

Learning Dexterous Manipulation from Suboptimal Experts

Rae Jeong
DeepMind
raejeong@google.com

Jost Tobias Springenberg
DeepMind
springenberg@google.com

Jackie Kay
DeepMind
kayj@google.com

Daniel Zheng
DeepMind
dhhzheng@google.com

Alexandre Galashov
DeepMind
agalashov@google.com

Nicolas Heess
DeepMind
heess@google.com

Francesco Nori
DeepMind
fnori@google.com

Abstract: Learning dexterous manipulation in high-dimensional state-action spaces is an important open challenge with exploration presenting a major bottleneck. Although in many cases the learning process could be guided by demonstrations or other suboptimal experts, current RL algorithms for continuous action spaces often fail to effectively utilize combinations of highly off-policy expert data and on-policy exploration data. As a solution, we introduce *Relative Entropy Q-Learning* (REQ), a simple policy iteration algorithm that combines ideas from successful offline and conventional RL algorithms. It represents the optimal policy via importance sampling from a learned prior and is well-suited to take advantage of mixed data distributions. We demonstrate experimentally that REQ outperforms several strong baselines on robotic manipulation tasks for which suboptimal experts are available. We show how suboptimal experts can be constructed effectively by composing simple waypoint tracking controllers, and we also show how learned primitives can be combined with waypoint controllers to obtain reference behaviors to bootstrap a complex manipulation task on a simulated bimanual robot with human-like hands. Finally, we show that REQ is also effective for general off-policy RL, offline RL, and RL from demonstrations. Videos and further materials are available at sites.google.com/view/rlfse.

Keywords: robotic manipulation, deep reinforcement learning, imitation learning

1 Introduction

In recent years, deep reinforcement learning (RL) algorithms have demonstrated increasing capabilities in solving complex robotic control problems both in simulation and on real robots. However, exploration remains a significant challenge for high-dimensional robotics tasks with sparse rewards – the prevalent setting in the domain of robotic manipulation. Crafting a shaped reward function for manipulation tasks is often highly non-trivial due to the characteristics of the desired behavior, which might require complex interactions with tools to accomplish the task. Constructing controllers that achieve partial task success and exhibit the desired behavior, on the other hand, is often possible with methods from classical robotics, such as defining waypoint tracking controllers for picking up tools in the environment. Ideally, our RL algorithms should be flexible enough to incorporate off-policy data generated by such a “suboptimal expert”. Indeed, recent work on deep RL from demonstrations (RLfD) [1, 2] as well as offline RL from fixed datasets [3, 4] has partially made progress in this direction. However, we find that current algorithms are ineffective in utilizing highly off-policy data generated from a suboptimal expert together with exploration data generated by their own policy.

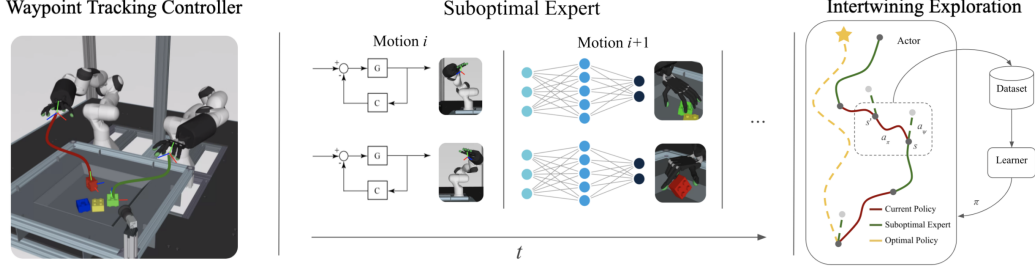


Figure 1: Left: Bimanual Shadow Hand LEGO stacking task with example waypoints. Middle: Motions are concatenated sequentially to construct a useful suboptimal expert. The robot arm end effector poses are controlled with waypoint tracking controllers, while the dexterous human-like hands are controlled with learned primitives. Right: The suboptimal experts’ actions can be intertwined with the current policy’s actions to achieve rich exploration. Note that it is also possible to record the actions that the suboptimal expert would have taken, shown as dotted lines.

In this work, we aim to develop an algorithm that excels in the setting of reinforcement learning from suboptimal experts (RLFSE) where we have direct access to an imperfect or partial task solution, as well as the ability to collect new data. To that end, we present three main contributions: 1) we develop a general policy iteration algorithm, *Relative Entropy Q-Learning* (REQ), which takes advantage of highly off-policy exploration data and shows strong performance across off-policy, offline RL, and RLfD; 2) we show that a simple exploration strategy which intertwines the policy’s actions with a suboptimal expert’s actions results in an effective exploration strategy for complex robotic manipulation tasks; 3) we demonstrate that suboptimal experts can be constructed by composing learned primitives and waypoint tracking controllers, allowing us to learn challenging dexterous manipulation task on a simulated bimanual robot with human-like hands.

2 Preliminaries

We consider the standard reinforcement learning setting in a Markov Decision Process (MDP) described by the tuple $(\mathcal{S}, \mathcal{A}, r, \gamma, P, \mu_0)$ consisting of: the state space \mathcal{S} , the action space \mathcal{A} , the instantaneous reward function $r(s_t, a_t)$ – obtained by executing action a_t in state s_t – the discount γ , the transition distribution $P(s_{t+1}|s_t, a_t)$ and the initial state distribution μ_0 . The RL objective \mathcal{J} corresponds to maximizing the expected reward under a given policy $\mathcal{J}(\pi) = \mathbb{E}_{\pi, P} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 \sim \mu_0(s)]$, where we have $a_t \sim \pi(\cdot | s_t)$ and $s_{t+1} \sim P(\cdot | s_t, a_t)$. We use $\pi^* = \arg \max_{\pi} \mathcal{J}(\pi)$ to denote the optimal policy. We make use of the state-action value function for policy π , also known as the Q-function which can be defined recursively as $Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{\pi, P} [Q^{\pi}(s_{t+1}, a_{t+1})]$, with which we can express \mathcal{J} in the alternate form $\mathcal{J}(\pi) = \mathbb{E}_{\pi, P} [Q^{\pi}(s_t, a_t) | s_0 \sim \mu_0(s)]$. Furthermore, we consider a policy iteration scheme [5] with additional KL constraints [6, 7, 8]. In this setting the goal is to maximize the Q-values while staying close to a reference (or prior) policy π^{prior} . In particular, given a current policy π_{i-1} that we want to improve upon, we can write the objective for iteration i as a constrained optimization problem, assuming access to a dataset \mathcal{D} :

$$\begin{aligned} \pi_i = \arg \max_{\pi} \mathcal{J}_c(\pi, \pi^{\text{prior}}, \epsilon) &= \arg \max_{\pi} \mathbb{E}_{s \sim \mathcal{D}} [\mathbb{E}_{a \sim \pi(\cdot | s)} [Q^{\pi_{i-1}}(a, s)]] \\ \text{s.t. } \forall s \in \mathcal{D} : \text{KL}(\pi(\cdot | s) || \pi^{\text{prior}}(\cdot | s)) &\leq \epsilon, \end{aligned} \quad (1)$$

where the policy to be optimized is denoted as $\pi(a|s)$, π^{prior} is the prior policy, and KL denotes the Kullback-Leibler divergence. Note that from the perspective presented here, π^{prior} is not necessarily a guess for an optimal policy but serves as the prior towards which we regularize.¹ Successively solving the optimization from Equation 1 for $i \in [1, N]$ – with interleaved policy evaluation to obtain Q^{π} – then constitutes the policy iteration loop.

¹Note that the per-state KL constraint is often relaxed to an average constraint over states from the buffer; e.g. in Nair et al. [4], Abdolmaleki et al. [7], Siegel et al. [9].

3 Relative Entropy Q-Learning

We introduce *Relative Entropy Q-Learning* (REQ). REQ is a policy iteration algorithm targeting the KL-constrained RL objective \mathcal{J}_c from Equation 1 in each iteration. We start by realizing that the solution to the KL-constrained objective at iteration i , $\pi_i = \arg \max_{\pi} \mathcal{J}_c(\pi, \pi_i^{\text{prior}}, \epsilon)$, can be obtained in closed form by formulating the Lagrangian of the constrained optimization problem and solving for π . The solution consists of a softmax over Q-values (a well known result, see e.g. [6, 7, 8, 9, 10]) $\pi_i(a|s) \propto \pi_i^{\text{prior}}(a|s) \exp(Q^{\pi_i-1}(s, a)/\eta_s)$ where the temperature η_s can be obtained by solving the dual function of \mathcal{J}_c ; a convex optimization problem. We refer to the Appendix A.3 for a derivation of the Lagrangian as well as how we optimize the dual function for η_s . Exactly calculating the normalization constant is intractable in continuous action spaces but we can, however, sample from $\pi_i(a|s)$ via importance weighting of samples from π_i^{prior} – an observation that we will now use to define the REQ policy evaluation step.

