Iterative Amortized Policy Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Policy networks are a central feature of deep reinforcement learning (RL) algorithms for continuous control, enabling the estimation and sampling of high-value actions. From the variational inference perspective on RL, policy networks, when used with entropy or KL regularization, are a form of *amortized optimization*, optimizing network parameters rather than the policy distributions directly. However, *direct* amortized mappings can yield suboptimal policy estimates and restricted distributions, limiting performance and exploration. Given this perspective, we consider the more flexible class of *iterative* amortized optimizers. We demonstrate that the resulting technique, iterative amortized policy optimization, yields performance improvements over direct amortization on benchmark continuous control tasks.

1 Introduction

Reinforcement learning (RL) algorithms involve policy evaluation and policy optimization [61]. Given a policy, one can estimate the value for each state or state-action pair following that policy, and given a value estimate, one can improve the policy to maximize the value. This latter procedure, policy optimization, can be challenging in continuous control due to instability and poor asymptotic performance. In deep RL, where policies over continuous actions are often parameterized by deep networks, such issues are typically tackled using regularization from previous policies [56, 57] or by maximizing policy entropy [48, 20]. These techniques can be interpreted as variational inference [43], using optimization to infer a policy that yields high expected return while satisfying prior policy constraints. This smooths the optimization landscape, improving stability and performance [3].

However, one subtlety arises: when used with entropy or KL regularization, policy networks perform *amortized* optimization [23]. That is, rather than optimizing the action distribution, e.g., mean and variance, many deep RL algorithms, such as soft actor-critic (SAC) [28, 29], instead optimize a network to output these parameters, *learning* to optimize the policy. Typically, this is implemented as a direct mapping from states to action distribution parameters. While such *direct* amortization schemes have improved the efficiency of variational inference as "encoder" networks [37, 53, 47], they also suffer from several drawbacks: ***1)*** they tend to provide suboptimal estimates [17, 36, 46], yielding a so-called "amortization gap" in performance [17], ***2)*** they are restricted to a single estimate [24], thereby limiting exploration, and ***3)*** they cannot generalize to new objectives, unlike, e.g., gradient-based [32] or gradient-free optimizers [55].

Inspired by techniques and improvements from variational inference, we investigate *iterative* amortized policy optimization. Iterative amortization [46] uses gradients or errors to iteratively update the parameters of a distribution. Unlike direct amortization, which receives gradients only *after* outputting the distribution, iterative amortization uses these gradients *online*, thereby learning to iteratively optimize. In generative modeling settings, iterative amortization empirically outperforms direct amortization [46, 45] and can find multiple modes of the optimization landscape [24].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

The contributions of this paper are as follows:

- We propose iterative amortized policy optimization, exploiting a new, fruitful connection between amortized variational inference and policy optimization.
- Using the suite of MuJoCo environments [66, 11], we demonstrate performance improvements over direct amortized policies, as well as more complex flow-based policies.
- We demonstrate novel benefits of this amortization technique: improved accuracy, providing multiple policy estimates, and generalizing to new objectives.

2 Background

2.1 Preliminaries

We consider Markov decision processes (MDPs), where $\mathbf{s}_t \in \mathcal{S}$ and $\mathbf{a}_t \in \mathcal{A}$ are the state and action at time t, resulting in reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$. Environment state transitions are given by $\mathbf{s}_{t+1} \sim p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, and the agent is defined by a parametric distribution, $p_\theta(\mathbf{a}_t|\mathbf{s}_t)$, with parameters θ.[1] The discounted sum of rewards is denoted as $\mathcal{R}(\tau) = \sum_t \gamma^t r_t$, where $\gamma \in (0, 1]$ is the discount factor, and $\tau = (\mathbf{s}_1, \mathbf{a}_1, \dots)$ is a trajectory. The distribution over trajectories is:

$$p(\tau) = \rho(\mathbf{s}_1) \prod_{t=1}^{T} p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) p_\theta(\mathbf{a}_t|\mathbf{s}_t), \tag{1}$$

where the initial state is drawn from the distribution $\rho(\mathbf{s}_1)$. The standard RL objective consists of maximizing the expected discounted return, $\mathbb{E}_{p(\tau)}\left[\mathcal{R}(\tau)\right]$. For convenience of presentation, we use the undiscounted setting ($\gamma = 1$), though the formulation can be applied with any valid γ.

2.2 KL-Regularized Reinforcement Learning

Various works have formulated RL, planning, and control problems in terms of probabilistic inference [18, 8, 67, 65, 10, 43]. These approaches consider the agent-environment interaction as a graphical model, then convert reward maximization into maximum marginal likelihood estimation, learning and inferring a policy that results in maximal reward. This conversion is accomplished by introducing one or more binary observed variables [16], denoted as $\mathcal{O}$ for "optimality" [43], with

$$p(\mathcal{O} = 1|\tau) \propto \exp\big(\mathcal{R}(\tau)/\alpha\big),$$

where α is a temperature hyper-parameter. We would like to infer latent variables, τ, and learn parameters, θ, that yield the maximum log-likelihood of optimality, i.e., $\log p(\mathcal{O} = 1)$. Evaluating this likelihood requires marginalizing the joint distribution, $p(\mathcal{O} = 1) = \int p(\tau, \mathcal{O} = 1) d\tau$. This involves averaging over all trajectories, which is intractable in high-dimensional spaces. Instead, we can use variational inference to lower bound this objective, introducing a structured approximate posterior distribution:

$$\pi(\tau|\mathcal{O}) = \prod_{t=1}^{T} p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \pi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}). \tag{2}$$

