Soft Actor-Critic

B. Ravindran

Soft Actor-Critic (SAC)

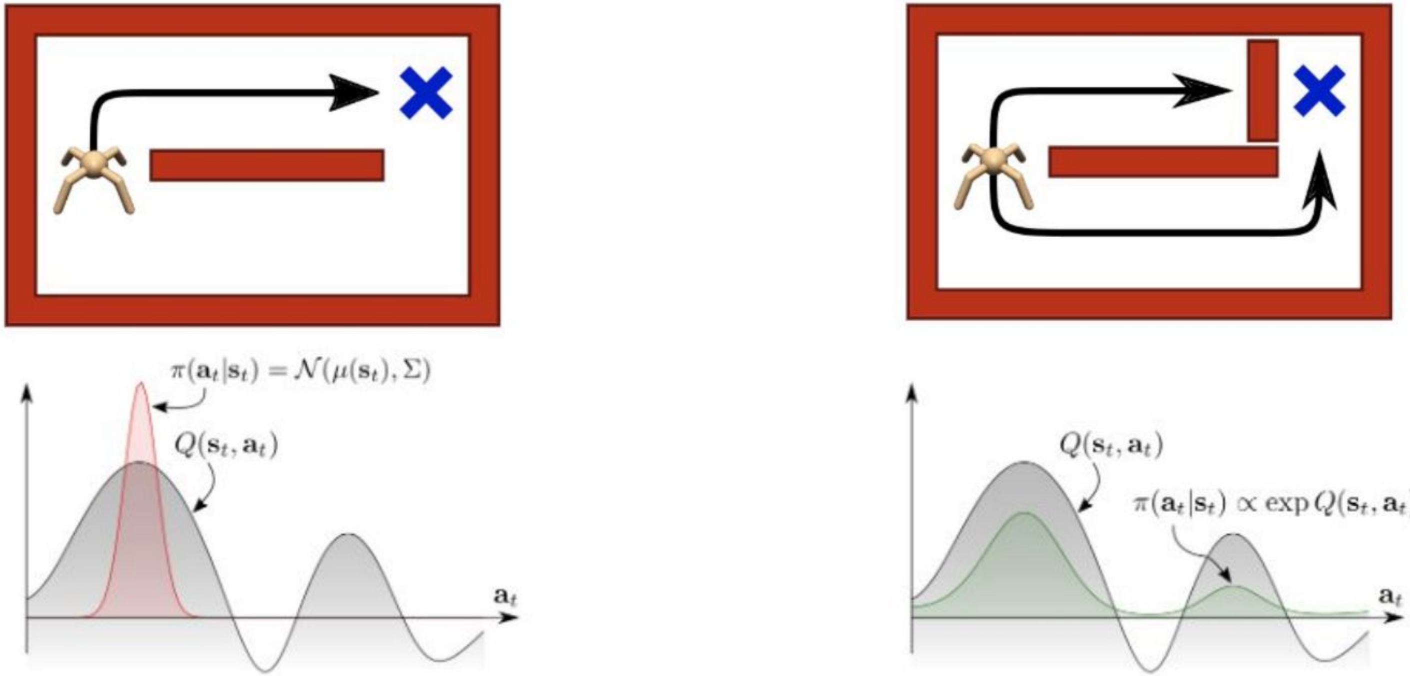
- Stochastic, off-policy, model-free RL algorithm
- Uses maximum entropy formulation to encourage stability and exploration
- Sample-efficient
- Scales to high-dimensional observation/action spaces
- Robust to random seeds, noise etc.
- State of the art!

- 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

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

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

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

Maximum Entropy RL

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

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

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

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

Maximum Entropy RL

- Thus

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

- Bellman equation

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

SAC

- Policy π_θ
- Two Q functions Q_{w_1} , Q_{w_2}
- Two target Q functions $Q_{w'_1}$, $Q_{w'_2}$
- SAC v1 : Temperature α is a hyperparameter
- SAC v2 : Temperature α is learnt

Learning the Q functions

- Both Q-functions are learned with Mean Squared Bellman Error minimization, by regressing to a single shared target y.