Policy Evaluation The first key observation is that we can learn the state-action value function of π_i without the need to explicitly represent π_i via a parametric policy. Instead, we can realize that the squared temporal difference error $(Q^{\pi_i}(s_t, a_t) - (r(s_t, a_t) + \gamma \mathbb{E}_{\pi_i}[Q^{\pi_i}(s_{t+1}, a_{t+1})]))^2$ can be evaluated using importance sampling, leading to the following objective:

$$Q_{\phi}^{\pi_i} = \arg \min_{Q_{\phi}} \mathbb{E}_{s, a, r, s' \in \mathcal{D}} \left[\left(Q_{\phi}(s, a) - \left[r + \gamma \sum_{j=1}^N \frac{\exp(Q_{\phi'}(s', a^j)/\eta_{s'})}{\sum_k \exp(Q_{\phi'}(s', a^k)/\eta_{s'})} Q_{\phi'}(s', a^j) \right] \right)^2 \right], \quad (2)$$

with $\forall j : a^j \sim \pi_i^{\text{prior}}(\cdot|s')$, where ϕ' denotes the parameters of a target network [11] and we estimate the expectation $\mathbb{E}_{\pi_i}[Q_{\phi}(s', a)] \propto \mathbb{E}_{a' \sim \pi_i^{\text{prior}}(\cdot|s')} [\exp(Q_{\phi}(s', a')/\eta_{s'}) Q_{\phi}(s', a)]$ with self-normalized importance sampling based on N samples.

A few observations can be made about this objective. The learned Q-function corresponds to the one considered in ABM+MPO [9], but with the difference that π_i is never projected onto a parametric policy. Instead it is only implicitly represented via importance sampling from the prior – we thus only need to learn π^{prior} and a Q-function. This can be beneficial when the prior is learned from data but is not well aligned with high-value regions in Q^{π_i} . In such a case π_i may become multimodal and hard to project to a parametric policy without accumulation of errors. Note that REQ can still represent the optimal policy, as for $\epsilon \rightarrow \infty$ the policy π_i will approach π^* . Hence, the constraint ϵ allows us to trade-off the exploitation of the Q-function and regularizing towards π^{prior} . An alternative view of the REQ policy evaluation is to consider it as a policy iteration algorithm that uses the the softmax Bellman operator [12, 13] for policy evaluation (with an adaptive method to satisfy a hard KL constraint with respect to a given prior). Further analysis of the REQ policy evaluation is provided in the Appendix A.2.

Prior Policy Improvement The Q-learning like algorithm from above can work with any prior as long as π^{prior} has probability mass everywhere; $\pi^{\text{prior}}(a|s) > 0 \forall a$. However, the sample based importance weighting scheme from Equation 2 becomes ineffective – potentially leading to learning Q-values of a suboptimal policy – if the number of samples N is low and the prior has small probability mass at actions with high Q-values. The policy improvement step of REQ thus is to learn an effective prior, π_i^{prior} , and improve it in each iteration. We achieve this by fitting the prior to all actions from the dataset, \mathcal{D} , whose value is estimated to be higher than the average value of the policy. Formally, we find

$$\pi_{i+1}^{\text{prior}} = \arg \max_{\pi_{\theta}^{\text{prior}}} \mathbb{E}_{a, s \sim \mathcal{D}} \left[1 \left[Q_{\phi'}^{\pi_i}(s, a) \geq \mathbb{E}_{a \sim \pi_i(\cdot|s)} [Q_{\phi'}^{\pi_i}(s, a)] \right] \log \pi_{\theta}^{\text{prior}}(a|s) \right], \quad (3)$$

where 1 is the indicator function. That is, we consider learning a prior similar to recent off-line RL algorithms such as ABM [9] and CRR-bin [3]. To avoid overfitting during training (e.g. due to suboptimal Q-values shrinking the prior distribution) we additionally regularize the prior update step. In particular, we employ constraints on the movement of the prior-policy mean ($\text{KL}(\pi_{i+1}^{\text{prior}}(a|s) \parallel \pi_i^{\text{prior}}(a|s; \Sigma = \Sigma_i) < \epsilon_{\Sigma})$ and the covariance ($\text{KL}(\pi_{i+1}^{\text{prior}}(a|s_j) \parallel \pi_i^{\text{prior}}(a|s; \mu = \mu_i) < \epsilon_{\mu})$ for a Gaussian prior – analogous to the trust-regions used for policy optimization in MPO [7], which we can enforce via a simple Lagrangian relaxation approach similar to MPO [8].

Practical algorithm A full listing of our procedure is presented in Algorithm 1. In contrast to previous works [7], we formulate a per-state KL constraint η_s that we optimize for each state in the batch; i.e. we perform multiple gradient steps on the dual for a given η_s to ensure the constraint is tight. In addition, instead of fully optimizing the policy and Q-function in each iteration, we switch to a new iteration after a fixed amount of gradient descent steps (via the use of target networks).

Algorithm 1 Relative Entropy Q-learning (REQ)

Input: number of learning steps N , steps between target updates U , number of action samples M , KL regularization parameter ϵ , initial parameters for θ, ϕ and η_s

def REQ_update(θ, π, \mathcal{B}):

// For this step let $\pi(a|s) \propto \pi_{\theta'}(a|s) \exp(Q_{\phi'}(a, s)/\eta_s)$ and $A^\pi(a, s) = Q_{\phi'}(a, s) - V^\pi(a, s)$

Find η_s for $s \in \mathcal{B}$ via gradient: $\nabla_{\eta_s} \frac{1}{|\mathcal{B}|} \eta_s \epsilon + \eta_s \log \frac{1}{M} \sum_{j=1}^M [\exp(Q_{\phi'}(s, a_j)/\eta_s) | a_j \sim \pi_{\theta'}(\cdot, s)]$

Compute $V^\pi(s) = \sum_{j=1}^M \frac{\exp(Q_{\phi'}(a_j, s)/\eta_s)}{\sum_{j=1}^M \exp(Q_{\phi'}(a_j, s)/\eta_s)} Q_{\phi'}(a_j, s)$, where $a_j \sim \pi_{\theta'}^{\text{prior}}(\cdot|s)$

Update Q-function with gradient: $\nabla_{\phi} \frac{1}{|\mathcal{B}|} \sum_{s, a, r, s' \in \mathcal{B}} \left(r + \gamma V^\pi(s') - Q_{\phi}(a, s) \right)^2$

Update prior π_{θ} with gradient: $\nabla_{\theta} - \frac{1}{|\mathcal{B}|} \sum_{s, a, r \in \mathcal{B}} 1[A^\pi(a, s) \geq 0] \log \pi_{\theta}(a|s)$

Initialize: $N = 0, \theta' = \theta, \phi' = \phi$

while $i \leq N$ **do**

Optionally: collect new data \mathcal{D}_i by following π_i or some mixture of π_i and an expert policy ψ

 Let $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$

 sample a batch \mathcal{B} from replay buffer \mathcal{D}

 execute **REQ_update**(θ, π, \mathcal{B})

 Update policy and Q-function every U steps by copying: $\theta' \leftarrow \theta, \phi' \leftarrow \phi$

end while

4 Reinforcement Learning from Suboptimal Experts

In this section we explain how our method can be used for reinforcement learning from suboptimal experts and describe a class of suboptimal experts for robotic manipulation consisting of simple waypoint tracking controllers.

Problem Formulation We consider an RL setting with additional access to a suboptimal expert $\psi(a|s)$. We assume that ψ exhibits behaviors that are relevant to the task but is not necessarily the optimal policy π^* . We refer to this setting as *Reinforcement Learning from Suboptimal Experts* (RLfSE). In RLfSE, our policy iteration scheme can be understood as a form of sample-based approximate policy iteration from mixed behavior data – somewhat similar to the AggreVaTe family of algorithms [14, 15]. This setting is motivated by real world problems for which domain-specific solutions are already deployed. RLfSE can give us access to broader data distributions than reinforcement learning from demonstrations (RLfD [1, 2]), as we can choose to collect data from ψ or a mixture of ψ and π . Additionally, direct access to ψ also allows us to label off-policy data with the expert’s actions, similar to commonly used no-regret imitation learning algorithms [14, 16, 17].

