arXiv:1812.05905v2 [cs.LG] 29 Jan 2019

Soft Actor-Critic Algorithms and Applications

Tuomas Haarnoja[*†‡] **Aurick Zhou**[*†] **Kristian Hartikainen**[*†] **George Tucker**[‡]

Sehoon Ha[‡] **Jie Tan**[‡] **Vikash Kumar**[‡] **Henry Zhu**[†] **Abhishek Gupta**[†]

Pieter Abbeel[†] **Sergey Levine**[†‡]

Abstract

Model-free deep reinforcement learning (RL) algorithms have been successfully applied to a range of challenging sequential decision making and control tasks. However, these methods typically suffer from two major challenges: high sample complexity and brittleness to hyperparameters. Both of these challenges limit the applicability of such methods to real-world domains. In this paper, we describe Soft Actor-Critic (SAC), our recently introduced off-policy actor-critic algorithm based on the maximum entropy RL framework. In this framework, the actor aims to simultaneously maximize expected return and entropy; that is, to succeed at the task while acting as randomly as possible. We extend SAC to incorporate a number of modifications that accelerate training and improve stability with respect to the hyperparameters, including a constrained formulation that automatically tunes the temperature hyperparameter. We systematically evaluate SAC on a range of benchmark tasks, as well as challenging real-world tasks such as locomotion for a quadrupedal robot and robotic manipulation with a dexterous hand. With these improvements, SAC achieves state-of-the-art performance, outperforming prior on-policy and off-policy methods in sample-efficiency and asymptotic performance. Furthermore, we demonstrate that, in contrast to other off-policy algorithms, our approach is very stable, achieving similar performance across different random seeds. These results suggest that SAC is a promising candidate for learning in real-world robotics tasks.

1 Introduction

Model-free deep reinforcement learning (RL) algorithms have been applied in a range of challenging domains, from games (Mnih et al., 2013; Silver et al., 2016) to robotic control (Gu et al., 2017; Haarnoja et al., 2018b). The combination of RL and high-capacity function approximators such as neural networks holds the promise of automating a wide range of decision making and control tasks, but widespread adoption of these methods in real-world domains has been hampered by two major challenges. First, model-free deep RL methods are notoriously expensive in terms of their sample complexity. Even relatively simple tasks can require millions of steps of data collection, and complex behaviors with high-dimensional observations might need substantially more. Second, these methods are often brittle with respect to their hyperparameters: learning rates, exploration constants, and other settings must be set carefully for different problem settings to achieve good results. Both of these challenges severely limit the applicability of model-free deep RL to real-world tasks.

One cause for the poor sample efficiency of deep RL methods is on-policy learning: some of the most commonly used deep RL algorithms, such as TRPO (Schulman et al., 2015), PPO (Schulman et al.,

[†]UC Berkeley, [‡]Google Brain, [*]Contributed equally

2017b) or A3C (Mnih et al., 2016), require new samples to be collected for (nearly) every update to the policy. This quickly becomes extravagantly expensive, as the number of gradient steps and samples per step needed to learn an effective policy increases with task complexity. Off-policy algorithms aim to reuse past experience. This is not directly feasible with conventional policy gradient formulations, but is relatively straightforward for Q-learning based methods (Mnih et al., 2015). Unfortunately, the combination of off-policy learning and high-dimensional, nonlinear function approximation with neural networks presents a major challenge for stability and convergence (Bhatnagar et al., 2009). This challenge is further exacerbated in continuous state and action spaces, where a separate actor network is often used to perform the maximization in Q-learning.

In (Haarnoja et al., 2018c), we introduced the Soft Actor-Critic (SAC) algorithm based on the maximum entropy framework (Ziebart et al., 2008; Toussaint, 2009; Rawlik et al., 2012; Fox et al., 2016; Haarnoja et al., 2017). In the first sections of this paper, we summarize the SAC algorithm, describe the reasoning behind the design choices, and present key theoretical results from (Haarnoja et al., 2018c). Unfortunately, SAC as presented in (Haarnoja et al., 2018c) can suffer from brittleness to the temperature hyperparameter. Unlike in conventional reinforcement learning, where the optimal policy is independent of scaling of the reward function, in maximum entropy reinforcement learning the scaling factor has to be compensated by the choice a of suitable temperature, and a sub-optimal temperature can drastically degrade performance (Haarnoja et al., 2018c). To resolve this issue, we devise an automatic gradient-based temperature tuning method that adjusts the expected entropy over the visited states to match a target value. Although this modification is technically simple, we find that in practice it largely eliminates the need for per-task hyperparameter tuning. Finally, we present empirical results that show that Soft Actor-Critic attains a substantial improvement in both performance and sample efficiency over prior off-policy and on-policy methods including the recently introduced twin delayed deep deterministic (TD3) policy gradient algorithm (Fujimoto et al., 2018). We also evaluate our method on real-world challenging tasks such as locomotion for a quadrupedal robot and robotic manipulation with a dexterous hand from image observations.

2 Related Work

Maximum entropy reinforcement learning generalizes the expected return RL objective, although the original objective can be recovered in the zero temperature limit (Haarnoja et al., 2017). More importantly, the maximum entropy formulation provides a substantial improvement in exploration and robustness: as discussed by Ziebart (2010), maximum entropy policies are robust in the face of model and estimation errors, and as demonstrated by (Haarnoja et al., 2017), they improve exploration by acquiring diverse behaviors. Prior work has proposed model-free deep RL algorithms that perform on-policy learning with entropy maximization (O'Donoghue et al., 2016), as well as off-policy methods based on soft Q-learning and its variants (Schulman et al., 2017a; Nachum et al., 2017a; Haarnoja et al., 2017). However, the on-policy variants suffer from poor sample complexity for the reasons discussed above, while the off-policy variants require complex approximate inference procedures in continuous action spaces.

Our soft actor-critic algorithm incorporates three key ingredients: an actor-critic architecture with separate policy and value function networks, an off-policy formulation that enables reuse of previously collected data for efficiency, and entropy maximization to encourage stability and exploration. We review prior work that draw on some of these ideas in this section. Actor-critic algorithms are typically derived starting from policy iteration, which alternates between *policy evaluation*—computing the value function for a policy—and *policy improvement*—using the value function to obtain a better policy (Barto et al., 1983; Sutton & Barto, 1998). In large-scale reinforcement learning problems, it is typically impractical to run either of these steps to convergence, and instead the value function and policy are optimized jointly. In this case, the policy is referred to as the actor, and the value function as the critic. Many actor-critic algorithms build on the standard, on-policy policy gradient formulation to update the actor (Peters & Schaal, 2008), and many of them also consider the entropy of the policy, but instead of maximizing the entropy, they use it as an regularizer (Schulman et al., 2017b, 2015; Mnih et al., 2016; Gruslys et al., 2017). On-policy training tends to improve stability but results in poor sample complexity.

