Composable Deep Reinforcement Learning for Robotic Manipulation

Tuomas Haarnoja[1], Vitchyr Pong[1], Aurick Zhou[1], Murtaza Dalal[1], Pieter Abbeel[1,2], Sergey Levine[1]

***Abstract*— Model-free deep reinforcement learning has been shown to exhibit good performance in domains ranging from video games to simulated robotic manipulation and locomotion. However, model-free methods are known to perform poorly when the interaction time with the environment is limited, as is the case for most real-world robotic tasks. In this paper, we study how maximum entropy policies trained using soft Q-learning can be applied to real-world robotic manipulation. The application of this method to real-world manipulation is facilitated by two important features of soft Q-learning. First, soft Q-learning can learn multimodal exploration strategies by learning policies represented by expressive energy-based models. Second, we show that policies learned with soft Q-learning can be composed to create new policies, and that the optimality of the resulting policy can be bounded in terms of the divergence between the composed policies. This compositionality provides an especially valuable tool for real-world manipulation, where constructing new policies by composing existing skills can provide a large gain in efficiency over training from scratch. Our experimental evaluation demonstrates that soft Q-learning is substantially more sample efficient than prior model-free deep reinforcement learning methods, and that compositionality can be performed for both simulated and real-world tasks.**

I. INTRODUCTION

The intersection of expressive, general-purpose function approximators, such as neural networks, with general purpose model-free reinforcement learning algorithms that can be used to acquire complex behavioral strategies holds the promise of automating a wide range of robotic behaviors: reinforcement learning provides the formalism for reasoning about sequential decision making, while large neural networks provide the representation that can, in principle, be used to represent any behavior with minimal manual engineering. However, applying model-free reinforcement learning algorithms with multilayer neural network representations (i.e., deep reinforcement learning) to real-world robotic control problems has proven to be very difficult in practice: the sample complexity of model-free methods tends to be quite high, and is increased further by the inclusion of high-capacity function approximators. Prior work has sought to alleviate these issues by parallelizing learning across multiple robots [1], making use of example demonstrations [2], [3], or training in simulation and relying on an accurate model to enable transfer to the real world [4], [5]. All of these approaches carry additional assumptions and limitations. Can we instead devise model-free reinforcement learning algorithms that are efficient enough to train multilayer neural network models directly in the real world, without reliance on simulation, demonstrations, or multiple robots?

[1]Berkeley Artificial Intelligence Research, UC Berkeley, [2]Open AI
{haarnoja, vitchyr, azhou42, mdalal, pabbeel, svlevine}@berkeley.edu

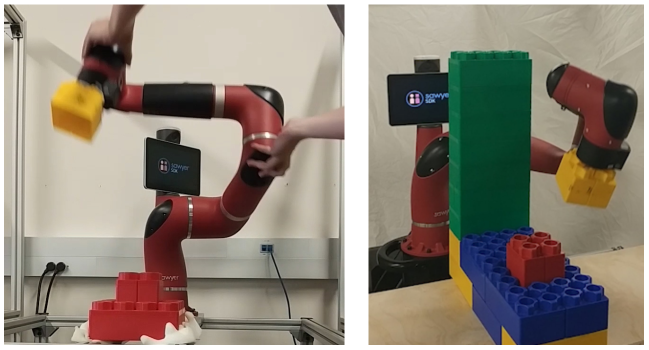

Fig. 1. We trained a Sawyer robot to stack Lego blocks together using maximum entropy reinforcement learning algorithm called soft Q-learning. Training a policy from scratch takes less than two hours, and the learned policy is extremely robust against perturbations (left figure). We also demonstrate how learned policies can be combined to form new compound skills, such as stacking while avoiding a tower of Lego blocks (right figure).

We hypothesize that the maximum entropy principle [6] can yield an effective framework for practical, real-world deep reinforcement learning due to the following two properties. First, maximum entropy policies provide an inherent, informed exploration strategy by expressing a stochastic policy via the Boltzmann distribution, with the energy corresponding to the reward-to-go or Q-function [7]. This distribution assigns a non-zero probability to all actions, but actions with higher expected rewards are more likely to be sampled. As a consequence, the policy will automatically direct exploration into regions of higher expected return. This property, which can be thought of as a soft combination of exploration and exploitation, can be highly beneficial in real-world applications, since it provides considerably more structure than ϵ-greedy exploration and, as shown in our experiments, substantially improves sample complexity. Second, as we show in this paper, independently trained maximum entropy policies can be composed together by adding their Q-functions, yielding a new policy for the combined reward function that is provably close to the corresponding optimal policy. Composability of controllers, which is not typically possible in standard reinforcement learning, is especially important for real-world applications, where reuse of past experience can greatly improve sample efficiency for tasks that can naturally be decomposed into simpler sub-problems. For instance, a policy for a pick-and-place task can be decomposed into (1) reaching specific x-coordinates, (2) reaching specific y-coordinates, and (3) avoiding certain obstacles. Such decomposable policies can therefore be learned in three stages, each yielding a sub-policy, which can later be combined offline without the need to interact with the environment.

The primary contribution of this paper is a framework, based on the recently introduced soft Q-learning (SQL) algorithm [7], for learning robotic manipulation skills with expressive neural network policies. We illustrate that this framework provides an efficient mechanism for learning a variety of robotic skills and outperforms state-of-the-art model-free deep reinforcement learning methods in terms of sample efficiency on a real robotic system. Our empirical results indicate that SQL substantially outperforms deep deterministic policy gradient (DDPG) and normalized advantage functions (NAF), which have previously been explored for real world model-free robotic learning with neural networks. We also demonstrate a novel extension of the SQL algorithm that enables composition of previously learned skills We present a novel theoretical bound on the difference between the policy obtained through composition and the optimal policy for the composed reward function, which applies to SQL and other reinforcement learning methods based on soft optimality. In our experiments, we leverage the compositionality of maximum entropy policies in both simulated and physical domains, showing robust learning of diverse skills and outperforming existing state-of-the-art methods in terms of sample efficiency.

II. RELATED WORK

One of the crucial choices in applying RL for robotic manipulation is the choice of representation. Policies based on dynamic movement primitives and other trajectory-centric representations, such as splines, have been particularly popular, due to the ease of incorporating demonstrations, stability, and relatively low dimensionality [8], [9], [10], [11]. However, trajectory-centric policies are limited in their expressive power, particularly when it comes to integrating rich sensory information and reacting continuously to the environment. For this reason, recent works have sought to explore more expressive representations, including deep neural networks, at the cost of higher sample complexity. A number of prior works have sought to address the increased sample complexity by employing model-based methods. For example, guided policy search [12] learns a model-based teacher that supervises the training of a deep policy network. Other works leverage simulation: Ghadirzadeh et al. [13] considers vision-based manipulation by training perception and behavior networks in simulation and learns only a low-dimensional intermediate layer with real-world interaction. Some works learn vision-based policies completely in simulation and then transfers them into real world [14], [15], [16], [17]. For this approach to work, the simulator needs to model the system accurately and there needs to be significant variation in the appearance of the synthetic images in order to handle the unavoidable domain-shift between the simulation and the real-world [15], [16], [18], [19]. Another common approach to make reinforcement learning algorithms more sample efficient is to learn from demonstrations [20], [21], [22], [3], but this comes at the cost of additional instrumentation and human supervision.