Waypoint Tracking Controllers As a concrete example, for each manipulation task, we construct a suboptimal expert $\psi(a|s)$ by composing waypoint (pose) tracking controllers. (Note ψ is deterministic in our case, thus we can write $a = \psi(s)$.) Such pose tracking controllers can be formulated by leveraging differential kinematics as well as velocity control modes of robotic arms. Using relative reference frames, these controllers can generalize under homogeneous transformations and provide an intuitive interface for humans to specify waypoints to follow. The pose controllers that we use are linear feedback controllers on the end-effector(s) of the robot arm(s) using velocity control. Formally we use $\psi(s) = [\psi_p(s), \psi_o(s)]$ for each controllable six degree of freedom with $\psi_p(s) = K_p e_p(s)$ and $\psi_o(s) = K_o e_o(s)$ where K_p and K_o are positive definite gain matrices. The position error is $e_p(s) = p_d(s) - p_t(s)$, where p_t and p_d are the measured and desired positions of the end-effector, respectively. We define the *orientation error* using unit quaternions where $\mathcal{Q}_d = \{\eta_d, \epsilon_d\}$ and $\mathcal{Q}_t = \{\eta_t, \epsilon_t\}$ represent the desired and measured orientations respectively, with η representing the real valued quaternion components and ϵ the imaginary values. We define the orientation error as $e_o(s) = \Delta\epsilon = \eta_t(s)\epsilon_d(s) - \eta_d(s)\epsilon_t(s) - \mathbf{S}(\epsilon_d(s))\epsilon_t(s)$ where $\mathbf{S}(\cdot)$ is the skew-symmetric operator [18]. Additional details of the composition of the waypoint tracking controller are presented in Appendix A.4.

Relative Entropy Q-learning from Suboptimal Experts To make use of the suboptimal expert ψ , we propose a simple exploration strategy: intertwining the execution of the current policy π with the execution of ψ . For each episode we first randomly choose with probability $\lambda_{\text{intertwine}}$ whether to execute a mix of the policy and the expert or either of the policy or the expert only. In episodes in which we choose actions according to a mixture of policy and expert we execute expert’s action with probability λ_{ψ} at every time step. Otherwise, we execute the entire episode with probability λ_{ψ} . This is illustrated in Figure 1 and the full procedure is presented in Appendix A.5 Algorithm 3. Note that setting $\lambda_{\text{intertwine}} = 0$ recovers the RLfD setting, where a certain portion of the data in the replay buffer are demonstrations from the suboptimal expert ψ . In addition to using ψ for data generation, we can also take advantage of access to the expert in the prior policy improvement step of REQ by using the following equation,

$$\pi_{i+1}^{\text{prior}} = \arg \max_{\pi_{\theta}} \mathbb{E}_{a,s \sim \mathcal{D}} \left[\left(1[A^q(a, s) \geq 0] + \mathbb{E}_{\bar{a} \sim \psi} [1[A^q(\bar{a}, s) \geq 0]] \right) \log \pi_{\theta}(a|s) \right], \quad (4)$$

where we now consider actions sampled from π and actions from the expert ψ for inclusion into the prior, by evaluating the suboptimal expert on states from \mathcal{D} . Note that since ψ is deterministic in our case, i.e. the expectation is over a delta distribution and can be evaluated using the single $\bar{a} = \psi(s)$.

5 Related Work

Reinforcement Learning from Demonstrations Our approach can be seen as a form of imitation learning from imperfect experts. Similar to DDPGfD [1], DQNfD [19] and MPOfD [20] we use off-policy data generated by experts in the replay buffer to augment reinforcement learning. However, our reinforcement learning from suboptimal experts setup allows us to both gather a mixture of data from the current policy and the suboptimal expert, as well as label any states with what the suboptimal expert would have done. Querying expert actions in states visited by the policy bears similarity to DAgger [16].

Off-Policy Reinforcement Learning When no experts are used and all data is collected by following the policy π , our algorithm bears similarity to a number of recent actor-critic and Q-learning algorithms. In the limit of $\epsilon \rightarrow \infty$ it can be seen as a form of deep Q-learning [11, 21] where the maximization of the Q-function is performed approximately via an amortized proposal – an idea that has previously been considered in the literature using a mixture of a uniform and a learned proposal fitted to the best current action for each state [22]. Our method can also directly be related to actor-critic algorithms that implement constrained policy updates – which have seen resurgence in recent years [6, 7, 8, 23, 24]. In particular REQ can be compared to MPO [7] which optimizes the objective from Equation 1, but uses an expected constraint and chooses the prior to be the policy π_{i-1} from the last iteration – i.e. it performs an EM-style policy optimization with a trust-region constraint. In addition MPO performs a projection step to obtain a parametric representation of π_i in each iteration, whereas REQ only learns the prior and represents the policy via importance sampling.

KL Regularized Reinforcement Learning Whereas REQ performs constrained updates with respect to the maximum reward objective, many recent algorithms directly optimize a regularized objective that trades off reward with the divergence to a fixed [25, 26, 27, 28] or learned [29, 30, 31] reference distribution. When the reference distribution is learned it plays a similar role to our prior. Learned reference distributions have so far mainly been considered for transfer and multi-task learning [29, 30, 31, 32].

Offline Reinforcement Learning Our algorithm can also be seen as a simplification of a recent offline RL algorithm, ABM+MPO [9] – where we only learn the prior and represent the optimal policy implicitly. While this may seem limiting, we find that removing the need for two policies – which also makes our algorithm closely related to [3] – results in better performance in practice. A similar effect was also observed in Novikov et al. [3], Nair et al. [4] where the authors found that CRR/AWAC outperformed ABM in high-dimensions. Note that the main difference between REQ and CRR is that we perform policy evaluation on the constrained optimal policy and treat the learned policy as the prior – CRR instead considers the behavior distribution (the dataset) as the prior. As the experiments will show, this allows us to obtain significantly better learning speed when new data can be collected.

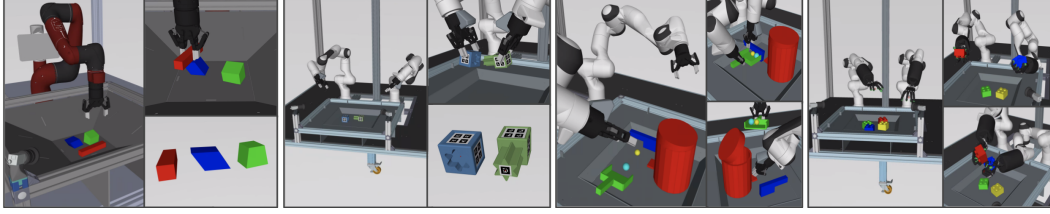


Figure 2: Simulated manipulation environments in the MuJoCo simulator [33]. From left to right: single arm stacking, bimanual insertion, bimanual cleanup and bimanual Shadow hand LEGO assembly. The environments are described in greater detail in Appendix A.1.

6 Experimental Results

We evaluate our algorithms, REQ and REQfSE, in several different reinforcement learning settings and demonstrate that the same algorithm performs well in all regimes. First, in section 6.1 we consider reinforcement learning from demonstrations (RLfD) and reinforcement learning from suboptimal experts (RLfSE) for the manipulation tasks shown in Figure 2. Next, in section 6.2, we evaluate REQ for (off-policy) reinforcement learning (on the DeepMind Control Suite benchmark [34]) as well as in a pure offline reinforcement learning setting. Finally, we demonstrate REQfSE’s potential to solve challenging dexterous manipulation tasks by controlling a simulated bimanual robot with human-like hands. Here, we construct a suboptimal expert by composing waypoint controllers with learned primitives (trained via REQ) which we then employ in the context of REQfSE. Hyperparameters for all algorithms can be found in the Appendix A.9.

6.1 Reinforcement Learning from Demonstrations and Suboptimal Experts

REQ bears similarity to both recent offline RL algorithms as well as to conventional policy iteration algorithms such as MPO. We thus hypothesize that REQ is well suited to take advantage of highly off-policy data but may also be able to make effective use of near on-policy data. We compare REQ to existing algorithms for learning from demonstrations in complex manipulation environments in both the RLfD and RLfSE settings. Specifically, we compare REQ to DDPGfD [1, 35] and MPOfD [20], which both make use of demonstration data that is added to the replay buffer. Additionally, we perform an ablation on the importance of the REQ policy evaluation by replacing the importance weighted formulation from Equation 2 with the standard TD(0) operator [5] which effectively recovers CRR-bin [3], a strong offline RL baseline, but with a trust-region in the policy improvement and without distributional Q-learning.² We refer to this as CRRfSE in our experiments. Figure 3 shows that especially in the bimanual environments, training with REQ provides a significant performance gain compared to other algorithms. We speculate that as the state-action space grows, using highly off-policy data for policy evaluation becomes significantly more challenging, similar to the difficulties encountered in offline RL with high state-action spaces. Given that most off-policy RL algorithms – unlike REQ or CRR – are designed to consider only the actions from their respective policy π for policy improvement, ignoring potentially good actions in the dataset can make them significantly less effective. This is evident in our results where CRR performs well compared to DDPG and MPO but significantly underperforms REQ. We highlight that CRR’s difference in speed can be solely attributed to the policy evaluation step: CRR performs policy evaluation with respect to what we consider to be the prior policy, but REQ can sharpen the prior to obtain better Q-values (and in turn a better policy), greatly improving learning speed.