There have been efforts to increase the sample efficiency while retaining robustness by incorporating off-policy samples and by using higher order variance reduction techniques (O'Donoghue et al., 2016; Gu et al., 2016). However, fully off-policy algorithms still attain better efficiency. A particularly

popular off-policy actor-critic method, DDPG (Lillicrap et al., 2015), which is a deep variant of the deterministic policy gradient (Silver et al., 2014) algorithm, uses a Q-function estimator to enable off-policy learning, and a deterministic actor that maximizes this Q-function. As such, this method can be viewed both as a deterministic actor-critic algorithm and an approximate Q-learning algorithm. Unfortunately, the interplay between the deterministic actor network and the Q-function typically makes DDPG extremely difficult to stabilize and brittle to hyperparameter settings (Duan et al., 2016; Henderson et al., 2017). As a consequence, it is difficult to extend DDPG to complex, high-dimensional tasks, and on-policy policy gradient methods still tend to produce the best results in such settings (Gu et al., 2016). Our method instead combines off-policy actor-critic training with a *stochastic* actor, and further aims to maximize the entropy of this actor with an entropy maximization objective. We find that this actually results in a considerably more stable and scalable algorithm that, in practice, exceeds both the efficiency and final performance of DDPG. A similar method can be derived as a zero-step special case of stochastic value gradients (SVG(0)) (Heess et al., 2015). However, SVG(0) differs from our method in that it optimizes the standard maximum expected return objective.

New:
- stochastic actor
- max. entropy of this actor

Maximum entropy reinforcement learning optimizes policies to maximize both the expected return and the expected entropy of the policy. This framework has been used in many contexts, from inverse reinforcement learning (Ziebart et al., 2008) to optimal control (Todorov, 2008; Toussaint, 2009; Rawlik et al., 2012). Maximum a posteriori policy optimization (MPO) makes use of the probablistic view and optimizes the standard RL objective via expectation maximization (Abdolmaleki et al., 2018). In guided policy search (Levine & Koltun, 2013; Levine et al., 2016), the maximum entropy distribution is used to guide policy learning towards high-reward regions. More recently, several papers have noted the connection between Q-learning and policy gradient methods in the framework of maximum entropy learning (O'Donoghue et al., 2016; Haarnoja et al., 2017; Nachum et al., 2017a; Schulman et al., 2017a). While most of the prior model-free works assume a discrete action space, Nachum et al. (2017b) approximate the maximum entropy distribution with a Gaussian, and Haarnoja et al. (2017) with a sampling network trained to draw samples from the optimal policy. Although the soft Q-learning algorithm proposed by Haarnoja et al. (2017) has a value function and actor network, it is not a true actor-critic algorithm: the Q-function is estimating the optimal Q-function, and the actor does not directly affect the Q-function except through the data distribution. Hence, Haarnoja et al. (2017) motivates the actor network as an approximate sampler, rather than the actor in an actor-critic algorithm. Crucially, the convergence of this method hinges on how well this sampler approximates the true posterior. In contrast, we prove that our method converges to the optimal policy from a given policy class, regardless of the policy parameterization. Furthermore, these prior maximum entropy methods generally do not exceed the performance of state-of-the-art off-policy algorithms, such as TD3 (Fujimoto et al., 2018) or MPO (Abdolmaleki et al., 2018), when learning from scratch, though they may have other benefits, such as improved exploration and ease of fine-tuning.

3 Preliminaries

We first introduce notation and summarize the standard and maximum entropy reinforcement learning frameworks.

3.1 Notation

We will address learning of maximum entropy policies in continuous action spaces. Our reinforcement learning problem can be defined as policy search in an a Markov decision process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, p, r)$. The state space $\mathcal{S}$ and action space $\mathcal{A}$ are assumed to be continuous, and the state transition probability $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ represents the probability density of the next state $\mathbf{s}_{t+1} \in \mathcal{S}$ given the current state $\mathbf{s}_t \in \mathcal{S}$ and action $\mathbf{a}_t \in \mathcal{A}$. The environment emits a reward $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ on each transition. We will also use $\rho_\pi(\mathbf{s}_t)$ and $\rho_\pi(\mathbf{s}_t, \mathbf{a}_t)$ to denote the state and state-action marginals of the trajectory distribution induced by a policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$.

3.2 Maximum Entropy Reinforcement Learning

The standard reinforcement learning objective is the expected sum of rewards $\sum_t \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} [r(\mathbf{s}_t, \mathbf{a}_t)]$ and our goal is to learn a policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$ that maximizes that ob-

jective. The maximum entropy objective (see e.g. (Ziebart, 2010) generalizes the standard objective by augmenting it with an entropy term, such that the optimal policy additionally aims to maximize its entropy at each visited state:

$$\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t)) \right], \tag{1}$$

where α is the temperature parameter that determines the relative importance of the entropy term versus the reward, and thus controls the stochasticity of the optimal policy. Although the maximum entropy objective differs from the standard maximum expected return objective used in conventional reinforcement learning, the conventional objective can be recovered in the limit as $\alpha \to 0$. If we wish to extend either the conventional or the maximum entropy RL objective to infinite horizon problems, it is convenient to also introduce a discount factor γ to ensure that the sum of expected rewards (and entropies) is finite. In the context of policy search algorithms, the use of a discount factor is actually a somewhat nuanced choice, and writing down the precise objective that is optimized when using the discount factor is non-trivial (Thomas, 2014). We include the discounted, infinite-horizon objective in Appendix A, but we will use the discount γ in the following derivations and in our final algorithm.

The maximum entropy objective has a number of conceptual and practical advantages. First, the policy is incentivized to explore more widely, while giving up on clearly unpromising avenues. Second, the policy can capture multiple modes of near-optimal behavior. In problem settings where multiple actions seem equally attractive, the policy will commit equal probability mass to those actions. In practice, we observe improved exploration with this objective, as also has been reported in the prior work (Schulman et al., 2017a), and we observe that it considerably improves learning speed over state-of-art methods that optimize the conventional RL objective function.

Optimization problems of this type have been explored in a number of prior works (Kappen, 2005; Todorov, 2007; Ziebart et al., 2008). These prior methods have proposed directly solving for the optimal Q-function, from which the optimal policy can be recovered (Ziebart et al., 2008; Fox et al., 2016; Haarnoja et al., 2017). In the following section, we discuss how we can devise a soft actor-critic algorithm through a policy iteration formulation, where we instead evaluate the Q-function of the current policy and update the policy through an *off-policy* gradient update. Though such algorithms have previously been proposed for conventional reinforcement learning, our method is, to our knowledge, the first off-policy actor-critic method in the maximum entropy reinforcement learning framework.

4 From Soft Policy Iteration to Soft Actor-Critic

Our off-policy soft actor-critic algorithm can be derived starting from a maximum entropy variant of the policy iteration method. We will first present this derivation, verify that the corresponding algorithm converges to the optimal policy from its density class, and then present a practical deep reinforcement learning algorithm based on this theory. In this section, we treat the temperature as a constant, and later in Section 5 propose an extension to SAC that adjusts the temperature automatically to match an entropy target in expectation.

4.1 Soft Policy Iteration

We will begin by deriving soft policy iteration, a general algorithm for learning optimal maximum entropy policies that alternates between policy evaluation and policy improvement in the maximum entropy framework. Our derivation is based on a tabular setting, to enable theoretical analysis and convergence guarantees, and we extend this method into the general continuous setting in the next section. We will show that soft policy iteration converges to the optimal policy within a set of policies which might correspond, for instance, to a set of parameterized densities.