This provides the following lower bound on the objective:

$$\begin{aligned}
\log p(\mathcal{O} = 1) &= \log \int p(\mathcal{O} = 1|\tau) p(\tau) d\tau && (3)\\
&\geq \int \pi(\tau|\mathcal{O}) \left[\log \frac{p(\mathcal{O} = 1|\tau) p(\tau)}{\pi(\tau|\mathcal{O})} \right] d\tau && (4)\\
&= \mathbb{E}_\pi[\mathcal{R}(\tau)/\alpha] - D_{\text{KL}}(\pi(\tau|\mathcal{O}) \| p(\tau)). && (5)
\end{aligned}$$

Equivalently, we can multiply by α, defining the variational RL objective as:

$$\mathcal{J}(\pi, \theta) \equiv \mathbb{E}_\pi[\mathcal{R}(\tau)] - \alpha D_{\text{KL}}(\pi(\tau|\mathcal{O}) \| p(\tau)). \tag{6}$$

[1] In this paper, we consider the entropy-regularized case, where $p_\theta(\mathbf{a}_t|\mathbf{s}_t) = \mathcal{U}(-1, 1)$, i.e., uniform. However, we present the derivation for the KL-regularized case for full generality.

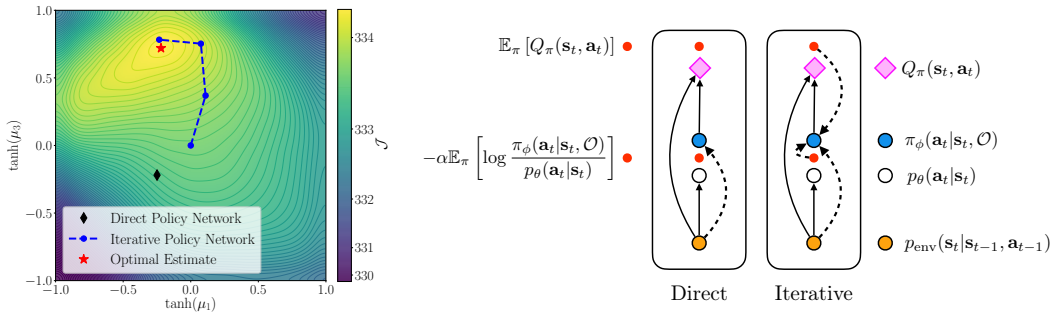


Figure 1: **Amortization**. **Left**: Optimization over two dimension of the policy mean, μ_1 and μ_3, for a particular state. A direct amortized policy network outputs a suboptimal estimate, yielding an *amortization gap* in performance. An iterative amortized policy network finds an improved estimate. **Right**: Diagrams of direct and iterative amortization. Larger circles denote distributions, and smaller red circles denote terms in the objective, $\mathcal{J}$ (Eq. 8). Dashed arrows denote amortization. Iterative amortization uses gradient feedback during optimization, while direct amortization does not.

This objective consists of the expected return (i.e., the standard RL objective) and a KL divergence between $\pi(\tau|\mathcal{O})$ and $p(\tau)$. In terms of states and actions, this objective is:

$$\mathcal{J}(\pi, \theta) = \mathbb{E}_{\substack{\mathbf{s}_t, r_t \sim p_{\text{env}} \\ \mathbf{a}_t \sim \pi}} \left[\sum_{t=1}^{T} r_t - \alpha \log \frac{\pi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O})}{p_\theta(\mathbf{a}_t|\mathbf{s}_t)} \right]. \tag{7}$$

At a given timestep, t, one can optimize this objective by estimating the future terms in the sum using a "soft" action-value (Q_π) network [27] or model [51]. For instance, sampling $\mathbf{s}_t \sim p_{\text{env}}$, slightly abusing notation, we can write the objective at time t as:

$$\mathcal{J}(\pi, \theta) = \mathbb{E}_\pi \left[Q_\pi(\mathbf{s}_t, \mathbf{a}_t) \right] - \alpha D_{\text{KL}}(\pi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}) || p_\theta(\mathbf{a}_t|\mathbf{s}_t)). \tag{8}$$

Policy optimization in the KL-regularized setting corresponds to maximizing $\mathcal{J}$ w.r.t. π. We often consider parametric policies, in which π is defined by distribution parameters, $\boldsymbol{\lambda}$, e.g., Gaussian mean, $\boldsymbol{\mu}$, and variance, $\boldsymbol{\sigma}^2$. In this case, policy optimization corresponds to maximizing:

$$\boldsymbol{\lambda} \leftarrow \arg\max_{\boldsymbol{\lambda}} \mathcal{J}(\pi, \theta). \tag{9}$$

Optionally, we can then also learn the policy prior parameters, θ [1].