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

- The shared target y is computed using target Q-networks and makes use of the **clipped double-Q** trick.

$$y = r + \gamma \left(\min_{i=1,2} Q_{w_i'}(s', a') - \alpha \log \pi_\theta(a'|s') \right)$$

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

$$a' \sim \pi(\cdot|s')$$

Learning the policy

- Maximize

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

- Policy is stochastic, therefore actions are sampled.
- To be able to backprop through sampled actions, we use the **reparameterization trick**
  - Policy outputs mean μ and standard deviation σ of a Gaussian distribution
  - We then sample a gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbb{I})$
  - We combine the noise with the policy outputs and
  - Use tanh to squash the action to [-1,1]

$$
a = a_{\theta}(s, \epsilon) = \tanh(\mu_{\theta}(s) + \sigma_{\theta}(s) \cdot \epsilon)
$$

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

$$\mathop{\mathbb{E}}_{a \sim \pi_\theta} [\, Q^{\pi_\theta}(s, a) - \alpha \log \pi_\theta(a|s) \,] = \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}} [\, Q^{\pi_\theta}(s, a_\theta(s, \epsilon)) - \alpha \log \pi_\theta(a_\theta(s, \epsilon)|s) \,]$$

- Thus the final objective becomes

$$\max_\theta \mathop{\mathbb{E}}_{\substack{s \sim \mathcal{D} \\ \epsilon \sim \mathcal{N}}} [\, (\min_{i=1,2} Q_{w_i}(s, a_\theta(s, \epsilon)) - \alpha \log \pi_\theta(a_\theta(s, \epsilon)|s) \,]$$

Algorithm 1 SAC Algorithm

Initialize networks $\pi_\theta, Q_{w_1}, Q_{w_2}$ with random weights
Initialize target networks $Q_{w_1'} \leftarrow Q_{w_1}$, $Q_{w_2'} \leftarrow Q_{w_2}$
for Each episode **do**
  Reset environment and get initial state S
  for Each time step **do**
    Select action $A \sim \pi_\theta(S)$
    Get next state S', reward R
    Push (S, A, R, S') into replay buffer $\mathcal{D}$
    Sample mini batch $\mathcal{B} = \{(s, a, r, s')\}$ from replay buffer.
    Compute target for the Q functions

$$y = r + \gamma\ (\ \min_{i=1,2} Q_{w_i'}(s', a') - \alpha \log \pi_\theta(a'|s')\ ),\ a' \sim \pi(\cdot|s')$$

    Update Q functions by performing one step of gradient descent on the loss

$$L(w_i) = \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s') \in \mathcal{B}} [\ (Q_{w_i}(s, a) - y)^2\ ]$$

    Update policy by performing one step of gradient ascent on the objective

$$J(\theta) = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} [\ (\ \min_{i=1,2} Q_{w_i}(s, a_\theta(s, \epsilon)) - \alpha \log \pi_\theta(a_\theta(s, \epsilon)|s)\ ]$$

    where $a_\theta(s, \epsilon) = \tanh(\mu_\theta(s) + \sigma_\theta(s) \cdot \epsilon)$ and $\epsilon \sim \mathcal{N}(0, \mathbb{I})$
    Update target networks