In the policy evaluation step of soft policy iteration, we wish to compute the value of a policy π according to the maximum entropy objective. For a fixed policy, the soft Q-value can be computed iteratively, starting from any function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and repeatedly applying a modified Bellman backup operator $\mathcal{T}^\pi$ given by

$$\mathcal{T}^\pi Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p} \left[V(\mathbf{s}_{t+1}) \right], \tag{2}$$

where

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi}\left[Q(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi(\mathbf{a}_t|\mathbf{s}_t)\right] \tag{3}$$

is the soft state value function. We can obtain the soft Q-function for any policy π by repeatedly applying $\mathcal{T}^\pi$ as formalized below.

Lemma 1 (Soft Policy Evaluation)**.** *Consider the soft Bellman backup operator $\mathcal{T}^\pi$ in Equation 2 and a mapping $Q^0 : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q^{k+1} = \mathcal{T}^\pi Q^k$. Then the sequence Q^k will converge to the soft Q-function of π as $k \rightarrow \infty$.*

Proof. See Appendix B.1. □

In the policy improvement step, we update the policy towards the exponential of the new soft Q-function. This particular choice of update can be guaranteed to result in an improved policy in terms of its soft value. Since in practice we prefer policies that are tractable, we will additionally restrict the policy to some set of policies Π, which can correspond, for example, to a parameterized family of distributions such as Gaussians. To account for the constraint that $\pi \in \Pi$, we project the improved policy into the desired set of policies. While in principle we could choose any projection, it will turn out to be convenient to use the information projection defined in terms of the Kullback-Leibler divergence. In the other words, in the policy improvement step, for each state, we update the policy according to

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(\mathbf{s}_t, \,\cdot\,)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)}\right). \tag{4}$$

The partition function $Z^{\pi_{\text{old}}}(\mathbf{s}_t)$ normalizes the distribution, and while it is intractable in general, it does not contribute to the gradient with respect to the new policy and can thus be ignored. For this projection, we can show that the new, projected policy has a higher value than the old policy with respect to the maximum entropy objective. We formalize this result in Lemma 2.

Lemma 2 (Soft Policy Improvement)**.** *Let $\pi_{\text{old}} \in \Pi$ and let π_{new} be the optimizer of the minimization problem defined in Equation 4. Then $Q^{\pi_{\text{new}}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$.*

Proof. See Appendix B.2. □

The full soft policy iteration algorithm alternates between the soft policy evaluation and the soft policy improvement steps, and it will provably converge to the optimal maximum entropy policy among the policies in Π (Theorem 1). Although this algorithm will provably find the optimal solution, we can perform it in its exact form only in the tabular case. Therefore, we will next approximate the algorithm for continuous domains, where we need to rely on a function approximator to represent the Q-values, and running the two steps until convergence would be computationally too expensive. The approximation gives rise to a new practical algorithm, called soft actor-critic.

Theorem 1 (Soft Policy Iteration)**.** *Repeated application of soft policy evaluation and soft policy improvement from any $\pi \in \Pi$ converges to a policy π^* such that $Q^{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ for all $\pi \in \Pi$ and $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$, assuming $|\mathcal{A}| < \infty$.*

Proof. See Appendix B.3. □

4.2 Soft Actor-Critic

As discussed above, large continuous domains require us to derive a practical approximation to soft policy iteration. To that end, we will use function approximators for both the soft Q-function and the policy, and instead of running evaluation and improvement to convergence, alternate between optimizing both networks with stochastic gradient descent. We will consider a parameterized soft Q-function $Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$ and a tractable policy $\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$. The parameters of these networks are θ and ϕ. For example, the soft Q-function can be modeled as expressive neural networks, and the policy as a Gaussian with mean and covariance given by neural networks. We will next derive update rules for these parameter vectors.

The soft Q-function parameters can be trained to minimize the soft Bellman residual

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - \left(r(\mathbf{s}_t,\mathbf{a}_t) + \gamma\,\mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V_{\bar{\theta}}(\mathbf{s}_{t+1})\right]\right)\right)^2\right], \tag{5}$$

where the value function is implicitly parameterized through the soft Q-function parameters via Equation 3, and it can be optimized with stochastic gradients[1]

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t,\mathbf{s}_t)\left(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - \left(r(\mathbf{s}_t,\mathbf{a}_t) + \gamma\left(Q_{\bar{\theta}}(\mathbf{s}_{t+1},\mathbf{a}_{t+1}) - \alpha\log\left(\pi_\phi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right)\right)\right)\right). \tag{6}$$

The update makes use of a target soft Q-function with parameters $\bar{\theta}$ that are obtained as an exponentially moving average of the soft Q-function weights, which has been shown to stabilize training (Mnih et al., 2015). Finally, the policy parameters can be learned by directly minimizing the expected KL-divergence in Equation 4 (multiplied by α and ignoring the constant log-partition function and by α):

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{a}_t\sim\pi_\phi}\left[\alpha\log\left(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right) - Q_\theta(\mathbf{s}_t,\mathbf{s}_t)\right]\right] \tag{7}$$

There are several options for minimizing J_π. A typical solution for policy gradient methods is to use the likelihood ratio gradient estimator (Williams, 1992), which does not require backpropagating the gradient through the policy and the target density networks. However, in our case, the target density is the Q-function, which is represented by a neural network an can be differentiated, and it is thus convenient to apply the reparameterization trick instead, resulting in a lower variance estimator. To that end, we reparameterize the policy using a neural network transformation

$$\mathbf{a}_t = f_\phi(\epsilon_t;\mathbf{s}_t), \tag{8}$$

where ϵ_t is an input noise vector, sampled from some fixed distribution, such as a spherical Gaussian. We can now rewrite the objective in Equation 7 as

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}\left[\alpha\log\pi_\phi(f_\phi(\epsilon_t;\mathbf{s}_t)|\mathbf{s}_t) - Q_\theta(\mathbf{s}_t,f_\phi(\epsilon_t;\mathbf{s}_t))\right], \tag{9}$$

where π_ϕ is defined implicitly in terms of f_ϕ. We can approximate the gradient of Equation 9 with

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi\alpha\log\left(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right) + \left(\nabla_{\mathbf{a}_t}\alpha\log\left(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right) - \nabla_{\mathbf{a}_t}Q(\mathbf{s}_t,\mathbf{a}_t)\right)\nabla_\phi f_\phi(\epsilon_t;\mathbf{s}_t), \tag{10}$$

where $\mathbf{a}_t$ is evaluated at $f_\phi(\epsilon_t;\mathbf{s}_t)$. This unbiased gradient estimator extends the DDPG style policy gradients (Lillicrap et al., 2015) to any tractable stochastic policy.