RLfSE also significantly outperforms RLfD. We speculate that at the beginning of learning, there may be a large difference in the state distribution between the demonstrations and on-policy experience. This can make it difficult to exploit the expert actions for policy evaluation and policy improvement. Executing suboptimal experts from states visited by the policy (as happens during “intertwining”) makes expert’s action available for states close to the policy’s state distribution, thus rendering the data more effective. Intertwining also has the effect of broadening the state-visitation distribution, allowing for a more robust value learning. Furthermore, as expected, REQfSE signif-

²Note also that CRR is equivalent to AWAC [4] when exponential weighting rather than binary weighting is used, a setting that we found to not perform better in our experiments.

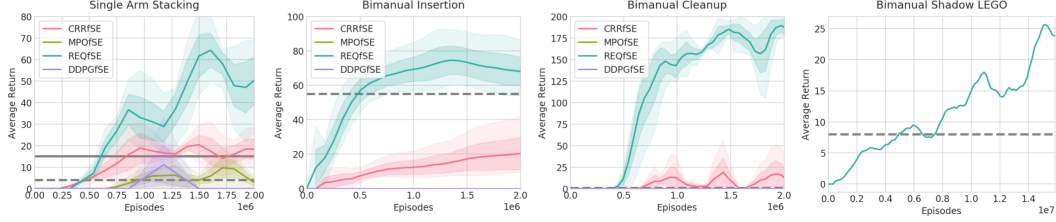


Figure 3: RLfD and RLfSE results on the tasks shown in Figure 2. The dotted line represents the performance of the suboptimal expert and the solid line shows the performance of the best RLfD agent over CRRfD, MPOfD, REQfD and DDPGfD. In both the bimanual insertion and the cleanup tasks the performance of RLfD agents is zero. For the bimanual Shadow LEGO task, we only evaluate REQfSE as this task is significantly harder than the first three. Additional comparisons of RLfSE, RLfD, BC, DAgger [16] and Residual RL [36] are available in the Appendix A.6.

icantly outperforms the suboptimal experts alone. Additional ablations of the REQ algorithm can be found in Appendix A.8. Finally, it is also worth observing the resulting behaviors of the policies optimized with REQfSE (footage available in the supplementary video). For example, the bimanual cleanup task shown in Figure 2 features two robotic arms in a workspace containing a trashcan, brush, dustpan and some balls. The goal is to pick up both the brush and the dustpan to sweep up the balls and place it in the trashcan. We find that when using a suboptimal expert with a low success rate to guide exploration (via intertwining) this difficult task can be learned successfully using a sparse reward that only indicates task success but provides no additional information, e.g. tool use. This suggests that REQfSE is well-suited to tasks where humans can provide additional cues for desired behavior or intermediary solution steps that are difficult to capture with a reward function.

6.2 Off-Policy and Offline Reinforcement Learning

To better understand the REQ algorithm, we evaluate it in the off-policy and offline RL settings. These settings can be interpreted as two ends of a spectrum: one end with access to data generated solely by the optimized policy, and the other where only offline data is available, with RLfSE and RLfD somewhere in the middle. Below we compare REQ to specialized algorithms for each setting. We hypothesize that in the standard off-policy RL setting REQ can perform significantly better than the closely related offline RL algorithm CRR. Recall that CRR considers the data itself to be the prior and hence performs policy evaluation for what we refer to as π^{prior} . In contrast REQ uses the optimal KL-constrained policy in each iteration to estimate the value (Equation 2). As a result, we would expect CRR to learn slowly when the data is bad or random. In contrast, REQ’s importance weighted policy can filter out the good actions for each state (subject to the KL constraint) and should result in evaluation of a better policy and thus faster learning. To test this hypothesis, we evaluate REQ and CRR as well as two state of the art off-policy RL algorithms (MPO [7] and SAC [26]) on the DeepMind Control Suite [34]. We again use the variant of CRR-bin [3] which differs from REQ only in the policy evaluation step. The results are shown in Figure 4. REQ’s policy evaluation indeed results in a significant improvement upon CRR. REQ’s performance is also comparable to state of the art off-policy RL algorithms, although it learns more slowly for the humanoid environments.

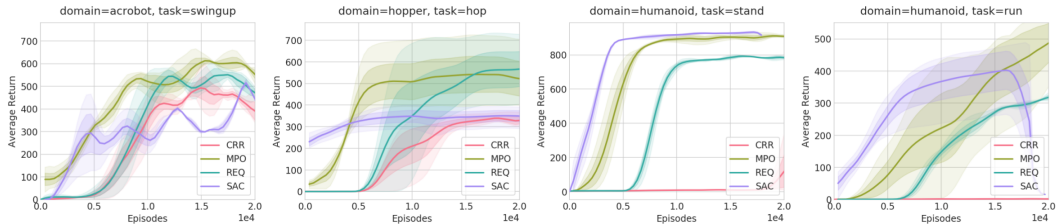


Figure 4: Online off-policy RL results on the DeepMind Control Suite [34]. MPO and SAC results matches [37] and [38] respectively. Note that SAC sometimes becomes unstable when run for longer.

	BC	D4PG	ABM	CRR-exp	CRR-bin	REQ	REQ-exp
Cartpole	386 \pm 6	855 \pm 13	798 \pm 30	664 \pm 22	860 \pm 7	855 \pm 10	741 \pm 29
Finger Turn	261 \pm 39	764 \pm 24	566 \pm 25	714 \pm 38	755 \pm 31	720 \pm 41	705 \pm 47
Walker Stand	386 \pm 6	929 \pm 46	689 \pm 13	797 \pm 30	881 \pm 13	901 \pm 27	810 \pm 17
Walker Walk	417 \pm 33	939 \pm 19	846 \pm 15	901 \pm 12	936 \pm 3	928 \pm 9	912 \pm 13
Cheetah Run	407 \pm 56	308 \pm 121	304 \pm 32	577 \pm 79	453 \pm 20	438 \pm 21	521 \pm 48
Fish Swim	466 \pm 8	281 \pm 77	527 \pm 19	517 \pm 21	585 \pm 23	592 \pm 41	586 \pm 36
Insert Ball	385 \pm 12	154 \pm 54	409 \pm 4	625 \pm 24	654 \pm 42	638 \pm 54	602 \pm 22
Insert Peg	324 \pm 31	71 \pm 2	345 \pm 12	387 \pm 36	365 \pm 28	397 \pm 41	389 \pm 21
Humanoid Run	382 \pm 2	1 \pm 1	302 \pm 6	586 \pm 6	412 \pm 10	408 \pm 24	596 \pm 15

Table 1: Offline RL results on the DeepMind Control Suite [34]. The reference numbers for baselines are from Novikov et al. [3].

We further hypothesize that REQ would also be competitive when we consider training from a fixed offline dataset. To this end, we compare REQ to the results of Novikov et al. [3] for the Control Suite tasks, using the same network architecture and evaluation criteria. As can be seen in Table 1, REQ performs comparably to CRR when the same advantage transformation is used (CRR-bin). Interestingly, the improved policy evaluation has less impact in this setting. We hypothesize that this is due to the fact that we need to stay close to the actions contained in the data to avoid problems with Q-value overestimation [3, 9]. When using another variant, REQ-exp, which uses the same exponential advantage transformation as CRR-exp, we again observe comparable performance. Overall, the results in off-policy and offline RL highlight the generality of the REQ algorithm. Although it does not completely outperform specialized algorithms for either setting, it simultaneously performs well in both. It also sheds light on important algorithmic components for learning efficiently and learning from highly off-policy data. The former is achieved by considering a KL-constrained optimal policy in the algorithm, like many of the algorithms for off-policy RL [7, 8]. The latter is achieved through considering the actions from the dataset, rather than the actions from the policy being optimized as proposed in recent offline RL algorithms [3, 9].

6.3 Reinforcement Learning from Composition of Learned Primitives and Controllers

Finally, to demonstrate REQ’s potential on challenging manipulation tasks, we evaluate REQfSE on the bimanual Shadow LEGO task. The goal in this task is to stack two LEGO blocks using a bimanual arm setup with two Shadow Hands. This problem is extremely challenging not only due to its high-dimensional state-action space (state size of 176 and action size of 52), but also because simple waypoint tracking controllers are not sufficient to provide reference behavior for the full task. Therefore, we first learn two primitive policies using REQ in an off-policy RL setting. The first primitive is trained to orient the green LEGO to an upwards facing position (while in the basket) with the left hand. The second primitive is trained to orient the red LEGO block to a downwards facing position with the right hand. Both primitives are trained “from-scratch” using a shaped reward function as described in the appendix A.1. We then construct a suboptimal “expert” for this task by composing the waypoint tracking controllers for reaching the blocks with the learned primitives to orient the blocks, followed by a third phase in which the LEGO blocks are joined together (again using simple waypoint tracking controllers). The results can be seen in Figure 3 where REQfSE is able to achieve higher performance than the suboptimal expert (footage available in the supplementary video).