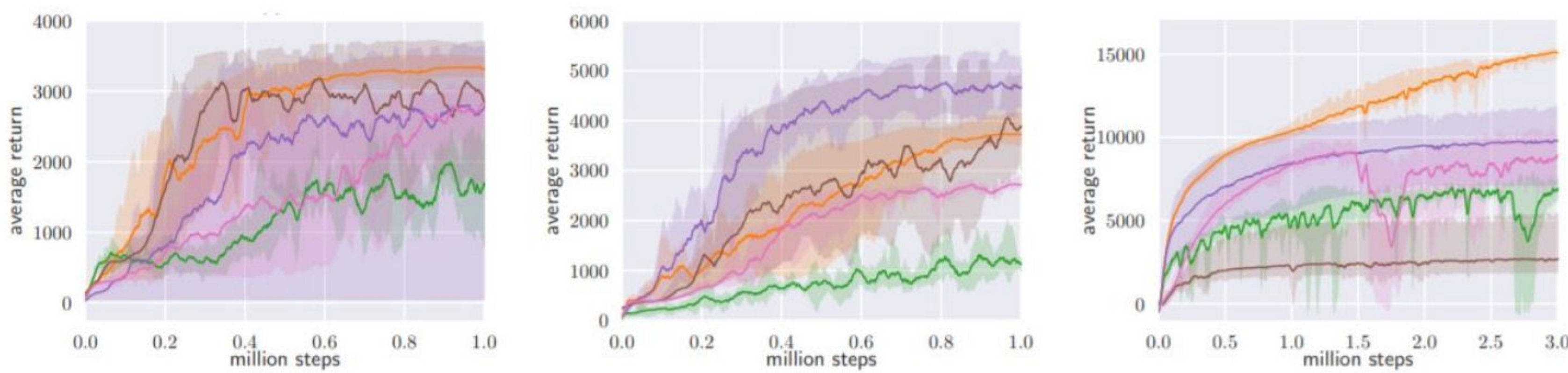
$$w_1' \leftarrow \tau w_1 + (1 - \tau) w_1'$$
$$w_2' \leftarrow \tau w_2 + (1 - \tau) w_2'$$

    $S \leftarrow S'$
  end for
end for

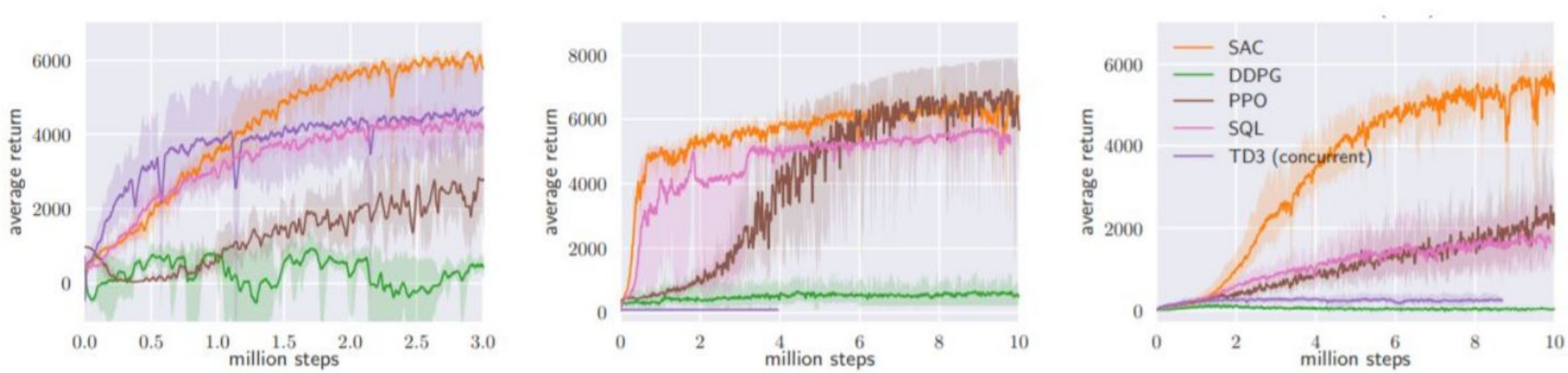
SAC v1

- "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
  - RL-Lab implementation of the Humanoid task
  - 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