Perhaps the most closely related recent work aims to apply deterministic model-free deep RL algorithms to real-world robotic manipulation. Gu et al. [23] used NAF to learn door opening and reaching by parallelizing across multiple real-world robots, and Večerík et al. [3] extended DDPG to real-world robotic manipulation by including example demonstrations. Our experiments show that SQL can learn real-world manipulation more proficiently and with fewer samples than methods such as DDPG and NAF, without requiring demonstrations, simulated experience, or additional supervision. We also show that SQL can be extended to enable composition of several previously trained skills.

Soft Q-learning trains policies that maximize both reward and entropy. Maximum entropy policies have been discussed in many contexts, ranging from applications to inverse reinforcement learning [6], connections to the Kalman duality and optimal control [24], [25], [26], and to incorporation of prior knowledge for reinforcement learning [27]. More recently, several papers have noted the connection between soft Q-learning and policy gradient methods [7], [28], [29]. While most of the prior works assume a discrete action space, Nachum et al. [30] approximates the maximum entropy distribution with a Gaussian, and, to our knowledge, soft Q-learning [7] is the only method that approximates the true maximum entropy policy with an expressive inference network, which is essential for tasks requiring diverse behaviour at test time and involving multimodal solutions.

We demonstrate that, not only does SQL lead to more sample-efficient real-world reinforcement learning, but that it can also provide a framework for composing several previously trained policies to initialize compound skills. We derive a novel bound in Section IV that provides insight into when composition is likely to succeed. Several prior works [31], [24] have studied composition in the context of soft optimality, but typically in a different framework: instead of considering composition where the reward functions of constituent skills are added (i.e., do X *and* Y), they considered settings where a soft maximum ("log-sum-exp") over the reward functions is used as the reward for the composite skill (i.e., do X *or* Y). We argue that the former is substantially more useful than the latter for robotic skills, since it allows us to decompose a task into multiple objectives that all need to be satisfied.

III. PRELIMINARIES

Soft Q-learning, which we extend in this work for learning composable controllers for real-world robotic manipulation, is based on the framework of maximum entropy reinforcement learning [6], [32], [25], [31]. In this section, we introduce the formalism of reinforcement learning, describe the soft Q-learning algorithm proposed in prior work [7], and discuss how it relates to conventional reinforcement learning algorithms, namely DDPG and NAF.

A. Notation

We will consider an infinite-horizon Markov decision process (MDP), defined by $(\mathcal{S}, \mathcal{A}, p, r)$, where the state space

$\mathcal{S}$ and the action space $\mathcal{A}$ are assumed to be continuous, and the unknown state transition probability $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ represents the probability density of the next state $\mathbf{s}_{t+1}$ given the current state $\mathbf{s}_t$ and current action $\mathbf{a}_t$. The environment emits a bounded reward $r(\mathbf{s}_t, \mathbf{a}_t)$ on each transition. We will use ρ_π to denote the state or state-action marginals of the trajectory distribution induced by a policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$. We will drop the time indices from $\mathbf{s}_t$ and $\mathbf{a}_t$ and use $(\mathbf{s}, \mathbf{a})$ and $(\mathbf{s}', \mathbf{a}')$ to denote states and actions for consecutive time steps whenever they can be inferred from the context.

B. Maximum Entropy Reinforcement Learning

The standard RL objective seeks a policy that maximizes the expected sum of rewards $\sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t, \mathbf{a}_t)\right]$. In this paper, we consider a more general maximum entropy objective [6], [32], [25], [31], which favors stochastic policies by augmenting the objective with an expected entropy over ρ_π:

$$J(\pi) = \sum_{t=0}^{T-1} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))\right]. \qquad (1)$$

The temperature parameter, α, determines the relative importance of the entropy term against the reward, and thus controls the stochasticity of the optimal policy, and the conventional objective can be recovered in the limit as $\alpha \to 0$. 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 optimal behavior. In particular, in problem settings where multiple actions seem equally attractive, the policy will commit equal probability mass to those actions. Lastly, prior work has observed substantially improved exploration from this property [7], [29], and in our experiments, we observe that it considerably improves learning speed for real-world robotic manipulation. If we wish to extend the 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. Writing down the precise maximum entropy objective for the infinite horizon discounted case is more involved, and we refer interested readers to prior work (see e.g. [7]).

C. Soft Q-Learning

In this work, we optimize the maximum entropy objective in (1) using the soft Q-learning algorithm [7], since, to our knowledge, it is the only existing deep RL algorithm for continuous actions that can capture arbitrarily complex action distributions—a key requirement for policies to be composable, as we discuss in Section IV. Soft Q-learning optimizes a Q-function $Q(\mathbf{s}, \mathbf{a})$ to predict the expected future return, which include the future entropy values, after taking an action $\mathbf{a}$ at state $\mathbf{s}$ and then following π. The optimal policy π^* can be expressed in terms of the optimal Q-function as an energy based model (EBM),

$$\pi^*(\mathbf{a}|\mathbf{s}) \propto \exp\left(\frac{1}{\alpha}Q^*(\mathbf{s}, \mathbf{a})\right), \qquad (2)$$

where the Q-function takes the role of negative energy. Unfortunately, we cannot evaluate the action likelihoods, since that would require knowing the partition function, which is intractable in general. However, it is possible to draw samples from this distribution by resorting to a approximate sampling method. To that end, soft Q-learning uses amortized Stein variational descent (SVGD) [33], [34] to learn a stochastic neural network to approximate samples from the desired EBM [7]. The main benefit of amortizing the cost of approximate sampling into a neural network is that producing samples at test time amounts to a single forward pass, which is fast enough to be done in real-time.