7 Conclusion and Future Work

We have presented an approach for learning from suboptimal experts for complex dexterous robotic manipulation. We demonstrated that our algorithm, *Relative Entropy Q-Learning* (REQ), is effective in many different learning scenarios – including off-policy and offline RL as well as RL from demonstration (RLfD). In addition, our proposed approach for *Reinforcement Learning from Suboptimal Experts* (RLfSE), significantly outperforms competitive baselines. In particular, REQfSE leverages intertwining exploration to solve highly complex tasks that are otherwise intractable. To achieve this, we proposed using waypoint tracking controllers as the suboptimal experts. We argue that this approach can provide an intuitive and simple interface for humans to specify the desired behaviors for the task without requiring human demonstrations or engineering shaped reward functions.

Acknowledgments

We would like to thank Abbas Abdolmaleki, Akhil Raju, Antoine Laurens, Charles Game, Celine Smith, Christopher Schuster, Claudio Fantacci, Dave Barker, David Khosid, Emre Karagozler, Federico Casarini, Francesco Romano, Giulia Vezzani, Jon Scholz, Jose Enrique Chen, Michael Neunert, Murilo Martins, Nathan Batchelor, Nicole Hurley, Nimrod Gileadi, Nylda Adelise, Oleg Sushkov, Stefano Saliceti, Thomas Lampe, Thomas Rothörl, Tom Erez, Tuomas Haarnoja and Yuval Tassa for fruitful discussion and support.

References

- [1] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*, abs/1707.08817, 2017.
- [2] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2017.
- [3] A. Novikov, Z. Wang, K. Zolna, J. T. Springenberg, S. Reed, B. Shahriari, N. Siegel, C. Gulcehre, N. Heess, and N. de Freitas. Critic regularized regression. *In submission*, 2020.
- [4] A. Nair, M. Dalal, A. Gupta, and S. Levine. Accelerating online reinforcement learning with offline datasets, 2020.
- [5] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [6] J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2010.
- [7] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.
- [8] A. Abdolmaleki, J. T. Springenberg, J. Degraeve, S. Bohez, Y. Tassa, D. Belov, N. Heess, and M. Riedmiller. Relative entropy regularized policy iteration, 2018.
- [9] N. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=rke7geHtwH>.
- [10] K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. 07 2012. doi:10.15607/RSS.2012.VIII.045.
- [11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. ISSN 00280836. URL <http://dx.doi.org/10.1038/nature14236>.
- [12] Z. Song, R. E. Parr, and L. Carin. Revisiting the softmax bellman operator: New benefits and new perspective, 2018.
- [13] M. L. Littman. *Algorithms for Sequential Decision-Making*. PhD thesis, USA, 1996. AAI9709069.
- [14] S. Ross and J. A. Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *CoRR*, abs/1406.5979, 2014. URL <http://arxiv.org/abs/1406.5979>.
- [15] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

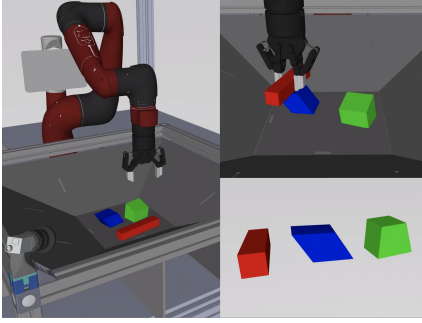
- [16] S. Ross, G. J. Gordon, and J. A. Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010.
- [17] K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daumé III, and J. Langford. Learning to search better than your teacher. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. URL <http://hal3.name/docs/#daume151ols>.
- [18] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008. ISBN 1846286417.
- [19] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.
- [20] R. Jeong, Y. Aytar, D. Khosid, Y. Zhou, J. Kay, T. Lampe, K. Bousmalis, and F. Nori. Self-supervised sim-to-real adaptation for visual robotic manipulation, 2019.
- [21] C. J. C. H. Watkins and P. Dayan. Q-learning. In *Machine Learning*, pages 279–292, 1992.
- [22] T. Van de Wiele, D. Warde-Farley, A. Mnih, and V. Mnih. Q-learning in enormous action spaces via amortized approximate maximization. *arXiv preprint arXiv:2001.08116*, 2020.
- [23] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.
- [24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.
- [25] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=rk07ZXZRB>.
- [26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning (ICML)*, 2018.
- [27] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. *CoRR*, abs/1702.08165, 2017. URL <http://arxiv.org/abs/1702.08165>.
- [28] J. Hunt, A. Barreto, T. Lillicrap, and N. Heess. Composing entropic policies using divergence correction. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2911–2920, 2019.
- [29] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. *CoRR*, abs/1707.04175, 2017. URL <http://arxiv.org/abs/1707.04175>.
- [30] A. Galashov, S. Jayakumar, L. Hasenclever, D. Tirumala, J. Schwarz, G. Desjardins, W. M. Czarnecki, Y. W. Teh, R. Pascanu, and N. Heess. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations*, 2019.
- [31] D. Tirumala, H. Noh, A. Galashov, L. Hasenclever, A. Ahuja, G. Wayne, R. Pascanu, Y. W. Teh, and N. Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl, 2019.
- [32] A. Galashov, J. Sygnowski, G. Desjardins, J. Humplik, L. Hasenclever, R. Jeong, Y. W. Teh, and N. Heess. Importance weighted policy learning and adaption, 2020.
- [33] 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*, pages 5026–5033. IEEE, 2012.
- [34] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. Deepmind control suite, 2018.

- [35] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. *CoRR*, abs/1806.07851, 2018.
- [36] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control, 2018.
- [37] M. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, F. Behbahani, T. Norman, A. Abdolmaleki, A. Cassirer, F. Yang, K. Baumli, S. Henderson, A. Novikov, S. G. Colmenarejo, S. Cabi, C. Gulcehre, T. L. Paine, A. Cowie, Z. Wang, B. Piot, and N. de Freitas. Acme: A research framework for distributed reinforcement learning, 2020.
- [38] D. Yarats and I. Kostrikov. Soft actor-critic (sac) implementation in pytorch. https://github.com/denisyarats/pytorch_sac, 2020.

A Appendix

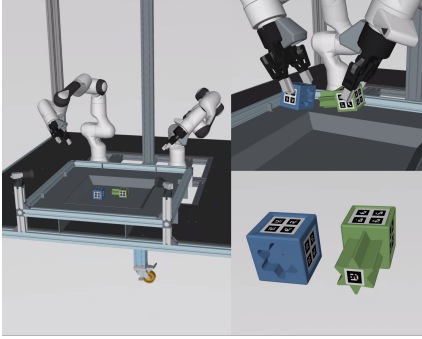
A.1 Environments

In this section, we will provide the details of the simulated environments used in our experiments. All of our simulated environments are modelled using the MuJoCo simulator [33].



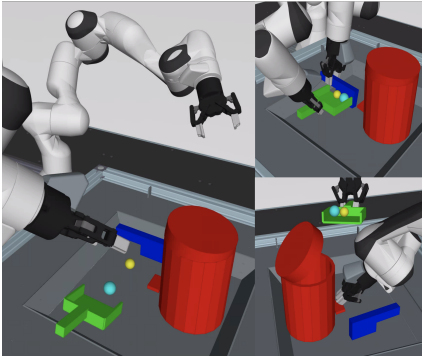
Single Arm Stacking (State Size: 69, Action Size: 5)

This environment features a 7 DOF arm that is controlled with a Cartesian velocity controller with the orientation of the end-effector always pointing down, reducing the action size to translation in X, Y, Z, rotation about the Z axis and the gripper control. The goal of the task is to stack the red object on top of the blue object. The reward is sparse and active when the red object is on the blue object without touching the gripper, green object and the basket.