5 Automating Entropy Adjustment for Maximum Entropy RL

In the previous section, we derived a practical off-policy algorithm for learning maximum entropy policies of a given temperature. Unfortunately, choosing the optimal temperature is non-trivial, and the temperature needs to be tuned for each task. Instead of requiring the user to set the temperature manually, we can automate this process by formulating a different maximum entropy reinforcement learning objective, where the entropy is treated as a constraint. The magnitude of the reward differs not only across tasks, but it also depends on the policy, which improves over time during training. Since the optimal entropy depends on this magnitude, this makes the temperature adjustment particularly difficult: the entropy can vary unpredictably both across tasks and during training as the policy becomes better. Simply forcing the entropy to a fixed value is a poor solution, since the policy should be free to explore more in regions where the optimal action is uncertain, but remain more deterministic in states with a clear distinction between good and bad actions. Instead, we formulate a constrained optimization problem where the average entropy of the policy is constrained, while the entropy at different states can vary. Similar approach was taken in (Abdolmaleki et al., 2018), where the policy was constrained to remain close to the previous policy. We show that the dual to this constrained optimization leads to the soft actor-critic updates, along with an additional update for the dual variable, which plays the role of the temperature. Our formulation also makes it possible to learn the entropy with more expressive policies that can model multi-modal distributions, such as policies based on normalizing flows (Haarnoja et al., 2018a) for which no closed form expression for

[1] In (Haarnoja et al., 2018c) we introduced an additional function approximator for the value function, but later found it to be unnecessary.

the entropy exists. We will derive the update for finite horizon case, and then derive an approximation for stationary policies by dropping the time dependencies from the policy, soft Q-function, and the temperature.

Our aim is to find a stochastic policy with maximal expected return that satisfies a minimum expected entropy constraint. Formally, we want to solve the constrained optimization problem

$$\max_{\pi_{0:T}} \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right] \text{ s.t. } \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[-\log(\pi_t(\mathbf{a}_t|\mathbf{s}_t))\right] \geq \mathcal{H} \;\; \forall t \tag{11}$$

where $\mathcal{H}$ is a desired minimum expected entropy. Note that, for fully observed MDPs, the policy that optimizes the expected return is deterministic, so we expect this constraint to usually be tight and do not need to impose an upper bound on the entropy.

Since the policy at time t can only affect the future objective value, we can employ an (approximate) dynamic programming approach, solving for the policy backward through time. We rewrite the objective as an iterated maximization

$$\max_{\pi_0}\left(\mathbb{E}\left[r(\mathbf{s}_0, \mathbf{a}_0)\right] + \max_{\pi_1}\left(\mathbb{E}\left[\ldots\right] + \max_{\pi_T}\mathbb{E}\left[r(\mathbf{s}_T, \mathbf{a}_T)\right]\right)\right), \tag{12}$$

subject to the constraint on entropy. Starting from the last time step, we change the constrained maximization to the dual problem. Subject to $\mathbb{E}_{(\mathbf{s}_T,\mathbf{a}_T)\sim\rho_\pi}\left[-\log(\pi_T(\mathbf{s}_T|\mathbf{s}_T))\right] \geq \mathcal{H}$,

$$\max_{\pi_T}\mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_T, \mathbf{a}_T)\right] = \min_{\alpha_T\geq 0}\max_{\pi_T}\mathbb{E}\left[r(\mathbf{s}_T, \mathbf{a}_T) - \alpha_T\log\pi(\mathbf{a}_T|\mathbf{s}_T)\right] - \alpha_T\mathcal{H}, \tag{13}$$

where α_T is the dual variable. We have also used strong duality, which holds since the objective is linear and the constraint (entropy) is convex function in π_T. This dual objective is closely related to the maximum entropy objective with respect to the policy, and the optimal policy is the maximum entropy policy corresponding to temperature α_T: $\pi_T^*(\mathbf{a}_T|\mathbf{s}_T;\alpha_T)$. We can solve for the optimal dual variable α_T^* as

$$\arg\min_{\alpha_T}\mathbb{E}_{\mathbf{s}_t,\mathbf{a}_t\sim\pi_t^*}\left[-\alpha_T\log\pi_T^*(\mathbf{a}_T|\mathbf{s}_T;\alpha_T) - \alpha_T\mathcal{H}\right]. \tag{14}$$

To simplify notation, we make use of the recursive definition of the soft Q-function

$$Q_t^*(\mathbf{s}_t, \mathbf{a}_t; \pi_{t+1:T}^*, \alpha_{t+1:T}^*) = \mathbb{E}\left[r(\mathbf{s}_t, \mathbf{a}_t)\right] + \mathbb{E}_{\rho_\pi}\left[Q_{t+1}^*(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \alpha_{t+1}^*\log\pi_{t+1}^*(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right], \tag{15}$$

with $Q_T^*(\mathbf{s}_T, \mathbf{a}_T) = \mathbb{E}\left[r(\mathbf{s}_T, \mathbf{a}_T)\right]$. Now, subject to the entropy constraints and again using the dual problem, we have

$$\begin{aligned}
&\max_{\pi_{T-1}}\left(\mathbb{E}\left[r(\mathbf{s}_{T-1}, \mathbf{a}_{T-1})\right] + \max_{\pi_T}\mathbb{E}\left[r(\mathbf{s}_T, \mathbf{a}_T)\right]\right) \qquad (16)\\
&\quad= \max_{\pi_{T-1}}\left(Q_{T-1}^*(\mathbf{s}_{T-1}, \mathbf{a}_{T-1}) - \alpha_T\mathcal{H}\right)\\
&\quad= \min_{\alpha_{T-1}\geq 0}\max_{\pi_{T-1}}\left(\mathbb{E}\left[Q_{T-1}^*(\mathbf{s}_{T-1}, \mathbf{a}_{T-1})\right] - \mathbb{E}\left[\alpha_{T-1}\log\pi(\mathbf{a}_{T-1}|\mathbf{s}_{T-1})\right] - \alpha_{T-1}\mathcal{H}\right) + \alpha_T^*\mathcal{H}.
\end{aligned}$$

In this way, we can proceed backwards in time and recursively optimize Equation 11. Note that the optimal policy at time t is a function of the dual variable α_t. Similarly, we can solve the optimal dual variable α_t^* after solving for Q_t^* and π_t^*:

$$\alpha_t^* = \arg\min_{\alpha_t}\mathbb{E}_{\mathbf{a}_t\sim\pi_t^*}\left[-\alpha_t\log\pi_t^*(\mathbf{a}_t|\mathbf{s}_t;\alpha_t) - \alpha_t\bar{\mathcal{H}}\right]. \tag{17}$$

The solution in Equation 17 along with the policy and soft Q-function updates described in Section 4 constitute the core of our algorithm, and in theory, exactly solving them recursively optimize the optimal entropy-constrained maximum expected return objective in Equation 11, but in practice, we will need to resort to function approximators and stochastic gradient descent.