2.3 Entropy & KL Regularized Policy Networks Perform Direct Amortization

Policy-based approaches to RL typically do not directly optimize the action distribution parameters, e.g., through gradient-based optimization. Instead, the action distribution parameters are output by a function approximator (deep network), f_ϕ, which is trained using deterministic [58, 44] or stochastic gradients [70, 31]. When combined with entropy or KL regularization, this policy network is a form of *amortized* optimization [23], learning to estimate policies. Again, denoting the action distribution parameters, e.g., mean and variance, as $\boldsymbol{\lambda}$, for a given state, $\mathbf{s}$, we can express this direct mapping as

$$\boldsymbol{\lambda} \leftarrow f_\phi(\mathbf{s}), \qquad \text{(direct amortization)} \tag{10}$$

denoting the corresponding policy as $\pi_\phi(\mathbf{a}|\mathbf{s}, \mathcal{O}; \boldsymbol{\lambda})$. Thus, f_ϕ attempts to *learn* to optimize Eq. 9. This setup is shown in Figure 1 (Right). Without entropy or KL regularization, i.e. $\pi_\phi(\mathbf{a}|\mathbf{s}) = p_\theta(\mathbf{a}|\mathbf{s})$, we can instead interpret the network as directly integrating the LHS of Eq. 4, which is less efficient and more challenging. Regularization smooths the optimization landscape, yielding more stable improvement and higher asymptotic performance [3].

Viewing policy networks as a form of direct amortized variational optimizer (Eq. 10) allows us to see that they are similar to "encoder" networks in variational autoencoders (VAEs) [37, 53]. However, there are several drawbacks to direct amortization.

Algorithm 1 Direct Amortization

Initialize ϕ
for each environment step **do**
  $\boldsymbol{\lambda} \leftarrow f_\phi(\mathbf{s}_t)$
  $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}; \boldsymbol{\lambda})$
  $\mathbf{s}_{t+1} \sim p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
end for
for each training step **do**
  $\phi \leftarrow \phi + \eta \nabla_\phi \mathcal{J}$
end for

Algorithm 2 Iterative Amortization

Initialize ϕ
for each environment step **do**
  Initialize $\boldsymbol{\lambda}$
  for each policy optimization iteration **do**
    $\boldsymbol{\lambda} \leftarrow f_\phi(\mathbf{s}_t, \boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}} \mathcal{J})$
  end for
  $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}; \boldsymbol{\lambda})$
  $\mathbf{s}_{t+1} \sim p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
end for
for each training step **do**
  $\phi \leftarrow \phi + \eta \nabla_\phi \mathcal{J}$
end for

Amortization Gap. Direct amortization results in suboptimal approximate posterior estimates, with the resulting gap in the variational bound referred to as the *amortization gap* [17]. Thus, in the RL setting, an amortized policy, π_ϕ, results in worse performance than the optimal policy within the parametric policy class, denoted as $\widehat{\pi}$. The amortization gap is the gap in following inequality:

$$\mathcal{J}(\pi_\phi, \theta) \leq \mathcal{J}(\widehat{\pi}, \theta).$$

Because $\mathcal{J}$ is a variational bound on the RL objective, i.e., expected return, a looser bound, due to amortization, prevents one from more completely optimizing this objective.

This is shown in Figure 1 (Left),[2] where $\mathcal{J}$ is plotted over two dimensions of the policy mean at a particular state in the MuJoCo environment `Hopper-v2`. The estimate of a direct amortized policy (⧫) is suboptimal, far from the optimal estimate (★). While the relative difference in the objective is relatively small, suboptimal estimates prevent sampling and exploring high-value regions of the action-space. That is, suboptimal estimates have only a *minor* impact on evaluation performance (see Appendix B.4) but hinder effective data collection.

Single Estimate. Direct amortization is limited to a single, static estimate. In other words, if there are multiple high-value regions of the action-space, a uni-modal (e.g., Gaussian) direct amortized policy is restricted to only one region, thereby limiting exploration. Note that this is an additional restriction beyond simply considering uni-modal distributions, as a generic optimization procedure may arrive at multiple uni-modal estimates depending on initialization and stochastic sampling (see Section 3.2). While multi-modal distributions reduce the severity of this restriction [62, 26], the other limitations of direct amortization still persist.

Inability to Generalize Across Objectives. Direct amortization is a feedforward procedure, receiving gradients from the objective only *after* estimation. This is contrast to other forms of optimization, which receive gradients (feedback) *during* estimation. Thus, unlike other optimizers, direct amortization is incapable of generalizing to new objectives, e.g., if $Q_\pi(\mathbf{s}, \mathbf{a})$ or $p_\theta(\mathbf{a}|\mathbf{s})$ change, which is a desirable capability for adapting to new tasks or environments.

To improve upon this scheme and overcome these drawbacks, in Section 3, we turn to a technique developed in generative modeling, *iterative amortization* [46], retaining the efficiency of amortization while employing a more flexible iterative estimation procedure.

2.4 Related Work

Previous works have investigated methods for improving policy optimization. QT-Opt [34] uses the cross-entropy method (CEM) [55], an iterative derivative-free optimizer, to optimize a Q-value estimator for robotic grasping. CEM and related methods are also used in model-based RL for performing model-predictive control [50, 13, 51, 30]. Gradient-based policy optimization [32, 59, 9], in contrast, is less common, however, gradient-based optimization can also be combined with CEM

[2] Additional 2D plots are shown in Appendix Figure B.3.