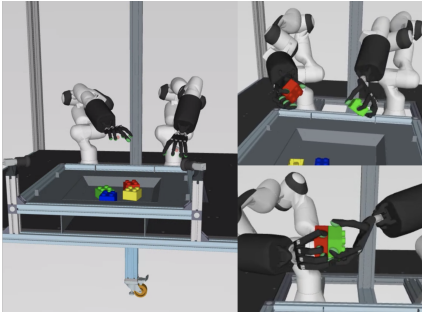
Bimanual Insertion (State Size: 86, Action Size: 14)

This environment features two 7 DOF arms that are each controlled using 6D Cartesian velocity controllers. Each arm has a 1 DOF gripper attached to it. The goal of the task is to insert the star shaped peg of the green object to star shaped hole of the blue object. The reward is zero when either of the objects' height from the basket is less than 0.15 meters, otherwise the reward is a function of the norm of the distance between the tip of the peg and the hole $dx = \|x_{peg} - x_{hole}\|_2$, where $r(dx) = \exp(-(\alpha * dx)^2)$ with $\alpha = 40$. If $r(dx) > 0.95$ it is rounded up to 1.



Bimanual Cleanup (State Size: 129, Action Size: 14)

This environment features two 7 DOF arms that are each controlled using 6D Cartesian velocity controllers. Each arm has a 1 DOF gripper attached to it. The goal of the task is to place the two balls inside the trashcan. The reward is 0.5 if either one of the balls are inside the trashcan and 1 if both balls are in the trashcan. Although not specified in the reward function, the desired behavior is for the robot to use both the brush and the dustpan to collect the balls and use the pedal to open the lid of the trashcan then pour in the balls in the trashcan.



Bimanual Shadow (State Size: 176, Action Size: 52)

This environment features two 7 DOF arms that are each controlled using 6D Cartesian velocity controllers. Each arm has a 20 DOA/24 DOF hand attached to it. The goal of the task is to assemble the red and the green block together. The reward is zero when either of the blocks' height from the basket is less than 0.15 meters, otherwise the reward function is shaped to minimize the distance between the top of the red block to the bottom of the green block using the reward tolerance function from Tassa et al. [34].

All of the tasks use the MuJoCo physics state [33] as the observation. Each of the tasks have been chosen to probe certain types of challenges that are widely discussed in the robotics research field. The single arm stacking task is not only a widely used benchmark for robot learning research, but it also provides a challenging RL task with its simple sparse reward and wide initial state distribution. On the other hand, the bimanual insertion task aims to focus on precision, using two controllable arms, similar to how humans would perform the insertion task using their two arms. Hence the bimanual insertion task has a small stochastic initial state distribution with the peg and hole placed on right and left side of the basket respectively. The bimanual cleanup task focuses on high dimensional state space and tool use, hence it also has a small stochastic initial state distribution. The bimanual shadow task aims to bring different types of complexity into a single task. It has a wide stochastic initial state distribution with the LEGO blocks starting anywhere on the basket with random orientations. The goal of assembling the LEGO blocks requires significant amount of precision and it has the largest state-action dimension with its human-like Shadow Hands.

A.2 Theoretical Analysis of Relative Entropy Q-Learning

TD(0) Operator as a special case of REQ It is easy to see that REQ performs policy evaluation for π^{prior} when $\epsilon = 0$, as the only valid solution (up to differences which occur with zero probability) to the constrained optimization problem in Equation 1 is $\pi = \pi^{\text{prior}}$. This is due to the Gibbs' inequality, where $\text{KL}(P\|Q) = 0$ if and only if $P = Q$ almost everywhere.

Q-Learning as a special case of REQ Let us consider an MDP with a discrete action space with π^{prior} having probability mass everywhere; $\pi^{\text{prior}}(a|s) > 0 \quad \forall a$. As $\epsilon \rightarrow \infty$ the solution to the constrained optimization problem in Equation 1 approaches the greedy policy because the KL constraint is effectively removed, in which case the REQ operator is similar to that of Q-learning or DQN. It is worth noting that even when π is $\arg \max_a Q(s, a)$, the KL divergence is still finite due to the asymmetry of KL divergence, hence, as $\epsilon \rightarrow \infty$, π can exactly represent $\arg \max_a Q(s, a)$. In practice, when applied to a continuous action space, the maximization over action samples approximates the maximization over the action space.

Policy Iteration Perspective of REQ One way to analyze the REQ algorithm is as a simple policy iteration by considering the target Q-function to be fixed during the policy evaluation step. Specifically, the optimization in Equation 1 is with respect to the target Q-function and hence the resulting policy is not conditioned on the current Q-function. In addition to the target Q-function for policy π , one can decide to use an additional target Q-function for the next state value estimate [11], although we find that in practice we are able to use the same target Q-function for both purposes, as described in Equation 2. With the fixed Q-function for policy π , the contraction of the policy evaluation is obvious as its a simple TD(0) operator with respect to π .

When target Q-function is not used, then it bears strong similarity with the softmax Bellman operator discussed by Littman [13] and Song et al. [12]. With the main difference being that the temperature parameter η_s in the REQ case is optimized adaptively to satisfy the KL constraint with respect to a prior policy π^{prior} whereas the softmax Bellman operator uses a constant temperature value. It is worth noting that the softmax transform is not a non-expansion as shown by Littman [13]. Surprisingly, despite its property, the softmax Bellman operator has been highly effectively shown by Song et al. [12] as well as our own empirical results. In practice, we find that using a fixed epsilon with target networks is sufficient for good performance.

A.3 KL constrained RL objective (closely following Abdolmaleki et al. [8])

Let us consider the following constrained optimization problem for a given $s \in \mathcal{S}$, analogous to Equation 1 in the main text but considering the optimization problem $\forall s \in \mathcal{S}$ given π^{prior} and ϵ :

$$\begin{aligned} & \max_{\pi} \int_{\mathcal{A}} \pi(a|s) Q^{\pi}(a, s) da \\ \text{s.t.} \quad & \text{KL}(\pi(\cdot|s) \| \pi^{\text{prior}}(\cdot|s)) \leq \epsilon, \\ & \int_{\mathcal{A}} \pi(a|s) da = 1 \end{aligned} \tag{5}$$

where the Lagrangian is,

$$L(\pi, \eta_s, \lambda_s) = \int_{\mathcal{A}} \pi(a|s) Q^\pi(a, s) da + \eta_s \left(\epsilon - \int_{\mathcal{A}} \pi(a|s) \log \frac{\pi(a|s)}{\pi^{\text{prior}}(a|s)} da \right) + \lambda_s \left(1 - \int_{\mathcal{A}} \pi(a|s) da \right) \quad (6)$$

Differentiating with respect to π yields,

$$\frac{\partial L}{\partial \pi} = Q^\pi(a, s) - \eta_s \log \pi(a|s) + \eta_s \log \pi^{\text{prior}}(a|s) - (\eta_s + \lambda_s) \quad (7)$$

where setting it to zero and rearranging terms to solve for π we get,

$$\pi(a|s) = \pi^{\text{prior}} \exp \left(\frac{Q^\pi(a, s)}{\eta_s} \right) \exp \left(- \frac{\eta_s + \lambda_s}{\eta_s} \right) \quad (8)$$

It is easy to see that when $\int_{\mathcal{A}} \pi(a|s) da = 1$ is satisfied, $\pi(a|s)$ is a valid probability distribution, hence, the last term in Equation 8 is the normalizing constant for π , and we can write,

$$\begin{aligned} \exp \left(\frac{\eta_s + \lambda_s}{\eta_s} \right) &= \int_{\mathcal{A}} \pi^{\text{prior}} \exp \left(\frac{Q^\pi(a, s)}{\eta_s} \right) da \\ \frac{\eta_s + \lambda_s}{\eta_s} &= \log \left(\int_{\mathcal{A}} \pi^{\text{prior}} \exp \left(\frac{Q^\pi(a, s)}{\eta_s} \right) da \right) \end{aligned} \quad (9)$$

The dual function $g(\eta_s)$ can be obtained by inserting above into the Lagrangian,

$$\begin{aligned} L(\pi, \eta_s, \lambda_s) &= \int_{\mathcal{A}} \pi(a|s) Q^\pi(a, s) da \\ &\quad - \eta_s \int_{\mathcal{A}} \pi(a|s) \left(\frac{Q^\pi(a, s)}{\eta_s} + \log \pi^{\text{prior}}(a|s) - \frac{\eta_s + \lambda_s}{\eta_s} \right) da + \eta_s \epsilon \\ &\quad + \eta_s \int_{\mathcal{A}} \pi(a|s) \log \pi^{\text{prior}}(a|s) da + \lambda_s \left(1 - \int_{\mathcal{A}} \pi(a|s) da \right) \end{aligned} \quad (10)$$

where simplifying yields,

$$L(\pi, \eta_s, \lambda_s) = \eta_s \epsilon + \eta_s \frac{\eta_s + \lambda_s}{\eta_s} \quad (11)$$

and plugging in Equation 9 gives the dual function for the Lagrangian multiplier η_s ,

$$g(\eta_s) = \eta_s \epsilon + \eta_s \log \left(\int_{\mathcal{A}} \pi^{\text{prior}} \exp \left(\frac{Q^\pi(a, s)}{\eta_s} \right) da \right). \quad (12)$$

We optimize the dual function $g(\eta_s)$ using multiple steps of gradient decent. The initial value of the temperature is set to the mean of the standard deviation over the number of action samples of the Q values,

$$\eta_0 = \frac{1}{N} \sum_n \sigma_M(Q(s, a_{nm})) \quad (13)$$

where N is size of the mini batch, n its and the M is the number of action samples, m its index. Additionally the gradient is scaled using the current value of the temperature, to ensure reliable convergence for arbitrary set of Q values.