Algorithm 1 Soft Actor-Critic

Input: θ_1, θ_2, ϕ ▷ Initial parameters
$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target network weights
$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay pool
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$ ▷ Sample action from the policy
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ ▷ Sample transition from the environment
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ ▷ Store the transition in the replay pool
  end for
  for each gradient step **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ ▷ Update the Q-function parameters
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights
    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ▷ Adjust temperature
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target network weights
  end for
end for
Output: θ_1, θ_2, ϕ ▷ Optimized parameters

6 Practical Algorithm

Our algorithm makes use of two soft Q-functions to mitigate positive bias in the policy improvement step that is known to degrade performance of value based methods (Hasselt, 2010; Fujimoto et al., 2018). In particular, we parameterize two soft Q-functions, with parameters θ_i, and train them independently to optimize $J_Q(\theta_i)$. We then use the minimum of the the soft Q-functions for the stochastic gradient in Equation 6 and policy gradient in Equation 10, as proposed by Fujimoto et al. (2018). Although our algorithm can learn challenging tasks, including a 21-dimensional Humanoid, using just a single Q-function, we found two soft Q-functions significantly speed up training, especially on harder tasks.

In addition to the soft Q-function and the policy, we also learn α by minimizing the dual objective in Equation 17. This can be done by approximating dual gradient descent (Boyd & Vandenberghe, 2004). Dual gradient descent alternates between optimizing the Lagrangian with respect to the primal variables to convergence, and then taking a gradient step on the dual variables. While optimizing with respect to the primal variables fully is impractical, a truncated version that performs incomplete optimization (even for a single gradient step) can be shown to converge under convexity assumptions (Boyd & Vandenberghe, 2004). While such assumptions do not apply to the case of nonlinear function approximators such as neural networks, we found this approach to still work in practice. Thus, we compute gradients for α with the following objective:

$$J(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t}\left[-\alpha \log \pi_t(\mathbf{a}_t|\mathbf{s}_t) - \alpha\bar{\mathcal{H}}\right]. \tag{18}$$

The final algorithm is listed in Algorithm 1. The method alternates between collecting experience from the environment with the current policy and updating the function approximators using the stochastic gradients from batches sampled from a replay pool. Using off-policy data from a replay pool is feasible because both value estimators and the policy can be trained entirely on off-policy data. The algorithm is agnostic to the parameterization of the policy, as long as it can be evaluated for any arbitrary state-action tuple.

7 Experiments

The goal of our experimental evaluation is to understand how the sample complexity and stability of our method compares with prior off-policy and on-policy deep reinforcement learning algorithms. We compare our method to prior techniques on a range of challenging continuous control tasks from the OpenAI gym benchmark suite (Brockman et al., 2016) and also on the rllab implementation of the Humanoid task (Duan et al., 2016). Although the easier tasks can be solved by a wide range of different algorithms, the more complex benchmarks, such as the 21-dimensional Humanoid (rllab),

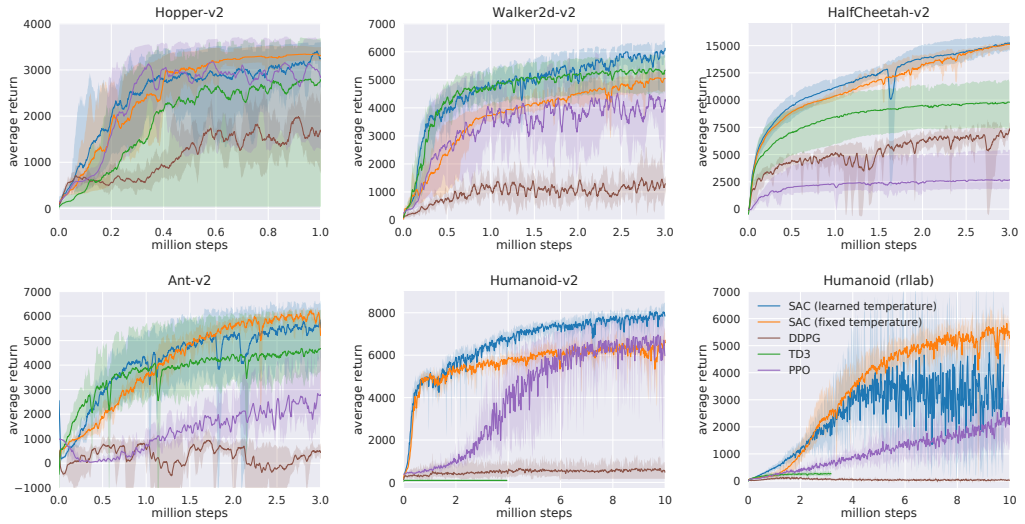


Figure 1: Training curves on continuous control benchmarks. Soft actor-critic (blue and yellow) performs consistently across all tasks and outperforming both on-policy and off-policy methods in the most challenging tasks.

are exceptionally difficult to solve with off-policy algorithms (Duan et al., 2016). The stability of the algorithm also plays a large role in performance: easier tasks make it more practical to tune hyperparameters to achieve good results, while the already narrow basins of effective hyperparameters become prohibitively small for the more sensitive algorithms on the hardest benchmarks, leading to poor performance (Gu et al., 2016).

7.1 Simulated Benchmarks

We compare our method to deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015), an algorithm that is regarded as one of the more efficient off-policy deep RL methods (Duan et al., 2016); proximal policy optimization (PPO) (Schulman et al., 2017b), a stable and effective on-policy policy gradient algorithm; and soft Q-learning (SQL) (Haarnoja et al., 2017), a recent off-policy algorithm for learning maximum entropy policies. Our SQL implementation also includes two Q-functions, which we found to improve its performance in most environments. We additionally compare to twin delayed deep deterministic policy gradient algorithm (TD3) (Fujimoto et al., 2018), using the author-provided implementation. This is an extension to DDPG, proposed concurrently to our method, that first applied the double Q-learning trick to continuous control along with other improvements. We turned off the exploration noise for evaluation for DDPG and PPO. For maximum entropy algorithms, which do not explicitly inject exploration noise, we either evaluated with the exploration noise (SQL) or use the mean action (SAC). The source code of our SAC implementation[2] is available online.

Figure 1 shows the total average return of evaluation rollouts during training for DDPG, PPO, and TD3. We train five different instances of each algorithm with different random seeds, with each performing one evaluation rollout every 1000 environment steps. The solid curves corresponds to the mean and the shaded region to the minimum and maximum returns over the five trials. For SAC, we include both a version, where the temperature parameter is fixed and treated as a hyperparameter and tuned for each environment separately (orange), and a version where the temperature is adjusted automatically (blue). The results show that, overall, SAC performs comparably to the baseline methods on the easier tasks and outperforms them on the harder tasks with a large margin, both in terms of learning speed and the final performance. For example, DDPG fails to make any progress on Ant-v1, Humanoid-v1, and Humanoid (rllab), a result that is corroborated by prior work (Gu et al., 2016; Duan et al., 2016). SAC also learns considerably faster than PPO as a consequence of the large batch sizes PPO needs to learn stably on more high-dimensional and complex tasks. Another

[2] http://github.com/rail-berkeley/softlearning/

maximum entropy RL algorithm, SQL, can also learn all tasks, but it is slower than SAC and has worse asymptotic performance. The quantitative results attained by SAC in our experiments also compare very favorably to results reported by other methods in prior work (Duan et al., 2016; Gu et al., 2016; Henderson et al., 2017), indicating that both the sample efficiency and final performance of SAC on these benchmark tasks exceeds the state of the art. The results also indicate that the automatic temperature tuning scheme works well across all the environments, and thus effectively eliminates the need for tuning the temperature. All hyperparameters used in this experiment for SAC are listed in Appendix D.