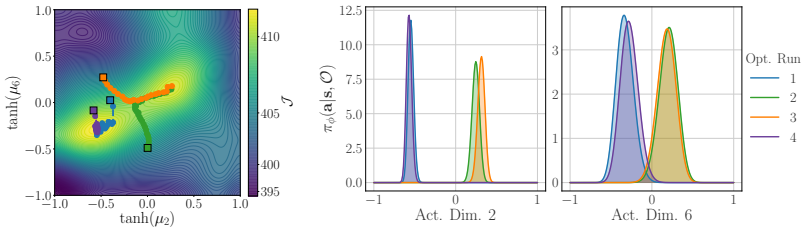


Figure 2: **Estimating Multiple Policy Modes**. Unlike direct amortization, which is restricted to a single estimate, iterative amortization can effectively sample from multiple high-value action modes. This is shown for a particular state in `Ant-v2`, showing multiple optimization runs across two action dimensions (**Left**). Each square denotes an initialization. The optimizer finds both modes, with the densities plotted on the **Right**. This capability provides increased flexibility in action exploration.

[5]. Most policy-based methods use direct amortization, either using a feedforward [28] or recurrent [25] network. Similar approaches have also been applied to model-based value estimates [12, 15, 4], as well as combining direct amortization with model predictive control [42] and planning [54]. A separate line of work has explored improving the policy distribution, using normalizing flows [26, 62] and latent variables [64]. In principle, iterative amortization can perform policy optimization in each of these settings.

Iterative amortized policy optimization is conceptually similar to negative feedback control [7], using errors to update policy estimates. However, while conventional feedback control methods are often restricted in their applicability, e.g., linear systems and quadratic cost, iterative amortization is generally applicable to any differentiable control objective. This is analogous to the generalization of Kalman filtering [35] to amortized filtering [45] for state estimation.

3 Iterative Amortized Policy Optimization

3.1 Formulation

Iterative amortization [46] utilizes errors or gradients to update the approximate posterior distribution parameters. While various forms exist, we consider gradient-encoding models [6] due to their generality. Compared with direct amortization (Eq. 10), we use iterative amortized optimizers of the general form

$$\boldsymbol{\lambda} \leftarrow f_\phi(\mathbf{s}, \boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}} \mathcal{J}), \qquad \text{(iterative amortization)} \tag{11}$$

also shown in Figure 1 (Right), where f_ϕ is a deep network and $\boldsymbol{\lambda}$ are the action distribution parameters. For example, if $\pi = \mathcal{N}(\mathbf{a}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$, then $\boldsymbol{\lambda} \equiv [\boldsymbol{\mu}, \boldsymbol{\sigma}]$. Technically, $\mathbf{s}$ is redundant, as the state dependence is already captured in $\mathcal{J}$, but this can empirically improve performance [46]. In practice, the update is carried out using a "highway" gating operation [33, 60]. Denoting $\boldsymbol{\omega}_\phi \in [0, 1]$ as the gate and $\boldsymbol{\delta}_\phi$ as the update, both of which are output by f_ϕ, the gating operation is expressed as

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\omega}_\phi \odot \boldsymbol{\lambda} + (\mathbf{1} - \boldsymbol{\omega}_\phi) \odot \boldsymbol{\delta}_\phi, \tag{12}$$

where $\odot$ denotes element-wise multiplication. This update is typically run for a fixed number of steps, and, as with a direct policy, the iterative optimizer is trained using stochastic gradient estimates of $\nabla_\phi \mathcal{J}$, obtained through the path-wise derivative estimator [37, 53, 31]. Because the gradients $\nabla_{\boldsymbol{\lambda}} \mathcal{J}$ must be estimated online, i.e., during policy optimization, this scheme requires some way of estimating $\mathcal{J}$ online through a parameterized Q-value network [49] or a differentiable model [31].

3.2 Benefits of Iterative Amortization

Reduced Amortization Gap. Iterative amortized optimizers are more flexible than their direct counterparts, incorporating feedback from the objective *during* policy optimization (Algorithm 2), rather than only *after* optimization (Algorithm 1). Increased flexibility improves the accuracy of optimization, thereby tightening the variational bound [46, 45]. We see this flexibility in Figure 1 (Left), where an iterative amortized policy network iteratively refines the policy estimate (●), quickly arriving near the optimal estimate.

Multiple Estimates. Iterative amortization, by using stochastic gradients and random initialization, can traverse the optimization landscape. As with any iterative optimization scheme, this allows iterative amortization to obtain multiple valid estimates, referred to as "multi-stability" in the generative modeling literature [24]. We illustrate this capability across two action dimensions in Figure 2 for a state in the `Ant-v2` MuJoCo environment. Over multiple policy optimization runs, iterative amortization finds multiple modes, sampling from two high-value regions of the action space. This provides increased flexibility in action exploration, despite only using a uni-modal policy distribution.

Generalization Across Objectives. Iterative amortization uses the gradients of the objective *during* optimization, i.e., feedback, allowing it to potentially generalize to new or updated objectives. We see this in Figure 1 (Left), where iterative amortization, despite being trained with a *different* value estimator, is capable of generalizing to this new objective. We demonstrate this capability further in Section 4. This opens the possibility of accurately and efficiently performing policy optimization in new settings, e.g., a rapidly changing model or new tasks.