In addition to learning the optimal policy, we also need to learn the optimal Q-function. It is possible to derive an update rule for the maximum entropy objective in (1) that resembles the Bellman backup used in conventional Q-learning. This *soft* Bellman operator updates the Q-function according to

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[V(\mathbf{s}')\right], \qquad (3)$$

where the value function $V(\mathbf{s})$ is defined as the soft maximum of the Q-function over actions: (4)

$$V(\mathbf{s}) = \underset{\mathbf{a}}{\operatorname{softmax}}\, Q(\mathbf{s}, \mathbf{a}) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha}Q(\mathbf{s}, \mathbf{a})\right) d\mathbf{a}.$$

It is possible to show that the soft Bellman backup is a contraction, and the optimal Q-function is the fixed point of the iteration in (3) (see e.g. [7]). We can therefore transform any bounded function to the optimal Q-function by iteratively applying the soft Bellman backup until convergence. In practice, we represent the Q-function with a parametric function approximator, such as a multilayer neural network, and therefore we cannot perform the soft Bellman backup in its exact from. Instead, we can learn the parameters by minimizing the squared soft Bellman residual, that is, the difference between the left and right hand sides of (3). However, we will make use of the soft Bellman backup in Section IV, where we propose a method for composing new policies from existing policies.

D. Baselines

In this section, we briefly introduce DDPG and NAF, which are prior model-free reinforcement learning methods that optimize the conventional maximum return objective and have been applied to real-world robotic tasks [1], [3], and discuss their connection to soft Q-learning.

1) Deep Deterministic Policy Gradient: Deep deterministic policy gradient (DDPG) [35] is a frequently used prior method that learns a deterministic policy and a Q-function jointly. The policy is trained to maximize the Q-function, whereas the Q-function is trained to represent the expected return of the current policy. The main difference to soft Q-learning is that DDPG replaces the soft maximization in (4) with an approximate hard maximum estimated with the current policy. Since DDPG uses a deterministic policy, exploration is usually achieved by adding independent noise to the policy output. In contrast, SQL explicitly balances exploration and exploitation: the policy can explore more when it is uncertain which actions are good (all Q-values are approximately equal), but will not take actions that are clearly sub-optimal, leading to more informative exploration.

2) Normalized Advantage Functions: Normalized advantage functions (NAF) [23] consider a Q-function of a special form that is quadratic with respect to the action. The benefit of constraining the form of the Q-function is that it allows closed form computation of the maximizer. Therefore NAF replaces the soft maximum in (4) with the exact hard maximum. Similarly, the policy is represented as the deterministic maximum of the Q-function, making it easy to sample actions. In contrast to soft Q-learning, NAF is limited in its representational power and cannot represent diverse or multimodal solutions, which is a major motivation for learning maximum entropy policies.

IV. COMPOSITIONALITY OF MAXIMUM ENTROPY POLICIES

Compositionality is a powerful property that naturally emerges from maximum entropy reinforcement learning. Compositionality means that multiple policies can be combined to create a new policy that simultaneously solves all of the tasks given to the constituent policies. This feature is desirable, as it provides reusability and enables quick initialization of policies for learning complex compound skills out of previously learned building blocks. A related idea has been discussed by Todorov [36], who considers combining the independent rewards via soft maximization. However, this type of composition corresponds to solving only one of the constituent tasks at a time—a kind of disjunction (e.g., move to the target *or* avoid the obstacle). In contrast, our approach to composition corresponds to a conjunction of the tasks, which is typically more useful (e.g., move to the target *and* avoid the obstacle).

In this section, we will discuss how soft Q-functions for different tasks can be combined additively to solve multiple tasks simultaneously. While simply adding Q-functions does not generally give the Q-function for the combined task, we show that the regret from using the policy obtained by adding the constituent Q-functions together is upper bounded by the difference between the two policies that are being composed. Intuitively, if two composed policies agree on an action, or if they are indifferent towards each other's actions, then the composed policy will be closer to the optimal one.

A. Learning with Multiple Objectives

Compositionality makes learning far more efficient in multi-objective settings, which naturally emerge in robotic tasks. For example, when training a robot to move objects, one objective may be to move these objects quickly, and another objective may be to avoid collisions with a wall or a person. More generally, assume there are K tasks, defined by reward functions r_i for $i = 1, ..., K$, and that we are interested in solving any subset $\mathcal{C} \subseteq \{1, ..., K\}$ of those tasks simultaneously. A compound task can be expressed in terms of the sum of the individual rewards:

$$r_{\mathcal{C}}(\mathbf{s}, \mathbf{a}) = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} r_i(\mathbf{s}, \mathbf{a}). \tag{5}$$

The conventional approach is to solve the compound tasks by directly optimizing this compound reward $r_{\mathcal{C}}$ for every possible combination $\mathcal{C}$. Unfortunately, there are exponentially many combinations. Compositionality makes this learning process much faster by instead training an optimal policy π_i^* for each reward r_i and later combining them.

How should we combine the policies π_i^*? A simple approach is to approximate the optimal Q-function of the composed task $Q_{\mathcal{C}}^*$ with the mean of the individual Q-functions:

$$Q_{\mathcal{C}}^*(\mathbf{s}, \mathbf{a}) \approx Q_{\Sigma}(\mathbf{s}, \mathbf{a}) = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} Q_i^*(\mathbf{s}, \mathbf{a}), \tag{6}$$

where Q_{Σ} represents an approximation to the true optimal Q-function of the composed task $Q_{\mathcal{C}}^*$. One can then extract a policy π_{Σ} from this approximate Q-function using any policy-extraction algorithm. In conventional reinforcement learning without entropy regularization, we cannot make any guarantees about how close Q_{Σ} is to $Q_{\mathcal{C}}^*$. However, we show in the next section that, if the constituent policies represent optimal maximum entropy policies, then we can bound the difference between the value of the approximate policy $Q_{\mathcal{C}}^{\pi_{\Sigma}}$ and the optimal value $Q_{\mathcal{C}}^*$ for the combined task.

B. Bounding the Sub-Optimality of Composed Policies

To understand what we can expect from the performance of the composed policy π_{Σ} that is induced by Q_{Σ}, we analyze how the value of π_{Σ} relates to the unknown optimal Q-function $Q_{\mathcal{C}}^*$ corresponding to the composed reward $r_{\mathcal{C}}$. For simplicity, we consider the special case where $\alpha = 1$ and we compose just two optimal policies, given by π_i^*, with Q-functions Q_i^* and reward functions r_i. Extending the proof to more than two policies and other values of α is straightforward. We start by introducing a lower bound for the optimal combined Q-function in terms of Q_i^* and π_i^*.