7.2 Quadrupedal Locomotion in the Real World

In this section, we describe an application of our method to learn walking gaits directly in the real world. We use the Minitaur robot, a small-scale quadruped with eight direct-drive actuators (Kenneally et al., 2016). Each leg is controlled by two actuators that allow it to move in the sagittal plane. The Minitaur is equipped with motor encoders that measure the motor angles and an IMU that measures orientation and angular velocity of Minitaur's base. The action space are the swing angle and the extension of each leg, which are then mapped to desired motor positions and tracked with a PD controller (Tan et al., 2018). The observations include the motor angles as well as roll and pitch angles and angular velocities of the base. We exclude yaw since it is unreliable due to drift and irrelevant for the walking task. Note that latencies and contacts in the system make the dynamics non-Markovian, which can significantly degrade learning performance. We therefore construct the state out of the current and past five observations and actions. The reward function rewards large forward velocity, which is estimated using a motion capture system, and penalizes large angular accelerations, computed via finite differentiation from the last three actions. We also found it necessary to penalize for large pitch angles and for extending the front legs under the robot, which we found to be the most common failure cases that would require manual reset. We parameterize the policy and the value functions with feed-forward neural networks with two hidden-layers and 256 neurons per layer.

We have developed a semi-automatic robot training pipeline that consists of two components parallel jobs: training and data collection. These jobs run asynchronously on two different computers. The training process runs on a workstation, which updates the neural networks and periodically downloads the latest data from the robot and uploads the latest policy to the robot. On the robot, the on-board Nvidia Jetson TX2 runs the data collection job, which executes the policy, collects the trajectory and uploads these data to the workstation through Ethernet. Once the training is started, minimal human intervention is needed, except for the need to reset the robot state if it falls or drifts far from the initial state.

This learning task presents substantially challenges for real-world reinforcement learning. The robot is underactuated, and must therefore delicately balance contact forces on the legs to make forward progress. An untrained policy can lose balance and fall, and too many falls will eventually damage the robot, making sample-efficient learning essentially. Our method successfully learns to walk from 160k environment steps, or approximately 400 episodes with maximum length of 500 steps, which amount to approximately 2 hours of real-world training time.

However, in the real world, the utility of a locomotion policy hinges critically on its ability to generalize to different terrains and obstacles. Although we trained our policy only on flat terrain, as illustrated in Figure 2 (first row), we then tested it on varied terrains and obstacles (other rows). Because soft actor-critic learns robust policies, due to entropy maximization at training time, the policy can readily generalize to these perturbations without any additional learning. The robot is able to walk up and down a slope (first row), ram through an obstacle made of wooden blocks (second row), and step down stairs (third row) without difficulty, despite not being trained in these settings. To our knowledge, this experiment is the first example of a deep reinforcement learning algorithm learning underactuated quadrupedal locomotion directly in the real world without any simulation or pretraining. We have included videos of the the training process and evaluation on our project website[3].

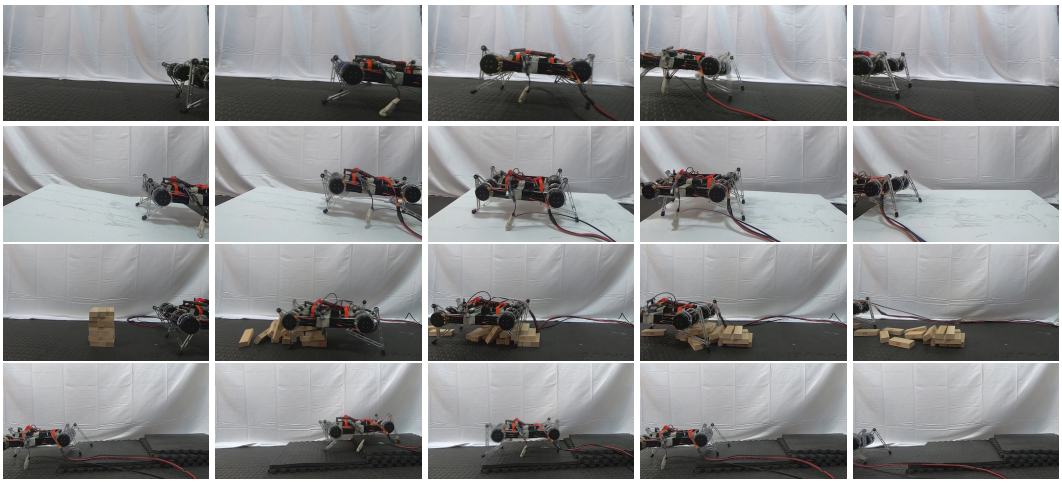


Figure 2: We trained the Minitaur robot to walk on flat terrain (first row) and tested how the learned gait generalizes to unseen situations (other rows)

7.3 Dexterous Hand Manipulation

Our second real-world robotic task involves training a 3-finger dexterous robotic hand to manipulate an object. The hand is based on the "dclaw" hand, discussed by (Zhu et al., 2018). This hand has 9 DoFs, each controlled by a Dynamixel servo-motor. The policy controls the hand by sending target joint angle positions for the on-board PID controller. The manipulation task requires the hand to rotate a "valve"-like object (resembling a sink faucet), as shown in Figure 3. In order to perceive the valve, the robot must use raw RGB images, which are illustrated in the second row of Figure 3. The images are processed in a neural network, consisting of two convolutional (four 3x3 filters) and max pool (3x3) layers, followed by two fully connected layers (256 units). The robot must rotate the valve into the correct position, with the colored part of the valve facing directly to the right, from any random starting position. The initial position of the valve is reset uniformly at random for each episode, forcing the policy to learn to use the raw RGB images to perceive the current valve orientation. A small motor is attached to the valve to automate resets and to provide the ground truth position for the determination of the reward function. The position of this motor is not provided to the policy.

This task is exceptionally challenging due to both the perception challenges and the physical difficulty of rotating the valve with such a complex robotic hand. As can be seen in the accompanying video on the project website[3], rotating the valve requires a complex finger gait where the robot moves the fingers over the valve in a coordinated pattern, and stops precisely at the desired position.

Learning this task directly from raw RGB images requires 300k environment interaction steps, which is the equivalent of 20 hours of training, including all resets and neural network training time (Figure 4). To our knowledge, this task represents one of the most complex robotic manipulation tasks learned directly end-to-end from raw images in the real world with deep reinforcement learning, without any simulation or pretraining. We also learned the same task without images by feeding the valve position directly to the neural networks. In that case, learning takes 3 hours, which is substantially faster than what has been reported earlier on the same task using PPO (7.4 hours) (Zhu et al., 2018).