3.3 Consideration: Mitigating Value Overestimation

Why are more powerful policy optimizers typically not used in practice? Part of the issue stems from value overestimation. Model-free approaches generally estimate Q_π using function approximation and temporal difference learning. However, this has the pitfall of value overestimation, i.e., positive bias in the estimate, $\widehat{Q}_\pi$ [63]. This issue is tied to uncertainty in the value estimate, though it is distinct from optimism under uncertainty. If the policy can exploit regions of high uncertainty, the resulting target values will introduce positive bias into the estimate. More flexible policy optimizers exacerbate the problem, exploiting this uncertainty to a greater degree. Further, a rapidly changing policy increases the difficulty of value estimation [52].

Various techniques have been proposed for mitigating value overestimation in deep RL. The most prominent technique, double deep Q-network [69] maintains two Q-value estimates [68], attempting to decouple policy optimization from value estimation. Fujimoto et al. [22] apply and improve upon this technique for actor-critic settings, estimating the target Q-value as the minimum of two Q-networks, Q_{ψ_1} and Q_{ψ_2}:

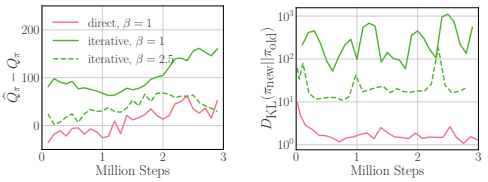
$$\widehat{Q}_\pi(\mathbf{s}, \mathbf{a}) = \min_{i=1,2} Q_{\psi_i'}(\mathbf{s}, \mathbf{a}),$$

where ψ_i' denotes the target network parameters. As noted by Fujimoto et al. [22], this not only counteracts value overestimation, but also penalizes high-variance value estimates, because the minimum decreases with the variance of the estimate. Ciosek et al. [14] noted that, for a bootstrapped ensemble of two Q-networks, the minimum operation can be interpreted as estimating

$$\widehat{Q}_\pi(\mathbf{s}, \mathbf{a}) = \mu_Q(\mathbf{s}, \mathbf{a}) - \beta\sigma_Q(\mathbf{s}, \mathbf{a}), \tag{13}$$

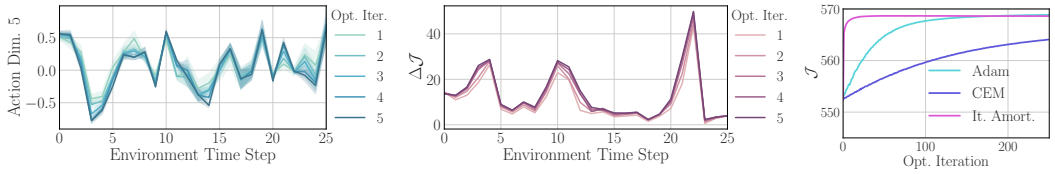
with mean $\mu_Q(\mathbf{s}, \mathbf{a}) \equiv \frac{1}{2}\sum_{i=1,2} Q_{\psi_i'}(\mathbf{s}, \mathbf{a})$, standard deviation $\sigma_Q(\mathbf{s}, \mathbf{a}) \equiv (\frac{1}{2}\sum_{i=1,2}(Q_{\psi_i'}(\mathbf{s}, \mathbf{a}) - \mu_Q(\mathbf{s}, \mathbf{a}))^2)^{1/2}$, and $\beta = 1$. Thus, to further penalize high-variance value estimates, preventing value overestimation, we can increase β. For large β, however, value estimates become overly pessimistic, negatively impacting training. Thus, β reduces target value variance at the cost of increased bias.

Due to the flexibility of iterative amortization, the default $\beta = 1$ results in increased value bias and a more rapidly changing policy as compared with direct amortization (Figure 3). Further penalizing high-variance target values ($\beta = 2.5$) reduces value overestimation and improves stability. For details, see Appendix A.2. Recent techniques for mitigating overestimation have been proposed, such as adjusting α [19]. In offline RL, this issue has been tackled through the action prior [21, 40, 71] or by altering Q-network training [2, 41]. While such techniques could be used here, increasing β provides a simple solution with no additional computational overhead. This is a meaningful insight toward applying more powerful policy optimizers.



(a) (b)

Figure 3: **Mitigating Value Overestimation**. With $\beta = 1$, iterative amortization results in (**a**) higher value overestimation and (**b**) a more rapidly changing policy as compared with direct amortization. Increasing β helps to mitigate these issues.



(a) Policy (b) Improvement (c) Comparison

Figure 4: **Policy Optimization**. Visualization over time steps of **(a)** one dimension of the policy distribution and **(b)** the improvement in the objective, $\Delta\mathcal{J}$, across policy optimization iterations. **(c)** Comparison of iterative amortization with Adam [38] (gradient-based) and CEM [55] (gradient-free). Iterative amortization is an order of magnitude more efficient.

