-free deep RL methods, including DDPG, PPO in terms of the policy's optimality, sample complexity and robustness.

Thank you !!