8 Conclusion

In this article, we presented soft actor-critic (SAC), an off-policy maximum entropy deep reinforcement learning algorithm that provides sample-efficient learning while retaining the benefits of entropy maximization and stability. Our theoretical results derive soft policy iteration, which we show to converge to the optimal policy. From this result, we can formulate a practical soft actor-critic algorithm

[3] https://sites.google.com/view/sac-and-applications/

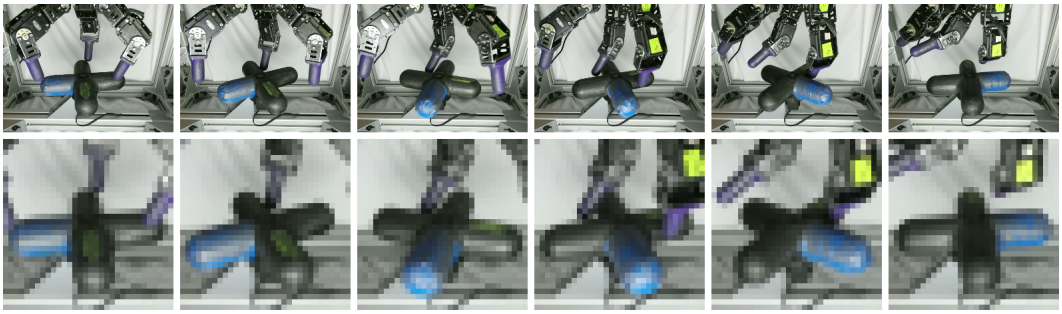
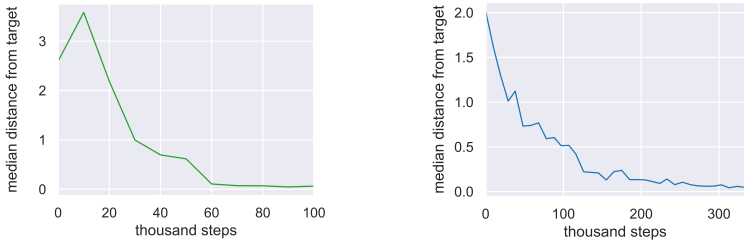


Figure 3: Valve rotation task with Dynamixel Claw. Top row shows a high resolution image sequence of a rollout with the policy. Bottom row shows the 32x32 pixel image observations of the same situations. The policy is also provided the joint positions and velocities of the fingers, but it has to infer the valve position from the image.



(a) Learning from valve joint angle.

(b) Learning from images.

Figure 4: Learning curves for the valve rotation task. The curves correspond to the median distance (measured in radians) of the valve from the target angle during a rollout. (a) Learning without images takes about 3 hours. In this case, the valve is reset to point away from the target initially. (b) Learning from images takes about 20 hours. The initial valve position is sampled uniformly at random.

that can be used to train deep neural network policies, and we empirically show that it matches or exceeds the performance of state-of-the-art model-free deep RL methods, including the off-policy TD3 algorithm and the on-policy PPO algorithm without any environment specific hyperparameter tuning. Our real-world experiments indicate that soft actor-critic is robust and sample efficient enough for robotic tasks learned directly in the real world, such as locomotion and dexterous manipulation. To our knowledge, these results represent the first evaluation of deep reinforcement learning for real-world training of underactuated walking skills with a quadrupedal robot, as well as one of the most complex dexterous manipulation behaviors learned with deep reinforcement learning end-to-end from raw image observations.

Acknowledgments

We would like to thank Vitchyr Pong and Haoran Tang for constructive discussions during the development of soft actor-critic, Vincent Vanhoucke for his support towards the project at Google, and Amazon for providing computing support.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, pp. 834–846, 1983.

Bhatnagar, S., Precup, D., Silver, D., Sutton, R. S., Maei, H. R., and Szepesvári, C. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1204–1212, 2009.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Duan, Y., Chen, X. Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning (ICML)*, 2016.

Fox, R., Pakman, A., and Tishby, N. Taming the noise in reinforcement learning via soft updates. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Gruslys, A., Azar, M. G., Bellemare, M. G., and Munos, R. The reactor: A sample-efficient actor-critic architecture. *arXiv preprint arXiv:1704.04651*, 2017.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *International Conference on Robotics and Automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning (ICML)*, pp. 1352–1361, 2017.

Haarnoja, T., Hartikainen, K., Abbeel, P., and Levine, S. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018a.

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., and Levine, S. Composable deep reinforcement learning for robotic manipulation. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018b.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018c.

Hasselt, H. V. Double Q-learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2613–2621, 2010.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2944–2952, 2015.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Kappen, H. J. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory And Experiment*, 2005(11):P11011, 2005.

Kenneally, G., De, A., and Koditschek, D. E. Design principles for a family of direct-drive legged robots. *IEEE Robotics and Automation Letters*, 1(2):900–907, 2016.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *International Conference for Learning Presentations (ICLR)*, 2015.

Levine, S. and Koltun, V. Guided policy search. In *International Conference on Machine Learning (ICML)*, pp. 1–9, 2013.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2772–2782, 2017a.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Trust-PCL: An off-policy trust region method for continuous control. *arXiv preprint arXiv:1707.01891*, 2017b.

O'Donoghue, B., Munos, R., Kavukcuoglu, K., and Mnih, V. PGQ: Combining policy gradient and Q-learning. *arXiv preprint arXiv:1611.01626*, 2016.

Peters, J. and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

Rawlik, K., Toussaint, M., and Vijayakumar, S. On stochastic optimal control and reinforcement learning by approximate inference. *Robotics: Science and Systems (RSS)*, 2012.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pp. 1889–1897, 2015.

Schulman, J., Abbeel, P., and Chen, X. Equivalence between policy gradients and soft Q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2014.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016. ISSN 0028-0836. Article.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems (RSS)*, 2018.

Thomas, P. Bias in natural actor-critic algorithms. In *International Conference on Machine Learning (ICML)*, pp. 441–448, 2014.

Todorov, E. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, pp. 1369–1376. MIT Press, 2007.

Todorov, E. General duality between optimal control and estimation. In *IEEE Conference on Decision and Control (CDC)*, pp. 4286–4292. IEEE, 2008.

Toussaint, M. Robot trajectory optimization using approximate inference. In *International Conference on Machine Learning (ICML)*, pp. 1049–1056. ACM, 2009.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zhu, H., Gupta, A., Rajeswaran, A., Levine, S., and Kumar, V. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. *arXiv preprint arXiv:1810.06045*, 2018.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1433–1438, 2008.

Appendix

A Infinite Horizon Discounted Maximum Entropy Objective

The exact definition of the discounted maximum entropy objective is complicated by the fact that, when using a discount factor for policy gradient methods, we typically do not discount the state distribution, only the rewards. In that sense, discounted policy gradients typically do not optimize the true discounted objective. Instead, they optimize average reward, with the discount serving to reduce variance, as discussed by Thomas (2014). However, we can define the objective that *is* optimized under a discount factor as

$$J(\pi) = \sum_{t=0}^{\infty} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[\sum_{l=t}^{\infty} \gamma^{l-t} \, \mathbb{E}_{\mathbf{s}_l \sim p, \mathbf{a}_l \sim \pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot \,|\mathbf{s}_t)) | \mathbf{s}_t, \mathbf{a}_t \right] \right]. \tag{19}$$

This objective corresponds to maximizing the discounted expected reward and entropy for future states originating from every state-action tuple $(\mathbf{s}_t, \mathbf{a}_t)$ weighted by its probability ρ_π under the current policy.