Lemma 1: Let Q_1^* and Q_2^* be the soft Q-function of the optimal policies corresponding to reward functions r_1 and r_2, and define $Q_{\Sigma} \triangleq \frac{1}{2}\left(Q_1^* + Q_2^*\right)$. Then the optimal soft Q-function of the combined reward $r_{\mathcal{C}} \triangleq \frac{1}{2}\left(r_1 + r_2\right)$ satisfies

$$Q_{\Sigma}(\mathbf{s}, \mathbf{a}) \geq Q_{\mathcal{C}}^*(\mathbf{s}, \mathbf{a}) \geq Q_{\Sigma}(\mathbf{s}, \mathbf{a}) - C^*(\mathbf{s}, \mathbf{a}), \ \forall \mathbf{s} \in \mathcal{S}, \forall \mathbf{a} \in \mathcal{A}, \tag{7}$$

where C^* is the fixed point of (8)

$$C(\mathbf{s}, \mathbf{a}) \leftarrow \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})}\left[\mathcal{D}_{\frac{1}{2}}\left(\pi_1^*(\,\cdot\,|\mathbf{s}') \,\|\, \pi_2^*(\,\cdot\,|\mathbf{s}')\right) + \max_{\mathbf{a}' \in \mathcal{A}} C(\mathbf{s}', \mathbf{a}')\right],$$

and $\mathcal{D}_{\frac{1}{2}}\left(\,\cdot\, \|\, \cdot\,\right)$ is the Rényi divergence of order $1/2$.

Proof: See Appendix A. ■

This lower bound tells us that the simple additive composition of Q-functions never overestimates $Q_{\mathcal{C}}^*$ by more than the divergence of the constituent policies. Interestingly, the constant C^* can be obtained as the fixed point of the conventional Bellman equation, where the divergence of the constituent policies acts as the "reward function." Thus, C^*

is the "value" of an adversarial policy that seeks to maximize the divergence. So far, we bounded Q^*_C in terms of the constituent Q-functions, which does not necessarily mean that the composed policy π_Σ will have a high value. To that end, we use soft policy evaluation [37] to bound the value of the composed policy $Q^{\pi_\Sigma}_C$:

Theorem 1: With the definitions in Lemma 1, the value of π_Σ satisfies

$$Q^{\pi_\Sigma}_C(\mathbf{s},\mathbf{a}) \geq Q^*_C(\mathbf{s},\mathbf{a}) - D^*(\mathbf{s},\mathbf{a}), \tag{9}$$

where $D^*(\mathbf{s},\mathbf{a})$ is the fixed point of (10)

$$D(\mathbf{s},\mathbf{a}) \leftarrow \gamma\,\mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\mathbb{E}_{\mathbf{a}'\sim\pi_\Sigma(\mathbf{a}'|\mathbf{s}')}\left[C^*(\mathbf{s}',\mathbf{a}') + D(\mathbf{s}',\mathbf{a}')\right]\right].$$

Proof: See Appendix B. ■

Analogously to the discussion of C^* in Lemma 1, D^* can be viewed as the fixed point of a Bellman equation, where now $\gamma C^*(\mathbf{s},\mathbf{a})$ has the role of the "reward function." Intuitively, this means that the bound becomes tight if the two policies agree in states that are visited by the composed policy. This result show that we can bound the regret from using the policy formed by composing optimal policies for the constituent rewards. While this bound is likely quite loose, it does indicate that the regret decreases as the divergence between the constituent policies decreases. This has a few interesting implications. First, deterministic policies always have infinite divergence, unless they are identical, which suggests that they are poorly suited for composition. Highly stochastic policies, such as those produced with maximum entropy algorithms like soft Q-learning, have much lower divergences. Intuitively, each policy has a wider distribution, and therefore the policies have more "overlap," thus reducing the regret from composition. As a consequence, we expect maximum entropy RL methods to produce much more composable policies.

V. EXPERIMENTS

In this section, we show that soft Q-learning can obtain substantially better sample complexity over existing model-free deep RL methods. Our experiments demonstrate fast and reliable learning both in simulated and real-world robotic manipulation domains. We also show how compositionality of maximum entropy policies, discussed in Section IV, provides a practical tool for composing new compound skills out of previously trained components. We evaluate our method on a pushing task in simulation as well as reaching, Lego block stacking, and combined stacking while avoiding tasks on a real-world Sawyer robot. Videos of all experiments can be found on our website[1] and the code is available on GitHub[2].

A. Experimental Setup

For the simulated tasks, we used the MuJoCo physics engine [38], and for the real-world experiments we used the 7-DoF Sawyer robotic manipulator. The actions are the torque commands at each joint, and the observations are the joint angles and angular velocities, as well as the end-effector position. For the simulated pushing tasks, we also include all the relevant object coordinates, and for the experiments on Sawyer, we include the end-effector position and forces estimated from the actuator currents as observations. We parameterize the Q-function and the policies with a 2-layer neural network with 100 or 200 units in each layer and rectifier linear activations.

B. Composing Policies for Pushing in Simulation

Does composing Q-functions from individual constituent policies allow us to quickly learn compound policies? To answer that question, we study the compositionality of policies in a simulated domain, where the task is to move a cylinder to a target location. This simulated evaluation allows us to provide a detailed comparison against prior methods, evaluating both their base performance and their performance under additive composition. We start by learning the Q-functions for the constituent tasks and then combining them by adding together the Q-functions to get Q_Σ. To extract a policy from Q_Σ, SQL requires training a policy to produce samples from $\exp(Q_\Sigma)$, and DDPG requires training a network that maximizes Q_Σ. In NAF, the optimal action for Q_Σ has a closed-form expression, since the constituent Q-functions are quadratic in the actions. We train the combined DDPG and SQL policies by reusing the data collected during the training of the constituent policies, thus imposing no additional sample cost.

The constituent tasks in our simulated evaluation require a planar arm to push a cylinder to specific positions along one of the two axes. A policy trained to push the disk to a specific x position can choose any arbitrary y position, and vice-versa. A maximum entropy policy, in this case, would attempt to randomly cover all y positions, while a deterministic one would pick one arbitrarily. The composed policy is formed by combining a policy trained to push a cylinder to a specific x position and a policy trained to push the cylinder to a specific y location, thus solving a combined objective for moving the disk to a particular 2D location on the table. An illustration of this task is depicted in Fig. 2, which shows a compound policy formed by combining a Q-function for pushing the disk the blue line ($x = -1$) and the orange line ($y = -1$). The final disk locations for 100 episodes of the final SQL policies are shown with dots of respective colors. The green dots illustrate the disk positions for the combined policy. Note that even though the orange policy never moves the disk to any point close to the intersection, the combined policy interpolates to the intended target correctly.