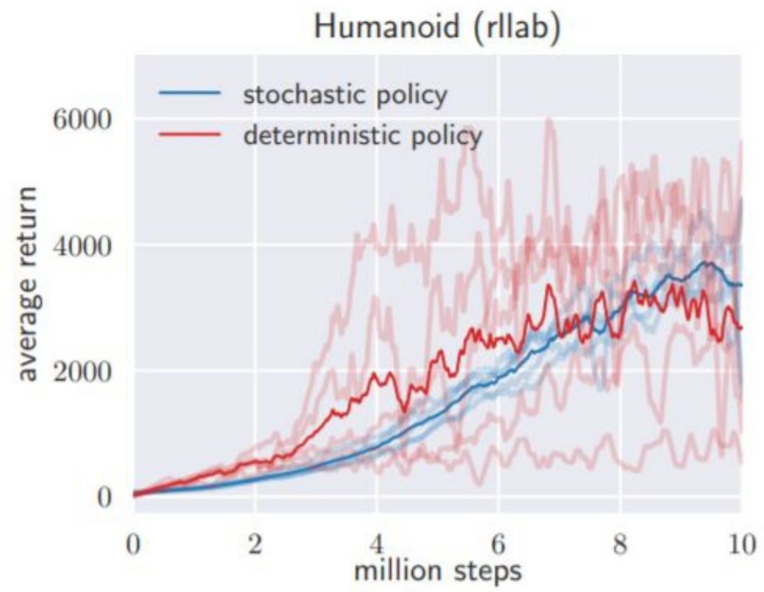


(d) Ant-v1 (e) Humanoid-v1 (f) Humanoid (rllab)

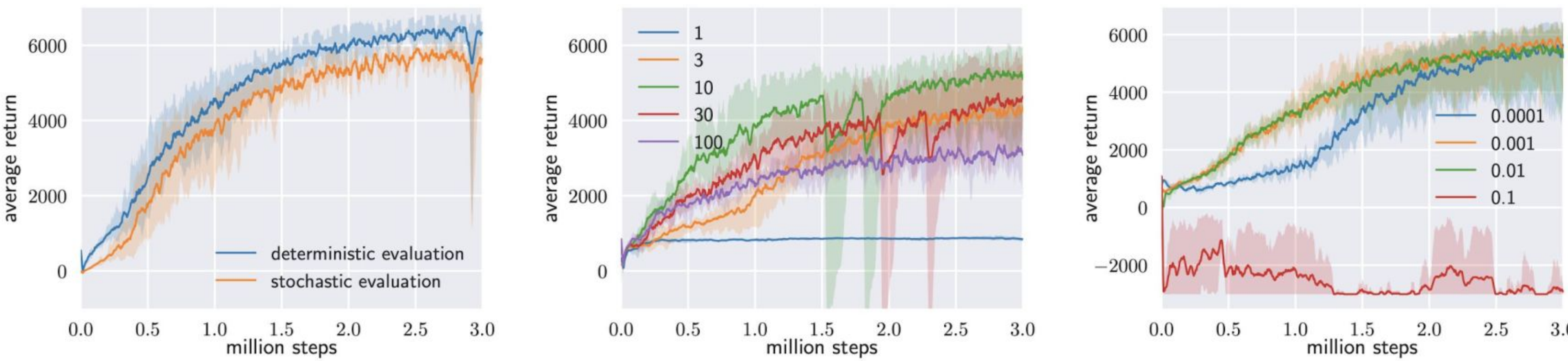
SAC v1 : Ablation Study

- 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 temperature α, a hyperparameter that controls exploration
    - Solution --> Automatic temperature tuning!

SAC v2

- "Soft Actor-Critic: Algorithms and Applications", Haarnoja et al
- Temperature α is learnt by minimizing the loss

$$L(\alpha) = \alpha\left(-\log \pi(a|s) - \widetilde{H}\right)$$

  where $\widetilde{H}$ is the entropy target.
- Typically, $\widetilde{H}$ it is set to be equal to the negative of the action space dimension i. e. $\widetilde{H} = -\dim(\mathcal{A})$
- SAC v2 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