4 Experiments

4.1 Setup

To focus on policy optimization, we implement iterative amortized policy optimization using the soft actor-critic (SAC) setup described by Haarnoja et al. [29]. This uses two Q-networks, uniform action prior, $p_\theta(\mathbf{a}|\mathbf{s}) = \mathcal{U}(-1, 1)$, and a tuning scheme for the temperature, α. In our experiments, "direct" refers to direct amortization employed in SAC, i.e., a direct policy network, and "iterative" refers to iterative amortization. Both approaches use the *same* network architecture, adjusting only the number of inputs and outputs to accommodate gradients, current policy estimates, and gated updates (Sec. 3.1). Unless otherwise stated, we use 5 iterations per time step for iterative amortization, following [46]. For details, refer to Appendix A and Haarnoja et al. [28, 29].

4.2 Analysis

4.2.1 Visualizing Policy Optimization

We provide 2D visualizations of iterative amortized policy optimization in Figures 1 & 2, with additional 2D plots in the appendix. In Figure 4, we visualize iterative refinement using a single action dimension from `Ant-v2` across time steps. The refinements in Figure 4a give rise to the objective improvements in Figure 4b. We compare with Adam [38] (gradient-based) and CEM [55] (gradient-free) in Figure 4c, where iterative amortization is *an order of magnitude* more efficient. This trend is consistent across environments (see Appendix B).

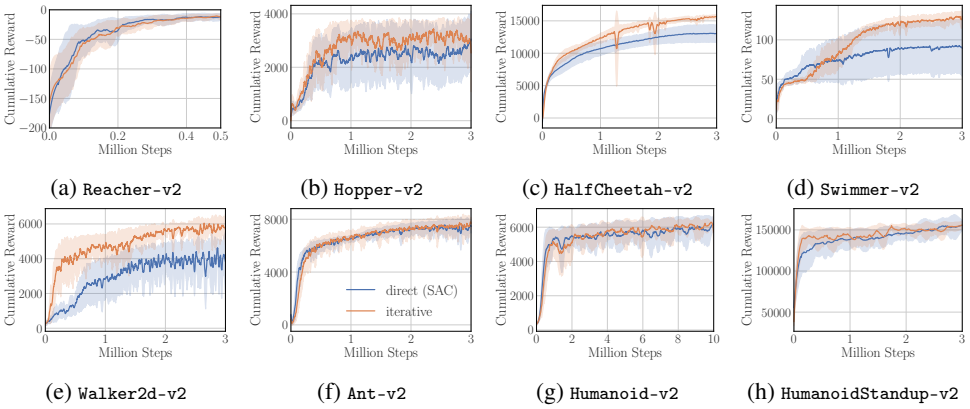
4.2.2 Performance Comparison

We evaluate iterative amortized policy optimization on the suite of MuJoCo [66] continuous control tasks from OpenAI gym [11]. In Figure 5, we compare the cumulative reward of direct and iterative amortized policy optimization across environments. Each curve shows the mean and $\pm$ standard deviation of 5 random seeds. In all cases, iterative amortized policy optimization matches or outperforms the baseline direct amortized method, both in sample efficiency and final performance. Iterative amortization also yields more consistent, lower variance performance.

4.2.3 Improved Exploration: Multiple Policy Modes

As described in Section 3.2, iterative amortization is capable of obtaining multiple estimates, i.e., multiple modes of the optimization objective. To confirm that iterative amortization has captured multiple modes, at the end of training, we take an iterative agent trained on `Walker2d-v2` and histogram the distances between policy means across separate runs of policy optimization per state (Fig. 7a). For the state with the largest distance, we plot 2D projections of the optimization objective, $\mathcal{J}$, across action dimensions in Figure 7b, as well as the policy density across 10 optimization runs (Fig. 7c). A subset of states indeed still retain multiple policy modes. Additional results on other environments are presented in the appendix.

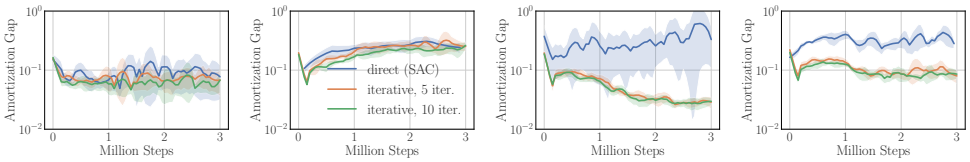
To better understand whether the performance benefits of iterative amortization are coming purely from improved exploration via multiple modes, we also compare with direct amortization with a



(a) `Reacher-v2` (b) `Hopper-v2` (c) `HalfCheetah-v2` (d) `Swimmer-v2`

(e) `Walker2d-v2` (f) `Ant-v2` (g) `Humanoid-v2` (h) `HumanoidStandup-v2`

Figure 5: **Performance Comparison**. Iterative amortized policy optimization performs comparably with or better than direct amortization across MuJoCo environments. Curves show the mean $\pm$ std. dev. of performance over 5 random seeds.