We trained four policies to push the cylinder to one of the following goals: left ($x = -1$), middle ($x = 0$), right ($x = 1$), and bottom ($y = -1$) using all three algorithm (SQL, DDPG, NAF). These goals gives three natural compositional objectives: bottom-left ($x = -1, y = -1$), bottom-middle ($x = 0, y = -1$), and bottom-right ($x = 1, y = -1$). Table I summarizes the error distance of each constituent policy (first four rows), policies trained to optimize the combined objective directly (e.g. "push bottom-left"), and composed

[1]sites.google.com/view/composing-real-world-policies/
[2]github.com/haarnoja/softqlearning

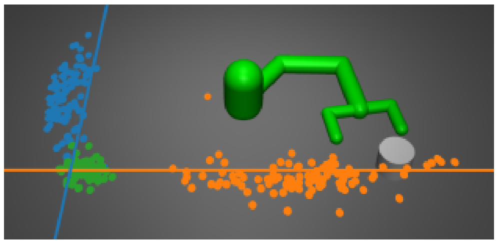

Fig. 2. Two independent policies are trained to push the cylinder to the orange line and blue line, respectively. The colored circles show samples of the final location of the cylinder for the respective policies. When the policies are combined, the resulting policy learns to push the cylinder to the lower intersection of the lines (green circle indicates final location). No additional samples from the environment are used to train the combined policy. The combined policy learns to satisfy both original goals, rather than simply averaging the final cylinder location.

TABLE I

SIMULATED PUSHING TASK: DISTANCE FROM CYLINDER TO GOAL

Task	SQL	DDPG	NAF
push left	0.13 $\pm$ 0.06	0.20 $\pm$ 0.01	**0.06** $\pm$ **0.01**
push middle	0.07 $\pm$ 0.08	0.05 $\pm$ 0.03	**0.02** $\pm$ **0.00**
push right	0.10 $\pm$ 0.08	0.10 $\pm$ 0.07	**0.09** $\pm$ **0.06**
push bottom	0.08 $\pm$ 0.07	**0.04** $\pm$ **0.02**	0.17 $\pm$ 0.08
push bottom-left	0.11 $\pm$ 0.11	0.24 $\pm$ 0.01	0.17 $\pm$ 0.01
merge bottom-left	**0.11** $\pm$ **0.05**	0.19 $\pm$ 0.10	0.16 $\pm$ 0.00
push bottom-middle	0.12 $\pm$ 0.09	0.14 $\pm$ 0.03	0.34 $\pm$ 0.08
merge bottom-middle	0.09 $\pm$ 0.09	0.12 $\pm$ 0.02	**0.02** $\pm$ **0.01**
push bottom-right	0.18 $\pm$ 0.17	0.14 $\pm$ 0.06	0.15 $\pm$ 0.06
merge bottom-right	0.15 $\pm$ 0.14	**0.09** $\pm$ **0.10**	0.43 $\pm$ 0.21

policies (e.g. "merge bottom-left"). The policies yielding the lowest error for the individual tasks are emphasized in the table. We see that SQL and DDPG perform well across all combined tasks, whereas NAF is able to combine some policies better (bottom-middle) but fails on others (bottom-right). Note that NAF policies are unimodal Gaussian, which are not well suited for compositions.

We also compared the different methods of learning a composed task in terms of training time. Example training curves for one of the targets are shown in Fig. 3. Training a policy from a given Q_Σ (red, green) takes virtually no time for any of the algorithms when compared to training a Q-function and policy from scratch (blue). In fact, the combination of NAF policies has a closed form, and it is thus not included in the graph. For comparison, we also trained a Q-function with SQL from scratch on the combined task in an offline fashion (orange), meaning that we only use samples collected by the policies trained on individual tasks. However, offline SQL fails to converge in reasonable time.

Our analysis shows that, both with SQL and DDPG, compound policies can be obtained quickly and efficiently simply by adding together the Q-functions of the individual constituent policies. We can bound the suboptimality of such policies for SQL Section IV, but for DDPG our analysis does not directly apply since DDPG is the limiting case $\alpha = 0$. Nonetheless, on simple practical problems like the one in our experiments, even the DDPG Q-functions can be composed reasonably. This suggests that composition is a powerful tool that can allow us to efficient build new skills out of old ones. In the next section, we will see that on more complex real-world manipulation tasks, SQL substantially outperforms DDPG in terms of both learning speed and final performance, and constituent policies trained with SQL can also be reused to form compound skills in the real world.

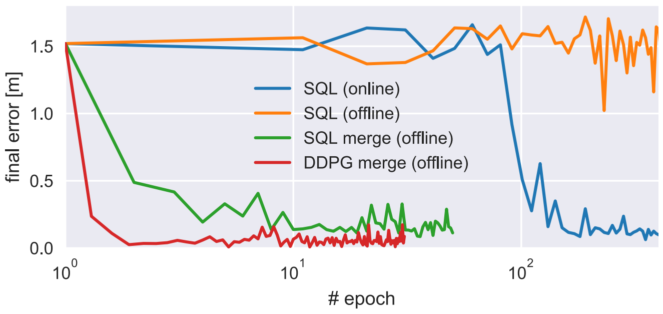

Fig. 3. Comparison of the number of iterations needed to extract a policy from a Q-function (red and green) and training a policy using off-policy (orange) or on-policy data (blue). Note that the x-axis is on a log scale. Our proposed method (green and red) extracts a policy from an additively composed Q-function, Q_Σ, almost instantly in an offline fashion from off-policy data that were collected for training the constituent Q-functions. In contrast, training a policy from scratch with online SQL (blue) takes orders of magnitude more gradient steps and requires collecting new experience from the environment. Lastly, training a policy on the composed task using offline SQL with pre-collected data fails to converge in a reasonable amount of time (orange).

C. *SQL for Real-World Manipulation*

To test the viability of deep RL with maximum entropy policies, we trained a Sawyer robot (Fig. 1) for reaching different locations and stacking Lego blocks. We also evaluated the compositionality of soft policies by combining the stacking policy with a policy that avoids an obstacle.

1) Reaching: Our first experiment explores how SQL compares with DDPG and NAF in terms of sample complexity. To that end, we trained the robot to move its end-effector to a specified target location in Cartesian space. Fig. 4 shows the learning curves for the three methods. Each experiment was repeated three times to analyze the variability of the methods. Since SQL does not rely on external exploration, we simply show training performance, whereas for DDPG and NAF, we show test performance at regular intervals, with the exploration noise switched off to provide a fair comparison. SQL solves the task in about 10 minutes, whereas DDPG and NAF are substantially slower, and exhibit large variation between training runs. We also trained a policy with SQL to reach randomly selected points that are provided as input to the policy, with a different target point selected for each episode. Learning this policy was almost as fast as training for a fixed target, but because some targets are easier to reach than others, the policy exhibited larger variation in terms of the final error of the end-effector position.