SAC v2 : Experimental Results - RL Lab

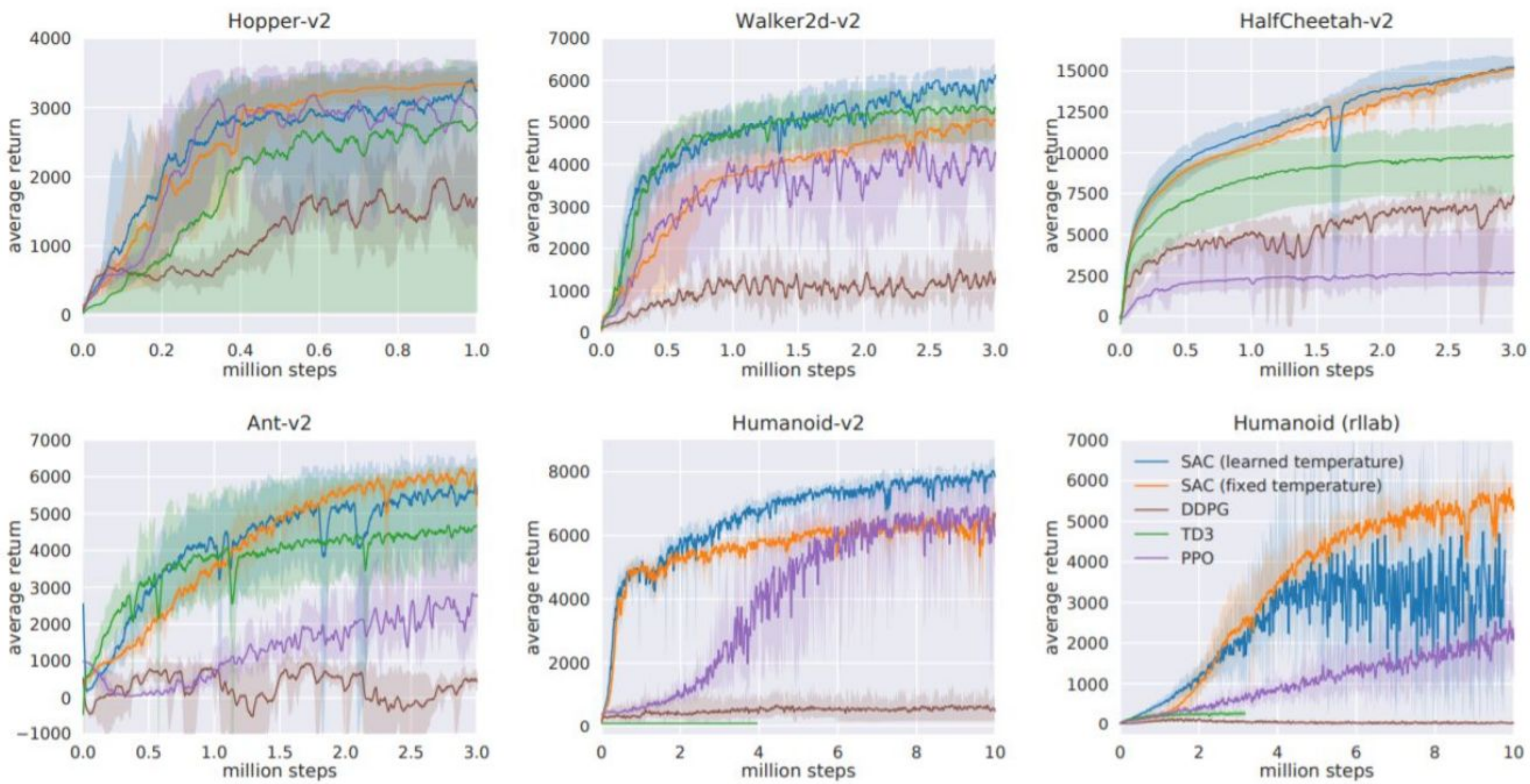
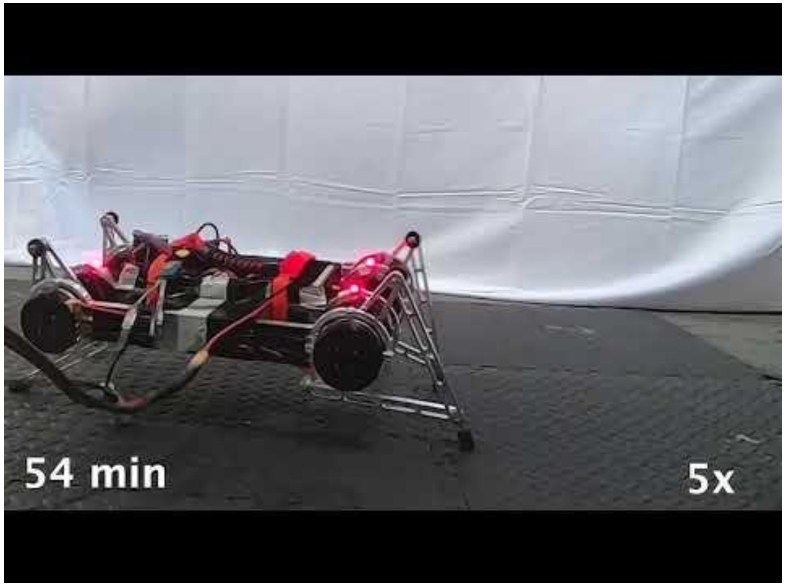