(a) `Hopper-v2` (b) `HalfCheetah-v2` (c) `Walker2d-v2` (d) `Ant-v2`

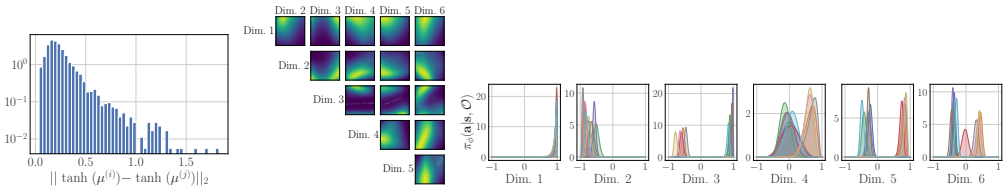
Figure 6: **Amortization Gap**. Iterative amortization achieves similar or lower amortization gaps than direct amortization. Gaps are estimated using stochastic gradient-based optimization over 100 random states. Curves show the mean and $\pm$ std. dev. over 5 random seeds.

multi-modal policy distribution. This is formed using inverse autoregressive flows [39], a type of normalizing flow (NF). Results are presented in the appendix. Using a multi-modal policy reduces the performance deficiencies on `Hopper-v2` and `Walker2d-v2`, indicating that much of the benefit of iterative amortization is due to lifting direct amortization's restriction to a single, uni-modal policy estimate. Yet, direct + NF still struggles on `HalfCheetah-v2` compared with iterative amortization, suggesting that more complex, multi-modal distributions are not the *only* consideration.

4.2.4 Improved Optimization: Amortization Gap

To evaluate policy optimization accuracy, we estimate per-step amortization gaps, performing additional iterations of gradient ascent on $\mathcal{J}$ w.r.t. the policy parameters, $\boldsymbol{\lambda} \equiv [\boldsymbol{\mu}, \boldsymbol{\sigma}]$ (see Appendix A.3). To analyze generalization, we also evaluate the iterative agents trained with 5 iterations for an additional 5 amortized iterations. Results are shown in Figure 6. We emphasize that it is challenging to *directly* compare amortization gaps across optimization schemes, as these involve different value functions, and therefore different objectives. Likewise, we estimate the amortization gap using the learned Q-networks, which may be biased (Figure 3). Nevertheless, we find that iterative amortized policy optimization achieves, on average, lower amortization gaps than direct amortization across all environments. Additional amortized iterations at evaluation yield further estimated improvement, demonstrating generalization beyond the optimization horizon used during training.

The amortization gaps are small relative to the objective, playing a negligible role in *evaluation* performance (see Appendix B). Rather, improved policy optimization is helpful for *training*, allowing the agent to explore states where value estimates are highest. To probe this further, we train iterative amortized policy optimization while varying the number of iterations per step in $\{1, 2, 5\}$, yielding optimizers with varying degrees of accuracy. Note that each optimizer is, in theory, capable of finding multiple modes. In Figure 8, we see that training with additional iterations improves performance and optimization accuracy. We stress that the exact form of this relationship depends on the Q-value estimator and other factors. We present additional results in the appendix.



(a) (b) (c)

Figure 7: **Multiple Policy Modes**. **(a)** Histogram of distances between policy means ($\boldsymbol{\mu}$) across optimization runs (i and j) over seeds and states on `Walker2d-v2` at 3 million environment steps. For the state with the largest distance, **(b)** shows the projected optimization surface on each pair of action dimensions, and **(c)** shows the policy density for 10 optimization runs.

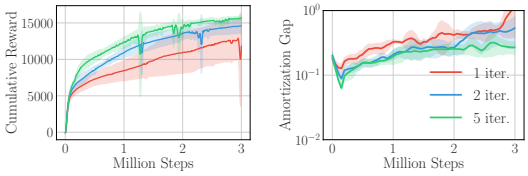


Figure 8: **Varying Iterations During Training**. Performance (**Left**) and estimated amortization gap (**Right**) for varying numbers of policy optimization iterations per step during training on `HalfCheetah-v2`. Increasing the iterations generally improves performance and decreases the estimated amortization gap.

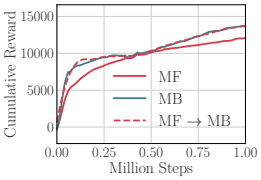


Figure 9: **Zero-shot generalization** of iterative amortization from model-free (MF) to model-based (MB) value estimates.

4.2.5 Generalizing to Model-Based Value Estimates

Direct amortization is a purely feedforward process and is therefore incapable of generalizing to new objectives. In contrast, because iterative amortization is formulated through gradient-based feedback, such optimizers may be capable of generalizing to new objective estimators, as shown in Figure 1. To demonstrate this capability further, we apply iterative amortization with model-based value estimators, using a learned deterministic model on `HalfCheetah-v2` (see Appendix A.5). We evaluate the generalizing capabilities in Figure 9 by transferring the policy optimizer from a model-free agent to a model-based agent. Iterative amortization generalizes to these new value estimates, *instantly* recovering the performance of the model-based agent. This highlights the opportunity for instantly incorporating new tasks, goals, or model estimates into policy optimization.

5 Discussion

We have introduced iterative amortized policy optimization, a flexible and powerful policy optimization technique. In so doing, we have highlighted several limitations of direct amortization: 1) limited accuracy, as quantified by the amortization gap, 2) restriction to a single estimate, limiting exploration, and 3) inability to generalize to new objectives, limiting the transfer of these policy optimizers. As confirmed through our empirical analysis on benchmark continuous control environments, iterative amortization provides a step toward improving each of these restrictions, with accompanying improvements in performance over current direct amortization methods. Thus, iterative amortization provides a drop-in replacement and improvement over direct policy networks in deep RL. While this requires an iterative optimization scheme, it is still vastly more efficient than standard optimizers.