2) Lego Block Stacking: In the next experiment, we used SQL to train a policy for stacking Lego blocks to test its ability to exercise precise control in the presence of contact dynamics (Fig. 1, left). The goal is to position the end effector, which holds a Lego block, into a position and

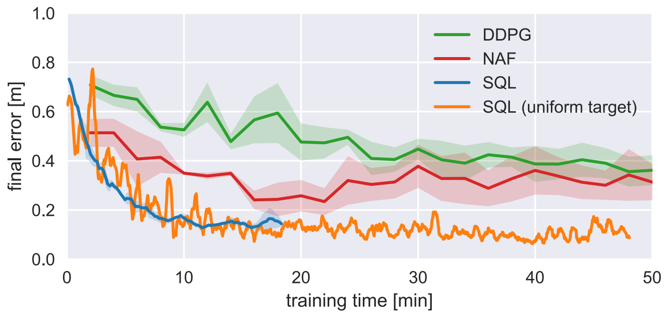

Fig. 4. The learning curve of DDPG (green), NAF (red), and SQL (blue) on the Sawyer robot when trained to move its end effector to a specific location. SQL learns much faster than the other methods. We also train SQL to reach randomly sampled end-effector locations by concatenating the desired location to the observation vector (orange). SQL learns to solve this task just as quickly. SQL curves show the moving average over 10 training episodes.

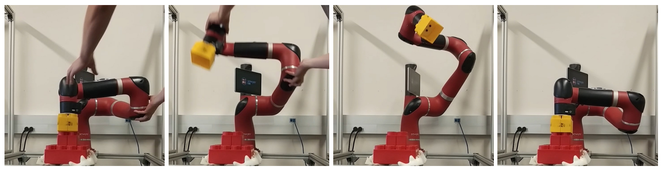

Fig. 5. A Lego stacking policy can be learned in less than two hours. The learned policy is remarkably robust against perturbations: the robot is able to recover and successfully stack the Lego blocks together after being pushed into a state that is substantially different from typical trajectories.

orientation that successfully stacks it on top of another block. To reduce variance in the end-effectors pose, we augmented the reward function with an additional negative log-distance term that incentivizes higher accuracy when the block is nearly inserted, and we also included a reward for downward force to overcome the friction forces. After half an hour of training, the robot was able to successfully insert the Lego block for the first time, and in two hours, the policy was fully converged. We evaluated the task visually by inspecting if all of the four studs were in contact and observed a success rate of 100 % on 20 trials of the final policy. The resulting policy does not only solve the task consistently, but it is also surprisingly robust to disturbances. To test robustness, we forced the arm into configurations that are far from the states that it encounters during normal execution, and the policy was able to recover every time and eventually solve the task (Fig. 5). We believe the robustness can be attributed to the effective and diverse exploration that is natural to maximum entropy policies, and, to our knowledge, no other prior deep RL work has shown similar results.

3) Composing Policies: In the last experiment, we evaluated compositionality on the Sawyer robot. We trained a new policy to avoid a fixed obstacle (Fig. 6, first row) and combined it with the policy to stack Lego blocks (Fig. 6, second row). The avoidance policy was rewarded for avoiding the obstacle and moving towards the target block without the shaping term that is required to successfully stack the two blocks. We then evaluated 1) the avoidance policy, 2) the stacking policy, and 3) the combined policy on the insertion task in the presence of the obstacle by executing each policy 10 times and counting the number of successful insertions. The avoidance policy was able to bring the Lego block close to the target block, but never successfully stacked the two blocks together. The stacking policy collided with the obstacle every time (Fig. 6, third row), but was still able to solve the task 30 % of the time. On the other hand, the combined policy (Fig. 6, bottom row) successfully avoided the obstacle and stacked the blocks 100 % of the time. This experiment illustrates that policy compositionality is an effective tool that can be successfully employed also to real-world tasks.

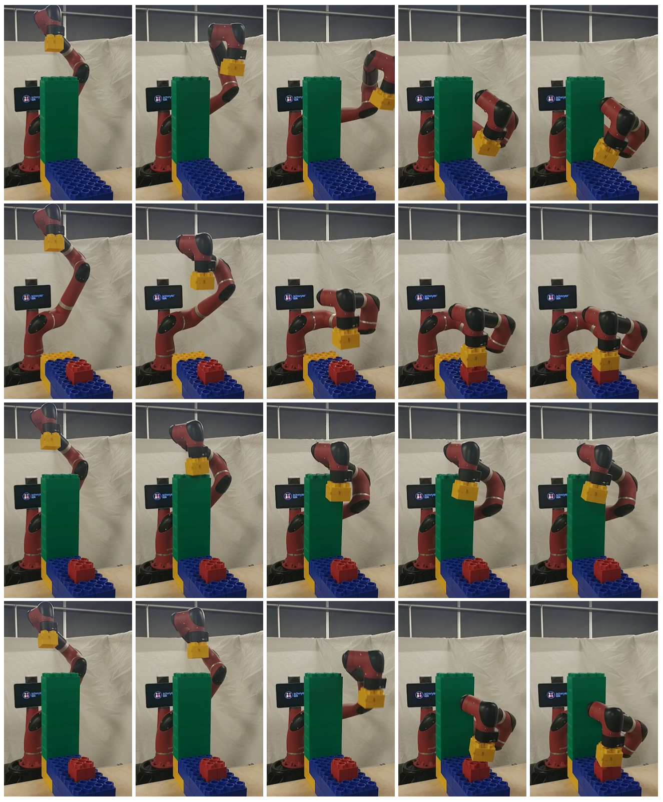

Fig. 6. To illustrate compositionality on physical hardware, we trained the Sawyer manipulator to avoid an obstacle (first row) and to stack Lego blocks together (second row). The stacking policy fails if it is executed in the presence of the obstacle (third row). However, by combining the two policies, we get a new combined skills that solves both tasks simultaneously and is able to stack the blocks while avoiding the obstacle.

VI. Discussion and Future Work

In this paper, we discuss how soft Q-learning can be extended to real-world robotic manipulation, both for learning individual manipulation tasks, and for learning constituent tasks that can then be composed into new policies. Our experiments show that soft Q-learning substantially outperforms prior model-free deep reinforcement learning. Soft Q-learning achieves substantially better performance than NAF on a simulated reaching task, including the case where multiple policies are composed to reach to new locations, and outperforms DDPG on a real-world reaching task evaluated on a Sawyer robot. The method exhibits better stability and convergence, and the ability to compose Q-functions obtained via soft Q-learning can make it particularly useful in real-world robotic scenarios, where retraining new policies for each new combination of reward factors is time-consuming and expensive.

In studying the composability of maximum entropy policies, we derive a bound on the error between the composed policy and the optimal policy for the composed reward function. This bound suggests that policies with higher entropy may be easier to compose. An interesting avenue for future work would be to further study the implications of this bound on compositionality. For example, can we derive a correction that can be applied to the composed Q-function to reduce bias? Answering such questions would make it more practical to construct new robotic skills out of previously trained building blocks, making it easier to endow robots with large repertoires of behaviors learned via reinforcement learning.