Figure 1: Training curves on continuous control benchmarks. Soft actor-critic (blue and yellow) performs consistently across all tasks and outperforming both on-policy and off-policy methods in the most challenging tasks.

SAC v2 :
Real World Robots

Quadrupedal Robot Locomotion

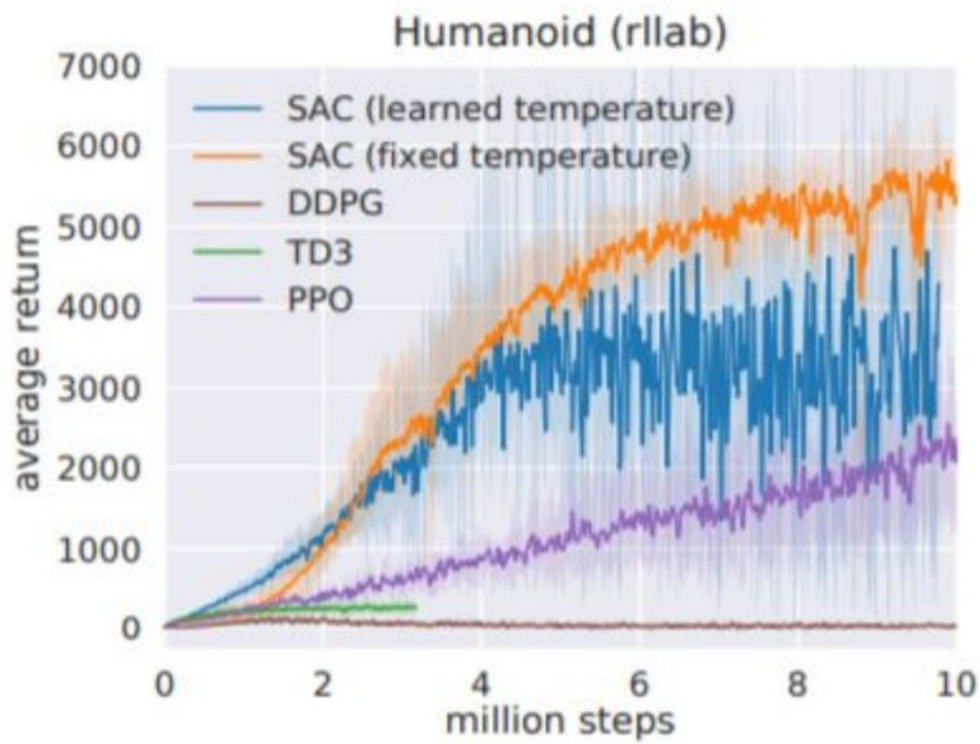


SAC v2 :
Real World Robots

- 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

- A stochastic, 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 average return, sample complexity and robustness.

Thank you !!