A.4 Waypoint Tracking Controllers as Suboptimal Expert

The feedback linear controllers described in [section 4](#) are concatenated sequentially (optionally with learnt primitives) to construct suboptimal experts. A full listing of our procedure is presented in [Algorithm 2](#).

Algorithm 2 Waypoint Tracking Controllers as Suboptimal Expert

Input: State s , N sequence of motions. Initialize index for motion $i = 0$, Motion is a tuple of (action \bar{a} , M target frames \mathcal{F} , optional learned controller ψ_π , optional jump function J).

```

 $a_i \leftarrow \bar{a}$ 
while  $j \leq M$  do
  set position velocity action  $a_{i,j,p} \leftarrow \psi_p(s, \mathcal{F}_j)$ 
  set orientation velocity action  $a_{i,j,o} \leftarrow \psi_o(s, \mathcal{F}_j)$ 
end while
if  $\psi_\pi(s)$  then
   $a_i \leftarrow \psi_\pi(s)$ 
end if
if timeout then
   $i \leftarrow i + 1$ 
end if
if  $J$  then
   $i \leftarrow J(s)$ 
end if
return  $a_i$ 

```

Single Arm Stacking While using action \bar{a} to specify the gripper to be open, the end-effector of the single arm is taken above the frame of the red object and lowered. The action \bar{a} is updated to close the gripper. The end-effector is taken above the blue object. The action \bar{a} is then updated to open the gripper.

Bimanual Insertion Both of the grippers are set to open in \bar{a} and the two end-effectors are taken to the peg and the hole object. Both of the grippers are set to close in \bar{a} . The end-effectors are lifted up and the peg and the hole are brought close together.

Bimanual Cleanup Both of the grippers are set to open in \bar{a} and the two end-effectors are taken to the brush and the dustpan object. Both of the grippers are set to close in \bar{a} . The end-effectors brought close together while staying close to the basket in attempt to lift up the two balls. The end-effectors lifted. The left end-effector is taken to pedal of the trashcan and lowered. The right end-effector is taken above the trashcan and the dustpan is tilted down towards the opening trashcan.

Bimanual Shadow LEGO The bimanual Shadow Hand LEGO stacking task is the only task which uses learned primitives. Two learned primitives are trained using standard off-policy RL with REQ. The first primitive is trained to orient the green LEGO to an upwards facing position with the left hand. The second primitive is trained to orient the red LEGO block to a downwards facing position with the right hand. In both cases the block must remain in the basket. The orient tasks uses a shaped reward as a function of the orientation error defined in [section 4](#) with respect to the closest target orientation (taking into account the four symmetric orientations of the LEGO blocks). At the start of the task, both of the Shadow Hands are set to open in \bar{a} and the right hand is taken above the red LEGO block. The learned primitive for the right hand executes in order to orient the red LEGO block to a downwards facing position. The right hand is updated to close in \bar{a} and the right hand is lifted. If the red LEGO block is not lifted, it retries using the jump function J . The green LEGO block is lifted in a similar way using the left hand. The red and the green LEGO blocks are then aligned and brought closer together.

A.5 Intertwining Exploration

Algorithm 3 Intertwining Exploration

Input: policy π , suboptimal policy ψ , number of episodes N , λ_ψ and $\lambda_{\text{intertwine}}$.

```

while  $i \leq N$  do
  Reset the environment
   $\text{should\_intertwine} = \text{uniform}(0, 1) < \lambda_{\text{intertwine}}$ 
  while episode not terminated do
    if  $\text{should\_intertwine}$  then
       $\text{act\_with\_}\psi = \text{uniform}(0, 1) < \lambda_\psi$ 
    end if
     $a = \begin{cases} a \sim \pi(a|s) & \text{if } \text{act\_with\_}\psi \\ a \sim \psi(a|s) & \text{else,} \end{cases}$ 
    Act in the environment with  $a$ 
  end while
end while

```

A.6 RLfD, BC and DAgger Results

The results in Figure 5 clearly shows that RLfD, BC and DAgger struggling to learn the manipulation tasks especially for the bimanual tasks. Overall, the results for all the algorithms in Figure 5 is significantly less than the results of our REQfSE algorithm in Figure 3. It is worth noting that the we only use MLP networks for these three tasks and that the suboptimal expert has an internal counter, which could explain the ineffectiveness of pure imitation learning algorithms, BC and DAgger.

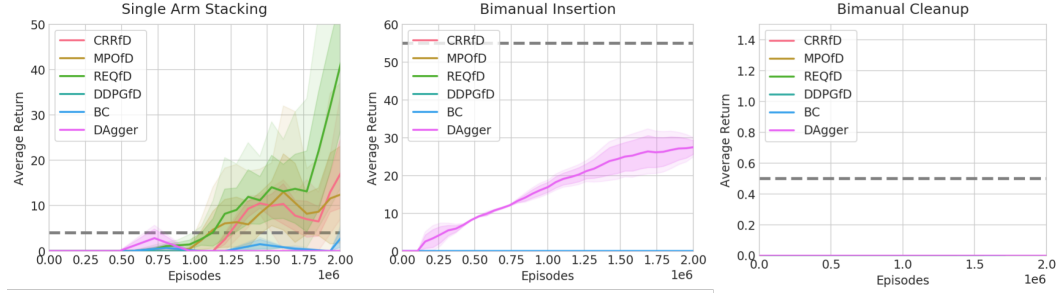


Figure 5: RLfD results on the tasks shown in Figure 2. The dotted line represents the performance of the suboptimal expert. Note that the DAgger refers to using the same intertwining exploration for RLfSE but optimizing the negative log-likelihood of the actions from the suboptimal expert.

A.7 Offline Reinforcement Learning Details

As described in the main text, the offline RL results use the same network architecture, same dataset and the evaluation criteria as Novikov et al. [3] to ensure a controlled and fair experiments.

A.8 Ablation Studies

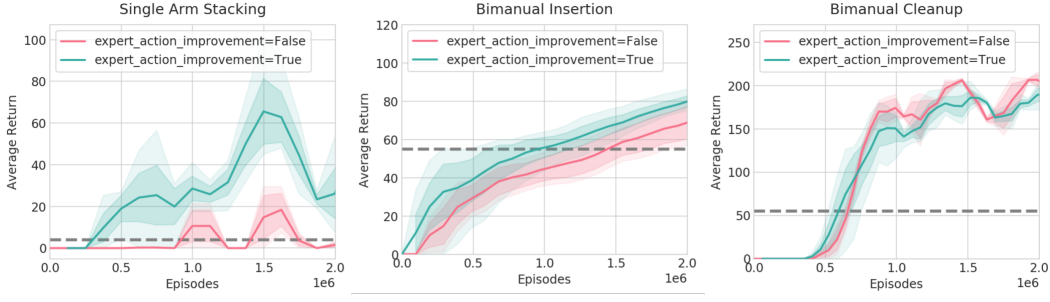


Figure 6: Ablation on using the *expert_action_improvement* described in Equation 3.

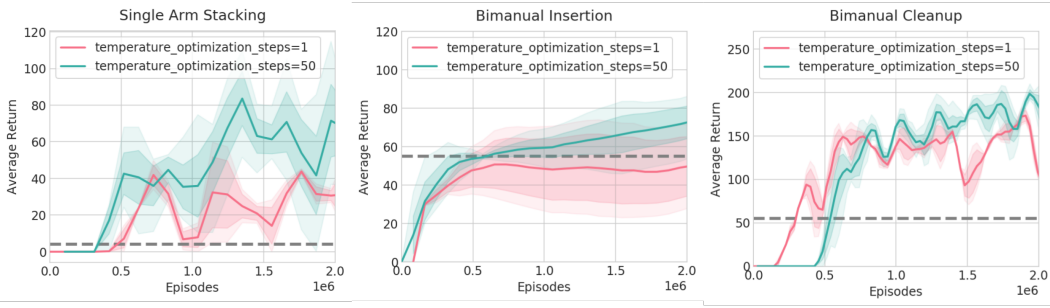


Figure 7: Ablation on number of gradient steps for the temperature described in section 3.