Acknowledgements. We thank Haoran Tang for suggestions on how to improve the proof and Abhishek Gupta for providing the simulated pusher environment. This work was supported by Berkeley DeepDrive.

References

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

[2] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *arXiv preprint arXiv:1611.03852*, 2016.

[3] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[4] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[5] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *arXiv preprint arXiv:1707.02267*, 2017.

[6] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI Conference on Artificial Intelligence*, 2008, pp. 1433–1438.

[7] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," *arXiv preprint arXiv:1702.08165*, 2017.

[8] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in neural information processing systems*, 2003, pp. 1547–1554.

[9] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[10] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 2219–2225.

[11] J. Peters and S. Schall, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–692, 2008.

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[13] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman, "Deep predictive policy training using reinforcement learning," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 2351–2358.

[14] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.

[15] F. Sadeghi and S. Levine, "(CAD)^{2}RL: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[16] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 23–30.

[17] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 5055–5065.

[18] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *CoRR*, vol. abs/1707.02267, 2017. [Online]. Available: http://arxiv.org/abs/1707.02267

[19] Q. Bateux, E. Marchand, J. Leitner, and F. Chaumette, "Visual servoing from deep neural networks," *CoRR*, vol. abs/1705.08940, 2017.

[20] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Advanced Robotics*, vol. 21, no. 13, pp. 1521–1544, 2007.

[21] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 763–768.

[22] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2397–2403.

[23] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Int. Conf. on Machine Learning*, 2016, pp. 2829–2838.

[24] E. Todorov, "General duality between optimal control and estimation," in *IEEE Conf. on Decision and Control*. IEEE, 2008, pp. 4286–4292.

[25] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," *Proceedings of Robotics: Science and Systems VIII*, 2012.

[26] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Int. Conf. on Machine Learning*. ACM, 2009, pp. 1049–1056.

[27] R. Fox, A. Pakman, and N. Tishby, "Taming the noise in reinforcement learning via soft updates," in *Conf. on Uncertainty in Artificial Intelligence*, 2016.

[28] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 2772–2782.

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

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

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

[32] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory," *Journal of Statistical Mechanics: Theory And Experiment*, vol. 2005, no. 11, p. P11011, 2005.

[33] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm," in *Advances In Neural Information Processing Systems*, 2016, pp. 2370–2378.

[34] D. Wang and Q. Liu, "Learning to draw samples: With application to amortized mle for generative adversarial learning," *arXiv preprint arXiv:1611.01722*, 2016.

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

[36] E. Todorov, "Compositionality of optimal control laws," in *Advances in Neural Information Processing Systems*, 2009, pp. 1856–1864.

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[38] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5026–5033.

APPENDIX

A. Proof of Lemma 1

Lemma 1: Let Q_1^* and Q_2^* be the soft Q-function of the optimal policies corresponding to reward functions r_1 and r_2, and define $Q_\Sigma \triangleq \frac{1}{2}(Q_1^* + Q_2^*)$. Then the optimal soft Q-function of the combined reward $r_\mathcal{C} \triangleq \frac{1}{2}(r_1 + r_2)$ satisfies

$$Q_\Sigma(\mathbf{s},\mathbf{a}) \geq Q_\mathcal{C}^*(\mathbf{s},\mathbf{a}) \geq Q_\Sigma(\mathbf{s},\mathbf{a}) - C^*(\mathbf{s},\mathbf{a}),\ \forall \mathbf{s} \in \mathcal{S}, \forall \mathbf{a} \in \mathcal{A}, \tag{11}$$

where C^* is the fixed point of

$$C(\mathbf{s},\mathbf{a}) \leftarrow \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\mathcal{D}_{\frac{1}{2}}\left(\pi_1^*(\,\cdot\,|\mathbf{s}') \parallel \pi_2^*(\,\cdot\,|\mathbf{s}')\right) + \max_{\mathbf{a}'\in\mathcal{A}} C(\mathbf{s}',\mathbf{a}')\right], \tag{12}$$

and $\mathcal{D}_{\frac{1}{2}}(\,\cdot \parallel \cdot\,)$ is the Rényi divergence of order $1/2$.

Proof: We will first prove the lower bound using induction. Let us define functions $Q^{(0)}(\mathbf{s},\mathbf{a}) \triangleq Q_\Sigma(\mathbf{s},\mathbf{a})$ and $C^{(0)}(\mathbf{s},\mathbf{a}) \triangleq 0$, and assume $Q^{(k)} \geq Q_\Sigma - C^{(k)}$ for some k. Note that this inequality is trivially satisfied when $k = 0$. Next, let us apply the soft Bellman backup at step k:

$$\begin{aligned}
Q^{(k+1)}(\mathbf{s},\mathbf{a}) &= r_\mathcal{C}(\mathbf{s},\mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\log \int_\mathcal{A} \exp Q^{(k)}(\mathbf{s}',\mathbf{a}')d\mathbf{a}'\right] && (13)\\
&\geq r_\mathcal{C}(\mathbf{s},\mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\log \int_\mathcal{A} \exp\left(\frac{1}{2}\left(Q_1^*(\mathbf{s}',\mathbf{a}') + Q_2^*(\mathbf{s}',\mathbf{a}')\right) - C^{(k)}(\mathbf{s}',\mathbf{a}')\right) d\mathbf{a}'\right] && (14)\\
&\geq r_\mathcal{C}(\mathbf{s},\mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\log \int_\mathcal{A} \exp \frac{1}{2}\left(Q_1^*(\mathbf{s}',\mathbf{a}') + Q_2^*(\mathbf{s}',\mathbf{a}')\right) d\mathbf{a}' - \max_{\mathbf{a}'} C^{(k)}(\mathbf{s}',\mathbf{a}')\right] && (15)\\
&= r_\mathcal{C}(\mathbf{s},\mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\frac{1}{2}\left(V_1^*(\mathbf{s}') + V_2^*(\mathbf{s}')\right) + \log \int_\mathcal{A} \sqrt{\pi_1^*(\mathbf{a}'|\mathbf{s}')\pi_2^*(\mathbf{a}'|\mathbf{s}')}d\mathbf{a}' - \max_{\mathbf{a}'} C^{(k)}(\mathbf{s}',\mathbf{a}')\right] && (16)\\
&= \frac{1}{2}\left(Q_1^*(\mathbf{s},\mathbf{a}) + Q_2^*(\mathbf{s},\mathbf{a})\right) + \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\log \int_\mathcal{A} \sqrt{\pi_1^*(\mathbf{a}'|\mathbf{s}')\pi_2^*(\mathbf{a}'|\mathbf{s}')}d\mathbf{a}' - \max_{\mathbf{a}'} C^{(k)}(\mathbf{s}',\mathbf{a}')\right] && (17)\\
&= Q_\Sigma(\mathbf{s},\mathbf{a}) - \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\frac{1}{2}\mathcal{D}_{\frac{1}{2}}\left(\pi_1^*(\,\cdot\,|\mathbf{s}') \parallel \pi_2^*(\,\cdot\,|\mathbf{s}')\right) + \max_{\mathbf{a}'} C^{(k)}(\mathbf{s}',\mathbf{a}')\right] && (18)\\
&= Q_\Sigma(\mathbf{s},\mathbf{a}) - C^{(k+1)}(\mathbf{s},\mathbf{a}) && (19)
\end{aligned}$$