B Proofs

B.1 Lemma 1

Lemma 1 (Soft Policy Evaluation). *Consider the soft Bellman backup operator $\mathcal{T}^\pi$ in Equation 2 and a mapping $Q^0 : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q^{k+1} = \mathcal{T}^\pi Q^k$. Then the sequence Q^k will converge to the soft Q-value of π as $k \rightarrow \infty$.*

Proof. Define the entropy augmented reward as $r_\pi(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{s}_{t+1} \sim p} \left[\mathcal{H} \left(\pi(\cdot \,|\mathbf{s}_{t+1}) \right) \right]$ and rewrite the update rule as

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r_\pi(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p, \mathbf{a}_{t+1} \sim \pi} \left[Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) \right] \tag{20}$$

and apply the standard convergence results for policy evaluation (Sutton & Barto, 1998). The assumption $|\mathcal{A}| < \infty$ is required to guarantee that the entropy augmented reward is bounded. □

B.2 Lemma 2

Lemma 2 (Soft Policy Improvement). *Let $\pi_{\text{old}} \in \Pi$ and let π_{new} be the optimizer of the minimization problem defined in Equation 4. Then $Q^{\pi_{\text{new}}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$.*

Proof. Let $\pi_{\text{old}} \in \Pi$ and let $Q^{\pi_{\text{old}}}$ and $V^{\pi_{\text{old}}}$ be the corresponding soft state-action value and soft state value, and let π_{new} be defined as

$$\begin{aligned}\pi_{\text{new}}(\,\cdot\,|\mathbf{s}_t) &= \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\,\cdot\,|\mathbf{s}_t) \parallel \exp\left(Q^{\pi_{\text{old}}}(\mathbf{s}_t,\,\cdot\,) - \log Z^{\pi_{\text{old}}}(\mathbf{s}_t)\right)\right) \\ &= \arg\min_{\pi' \in \Pi} J_{\pi_{\text{old}}}(\pi'(\,\cdot\,|\mathbf{s}_t)). \end{aligned} \tag{21}$$

It must be the case that $J_{\pi_{\text{old}}}(\pi_{\text{new}}(\,\cdot\,|\mathbf{s}_t)) \leq J_{\pi_{\text{old}}}(\pi_{\text{old}}(\,\cdot\,|\mathbf{s}_t))$, since we can always choose $\pi_{\text{new}} = \pi_{\text{old}} \in \Pi$. Hence

$$\mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{new}}}\left[\log \pi_{\text{new}}(\mathbf{a}_t|\mathbf{s}_t) - Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t) + \log Z^{\pi_{\text{old}}}(\mathbf{s}_t)\right] \leq \mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{old}}}\left[\log \pi_{\text{old}}(\mathbf{a}_t|\mathbf{s}_t) - Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t) + \log Z^{\pi_{\text{old}}}(\mathbf{s}_t)\right], \tag{22}$$

and since partition function $Z^{\pi_{\text{old}}}$ depends only on the state, the inequality reduces to

$$\mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_{\text{new}}(\mathbf{a}_t|\mathbf{s}_t)\right] \geq V^{\pi_{\text{old}}}(\mathbf{s}_t). \tag{23}$$

Next, consider the soft Bellman equation:

$$\begin{aligned} Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t) &= r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V^{\pi_{\text{old}}}(\mathbf{s}_{t+1})\right] \\ &\leq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[\mathbb{E}_{\mathbf{a}_{t+1} \sim \pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \log \pi_{\text{new}}(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right]\right] \\ &\;\;\vdots \\ &\leq Q^{\pi_{\text{new}}}(\mathbf{s}_t, \mathbf{a}_t), \end{aligned} \tag{24}$$

where we have repeatedly expanded $Q^{\pi_{\text{old}}}$ on the RHS by applying the soft Bellman equation and the bound in Equation 23. Convergence to $Q^{\pi_{\text{new}}}$ follows from Lemma 1. □

B.3 Theorem 1

Theorem 1 (Soft Policy Iteration). *Repeated application of soft policy evaluation and soft policy improvement to any $\pi \in \Pi$ converges to a policy π^* such that $Q^{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ for all $\pi \in \Pi$ and $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$, assuming $|\mathcal{A}| < \infty$.*

Proof. Let π_i be the policy at iteration i. By Lemma 2, the sequence Q^{π_i} is monotonically increasing. Since Q^{π} is bounded above for $\pi \in \Pi$ (both the reward and entropy are bounded), the sequence converges to some π^*. We will still need to show that π^* is indeed optimal. At convergence, it must be case that $J_{\pi^*}(\pi^*(\,\cdot\,|\mathbf{s}_t)) < J_{\pi^*}(\pi(\,\cdot\,|\mathbf{s}_t))$ for all $\pi \in \Pi$, $\pi \neq \pi^*$. Using the same iterative argument as in the proof of Lemma 2, we get $Q^{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) > Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$, that is, the soft value of any other policy in Π is lower than that of the converged policy. Hence π^* is optimal in Π. □

C Enforcing Action Bounds

We use an unbounded Gaussian as the action distribution. However, in practice, the actions needs to be bounded to a finite interval. To that end, we apply an invertible squashing function ($\tanh$) to the Gaussian samples, and employ the change of variables formula to compute the likelihoods of the bounded actions. In the other words, let $\mathbf{u} \in \mathbb{R}^D$ be a random variable and $\mu(\mathbf{u}|\mathbf{s})$ the corresponding density with infinite support. Then $\mathbf{a} = \tanh(\mathbf{u})$, where $\tanh$ is applied elementwise, is a random variable with support in $(-1, 1)$ with a density given by

$$\pi(\mathbf{a}|\mathbf{s}) = \mu(\mathbf{u}|\mathbf{s}) \left|\det\left(\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}\mathbf{u}}\right)\right|^{-1}. \tag{25}$$

Since the Jacobian $\mathrm{d}\mathbf{a}/\mathrm{d}\mathbf{u} = \mathrm{diag}(1 - \tanh^2(\mathbf{u}))$ is diagonal, the log-likelihood has a simple form

$$\log \pi(\mathbf{a}|\mathbf{s}) = \log \mu(\mathbf{u}|\mathbf{s}) - \sum_{i=1}^{D} \log\left(1 - \tanh^2(u_i)\right), \tag{26}$$

where u_i is the i^{th} element of $\mathbf{u}$.

D Hyperparameters

Table 1 lists the common SAC parameters used in the comparative evaluation in Figure 1.

Table 1: SAC Hyperparameters

Parameter	Value
optimizer	Adam (Kingma & Ba, 2015)
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers (all networks)	2
number of hidden units per layer	256
number of samples per minibatch	256
entropy target	$-\dim(\mathcal{A})$ (e.g. , -6 for HalfCheetah-v1)
nonlinearity	ReLU
target smoothing coefficient (τ)	0.005
target update interval	1
gradient steps	1