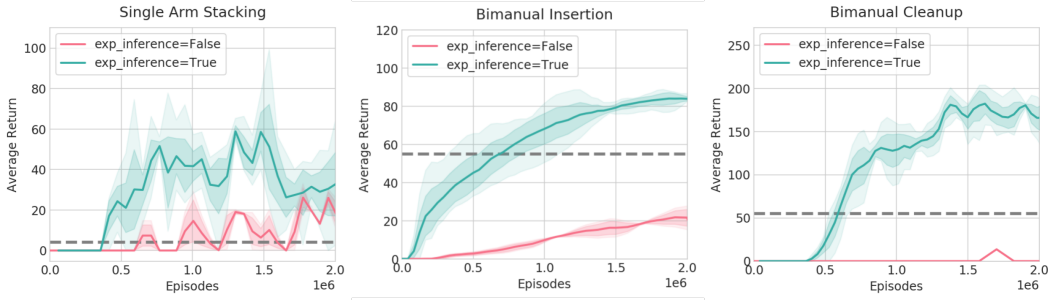


Figure 8: Ablation of using the exponential transform during inference or using the prior policy for acting described in section 3.

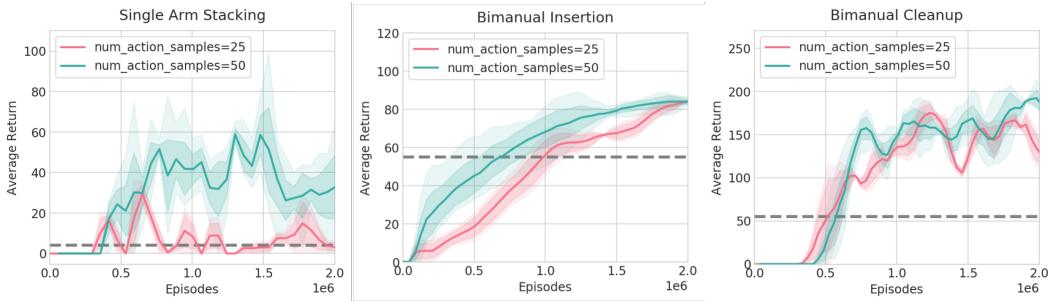


Figure 9: Ablation on number of action samples for value estimate and importance sampling described in section 3.



Figure 10: Ablation on the probability of acting with the suboptimal expert described in section 3.

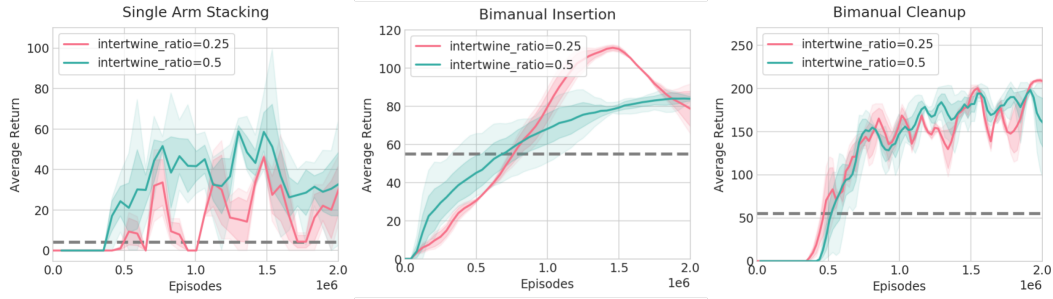


Figure 11: Ablation on the intertwining exploration probability described in section 3.

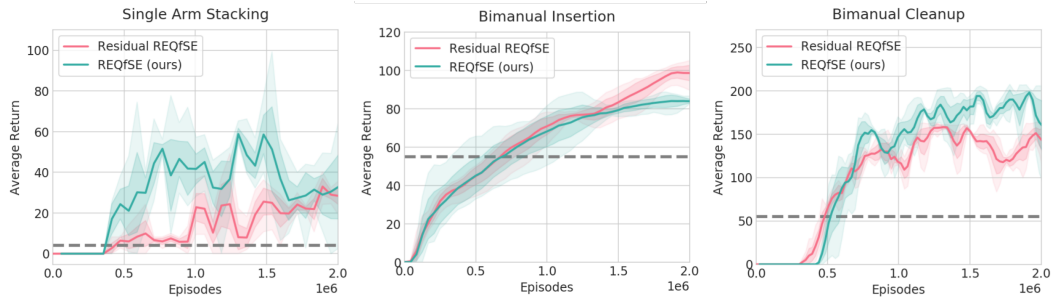


Figure 12: Comparison to a setup similar to that of residual reinforcement learning [36] for RE-QfSE, where the action of the suboptimal expert is provided as an input to agent.

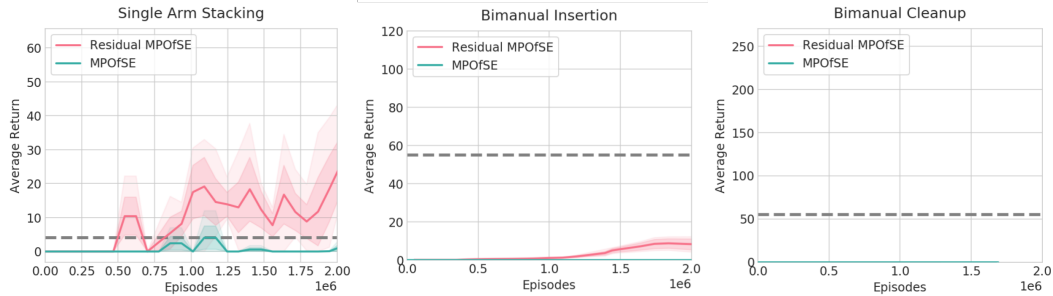


Figure 13: Comparison to a setup similar to that of residual reinforcement learning [36] for MPOfSE, where the action of the suboptimal expert is provided as an input to agent.

A.9 Hyperparameters

Neural networks are assumed to be MLP unless stated otherwise.³

Hyperparameters	CRR REQ
Policy net	256-256-256
Q function net	256-256-256
Discount factor (γ)	0.99
Adam learning rate	0.0003
Replay buffer size	1000000
Target network update period	20
Batch size	512
Unroll length	2
Activation function	elu
Layer norm on first layer	Yes
Min variance	0.00001
Max variance	unbounded
Number of action samples	20
REQ epsilon (ϵ)	0.75
Number of gradient decent steps for η_s	20
Epsilon mean (ϵ_μ)	0.01
Epsilon covariance (ϵ_μ)	0.00001
Policy evaluation method	TD(0) Operator REQ Operator

Table 2: Hyperparameters for CRR and REQ

Hyperparameters	DDPGfD and DDPGfSE
Policy net	256-256-256
Q function net	256-256-256
Discount factor (γ)	0.99
Adam learning rate	0.0001
Replay buffer size	1000000
Target network update period	200
Batch size	128
Unroll length	32
Activation function	elu
Layer norm on first layer	Yes
Action noise standard deviation	0.3

Table 3: Hyperparameters for DDPGfD and DDPGfSE

³The bimanual shadow task with REQfSE used slightly different hyperparameters with $\lambda_\psi = 0.5$ and using an LSTM instead of the middle MLP.

Hyperparameters	MPOfD and MPOfSE
Policy net	256-256-256
Q function net	256-256-256
Discount factor (γ)	0.99
Adam learning rate	0.0001
Replay buffer size	1000000
Target network update period	200
Batch size	128
Unroll length	32
Activation function	elu
Layer norm on first layer	Yes
Min variance	0.00001
Max variance	unbounded
Number of action samples	20
MPO epsilon	0.1
Epsilon mean (ϵ_μ)	0.001
Epsilon covariance (ϵ_μ)	0.00001

Table 4: Hyperparameters for MPOfD and MPOfSE

Hyperparameters	CRRfSE and CRRfSE REQfD and REQfSE
Policy net	256-256-256
Q function net	256-256-256
Discount factor (γ)	0.99
Adam learning rate	0.0001
Replay buffer size	1000000
Target network update period	500
Batch size	128
Unroll length	32
Activation function	elu
Layer norm on first layer	Yes
Min variance	0.00001
Max variance	unbounded
Number of action samples	50
REQ epsilon (ϵ)	0.75
Number of gradient decent steps for η_s	50
Epsilon mean (ϵ_μ)	0.005
Epsilon covariance (ϵ_μ)	0.00001
Policy evaluation method	TD(0) Operator REQ Operator

Table 5: Hyperparameters for CRRfSE, CRRfSE, REQfD and REQfSE

Hyperparameters	RLfD	RLfSE
λ_ψ	0.25	0.75
$\lambda_{intertwine}$	0.0	0.5

Table 6: Hyperparameters for Intertwine Exploration