The last form shows that the induction hypothesis is true for $k + 1$. Since soft Bellman iteration converges to the optimal soft Q-function from any bounded $Q^{(0)}$, at the limit, we will have

$$Q_\mathcal{C}^*(\mathbf{s},\mathbf{a}) \geq Q_\Sigma(\mathbf{s},\mathbf{a}) - C^*(\mathbf{s},\mathbf{a}), \tag{20}$$

where the value of C^* is the fixed point of the following recursion:

$$C^{(k+1)} = \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\frac{1}{2}\mathcal{D}_{\frac{1}{2}}\left(\pi_1^*(\,\cdot\,|\mathbf{s}') \parallel \pi_2^*(\,\cdot\,|\mathbf{s}')\right) + \max_{\mathbf{a}'} C^{(k)}(\mathbf{s}',\mathbf{a}')\right]. \tag{21}$$

The upper bound follows analogously by assuming $Q^{(k)} \leq Q_\Sigma$, defining $Q^{(0)} = Q_\Sigma$ and noting that the Rényi divergence is always positive. ■

Corollary 1: As a corollary of Lemma 1, we have

$$V_\Sigma(\mathbf{s}) \geq V_\mathcal{C}^*(\mathbf{s}) \geq V_\Sigma(\mathbf{s}) - \max_\mathbf{a} C^*(\mathbf{s},\mathbf{a}),\ \forall \mathbf{s} \in \mathcal{S}, \tag{22}$$

where $V_\Sigma(\mathbf{s}) = \log \int_\mathcal{A} \exp(Q(\mathbf{s},\mathbf{a}))d\mathbf{a}$

Proof: Take the “log-sum-exp” of (11). ■

B. Proof of Theorem 1

Theorem 1: With the definitions in Lemma 1, the value of π_Σ satisfies

$$Q_\mathcal{C}^{\pi_\Sigma}(\mathbf{s},\mathbf{a}) \geq Q_\mathcal{C}^*(\mathbf{s},\mathbf{a}) - D^*(\mathbf{s},\mathbf{a}), \tag{23}$$

where $D^*(\mathbf{s},\mathbf{a})$ is the fixed point of

$$D(\mathbf{s},\mathbf{a}) \leftarrow \gamma\, \mathbb{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[\mathbb{E}_{\mathbf{a}'\sim\pi_\Sigma(\mathbf{a}'|\mathbf{s}')}\left[C^*(\mathbf{s}',\mathbf{a}') + D(\mathbf{s}',\mathbf{a}')\right]\right]. \tag{24}$$

Proof: Let us define functions $Q^{(0)}(\mathbf{s},\mathbf{a}) \triangleq Q_\mathcal{C}^*(\mathbf{s},\mathbf{a})$ and $D^{(0)}(\mathbf{s},\mathbf{a}) \triangleq 0$, and assume $Q^{(k)}(\mathbf{s},\mathbf{a}) \geq Q_\mathcal{C}^*(\mathbf{s},\mathbf{a}) - D^{(k)}(\mathbf{s},\mathbf{a})$, which is trivially satisfied for $k = 0$. We will use induction to show the inequality holds for any k by applying "soft policy evaluation" (see [7]):

$$\begin{aligned}
Q^{(k+1)}(\mathbf{s}, \mathbf{a}) &= r_{\mathcal{C}}(\mathbf{s}, \mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[\mathbb{E}_{\mathbf{a}' \sim \pi_\Sigma(\mathbf{a}'|\mathbf{s}')} \left[Q^{(k)}(\mathbf{s}', \mathbf{a}') - \log \pi_\Sigma(\mathbf{a}'|\mathbf{s}') \right] \right] && (25)\\
&\geq r_{\mathcal{C}}(\mathbf{s}, \mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[\mathbb{E}_{\mathbf{a}' \sim \pi_\Sigma(\mathbf{a}'|\mathbf{s}')} \left[Q^*_{\mathcal{C}}(\mathbf{s}', \mathbf{a}') - D^{(k)}(\mathbf{s}', \mathbf{a}') - Q_\Sigma(\mathbf{s}', \mathbf{a}') + V_\Sigma(\mathbf{s}') \right] \right] && (26)\\
&\geq r_{\mathcal{C}}(\mathbf{s}, \mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[\mathbb{E}_{\mathbf{a}' \sim \pi_\Sigma(\mathbf{a}'|\mathbf{s}')} \left[Q^*_{\mathcal{C}}(\mathbf{s}', \mathbf{a}') - D^{(k)}(\mathbf{s}', \mathbf{a}') - Q^*_{\mathcal{C}}(\mathbf{s}', \mathbf{a}') - C^*(\mathbf{s}', \mathbf{a}') + V^*_{\mathcal{C}}(\mathbf{s}') \right] \right] && (27)\\
&\geq r_{\mathcal{C}}(\mathbf{s}, \mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[V^*_{\mathcal{C}}(\mathbf{s}') - \mathbb{E}_{\mathbf{a}' \sim \pi_\Sigma(\mathbf{a}'|\mathbf{s}')} \left[C^*(\mathbf{s}', \mathbf{a}') + D^{(k)}(\mathbf{s}', \mathbf{a}') \right] \right] && (28)\\
&= Q^*_{\mathcal{C}}(\mathbf{s}, \mathbf{a}) - \gamma\, \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[\mathbb{E}_{\mathbf{a}' \sim \pi_\Sigma(\mathbf{a}'|\mathbf{s}')} \left[C^*(\mathbf{s}', \mathbf{a}') + D^{(k)}(\mathbf{s}', \mathbf{a}') \right] \right] && (29)\\
&= Q^*_{\mathcal{C}}(\mathbf{s}, \mathbf{a}) - D^{(k+1)}(\mathbf{s}, \mathbf{a}). && (30)
\end{aligned}$$

The second inequality follows from Lemma 1 and Corollary 1. In the limit as $k \to \infty$, we have $Q^{\pi_\Sigma}_{\mathcal{C}}(\mathbf{s}, \mathbf{a}) \geq Q^*_{\mathcal{C}}(\mathbf{s}, \mathbf{a}) - D^*(\mathbf{s}, \mathbf{a})$ (see [37]). ■