Although we have discussed three separate limitations of direct amortization, these factors are highly interconnected. By broadening policy optimization to an iterative procedure, we automatically obtain a potentially more accurate and general policy optimizer, with the capability of obtaining multiple modes. While our analysis suggests that the improved exploration resulting from multiple modes is the primary factor affecting performance, future work could tease out these effects further and assess the relative contributions of these improvements in additional environments. We are hopeful that iterative amortized policy optimization, by providing a more powerful, exploratory, and general optimizer, will enable a range of improved RL algorithms.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. *arXiv preprint arXiv:1907.04543*, 2019.

[3] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pages 151–160, 2019.

[4] Brandon Amos, Samuel Stanton, Denis Yarats, and Andrew Gordon Wilson. On the model-based stochastic value gradient for continuous reinforcement learning. *arXiv preprint arXiv:2008.12775*, 2020.

[5] Brandon Amos and Denis Yarats. The differentiable cross-entropy method. In *International Conference on Machine Learning*, 2020.

[6] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3981–3989, 2016.

[7] Karl Johan Astrom and Richard M Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[8] Hagai Attias. Planning by probabilistic inference. In *AISTATS*. Citeseer, 2003.

[9] Homanga Bharadhwaj, Kevin Xie, and Florian Shkurti. Model-predictive planning via cross-entropy and gradient-based optimization. In *Learning for Dynamics and Control*, pages 277–286, 2020.

[10] Matthew Botvinick and Marc Toussaint. Planning as inference. *Trends in cognitive sciences*, 16(10):485–488, 2012.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[12] Arunkumar Byravan, Jost Tobias Springenberg, Abbas Abdolmaleki, Roland Hafner, Michael Neunert, Thomas Lampe, Noah Siegel, Nicolas Heess, and Martin Riedmiller. Imagined value gradients: Model-based policy optimization with tranferable latent dynamics models. In *Conference on Robot Learning*, pages 566–589, 2020.

[13] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

[14] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, pages 1787–1798, 2019.

[15] Ignasi Clavera, Yao Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. In *International Conference on Learning Representations*, 2020.

[16] Gregory F Cooper. A method for using belief networks as influence diagrams. In *Fourth Workshop on Uncertainty in Artificial Intelligence.*, 1988.

[17] Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086, 2018.

[18] Peter Dayan and Geoffrey E Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

[19] Roy Fox. Toward provably unbiased temporal-difference value estimation. *Optimization Foundations for Reinforcement Learning Workshop at NeurIPS*, 2019.

[20] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 202–211. AUAI Press, 2016.

[21] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

[22] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596, 2018.

[23] Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the Cognitive Science Society*, volume 36, 2014.

[24] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433, 2019.

[25] Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sebastien Racaniere, Theophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, et al. An investigation of model-free planning. In *International Conference on Machine Learning*, pages 2464–2473, 2019.

[26] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1851–1860, 2018.

[27] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1352–1361. JMLR. org, 2017.

[28] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1856–1865, 2018.

[29] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[30] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[31] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pages 2944–2952, 2015.

[32] Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[34] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[35] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[36] Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. Semi-amortized variational autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[37] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Proceedings of the International Conference on Learning Representations*, 2014.

[38] Durk P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[40] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11784–11794, 2019.

[41] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[42] Keuntaek Lee, Kamil Saigol, and Evangelos A Theodorou. Safe end-to-end imitation learning for model predictive control. In *International Conference on Robotics and Automation*, 2019.

[43] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[44] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[45] Joseph Marino, Milan Cvitkovic, and Yisong Yue. A general method for amortizing variational filtering. In *Advances in Neural Information Processing Systems*, 2018.

[46] Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pages 3403–3412, 2018.

[47] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799, 2014.

[48] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[49] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*, 2013.

[50] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[51] Alexandre Piché, Valentin Thomas, Cyril Ibrahim, Yoshua Bengio, and Chris Pal. Probabilistic planning with sequential monte carlo methods. In *International Conference on Learning Representations*, 2019.

[52] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. *arXiv preprint arXiv:2004.07804*, 2020.

[53] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1278–1286, 2014.

[54] Benjamin Rivière, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning. *IEEE Robotics and Automation Letters*, 5(3):4249–4256, 2020.

[55] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[56] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[57] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[58] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pages 387–395, 2014.

[59] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning*, pages 4732–4741, 2018.

[60] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems (NIPS)*, pages 2377–2385, 2015.

[61] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[62] Yunhao Tang and Shipra Agrawal. Boosting trust region policy optimization by normalizing flows policy. *arXiv preprint arXiv:1809.10326*, 2018.

[63] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[64] Dhruva Tirumala, Hyeonwoo Noh, Alexandre Galashov, Leonard Hasenclever, Arun Ahuja, Greg Wayne, Razvan Pascanu, Yee Whye Teh, and Nicolas Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019.

[65] Emanuel Todorov. General duality between optimal control and estimation. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4286–4292. IEEE, 2008.

[66] Emanuel Todorov, Tom Erez, and Yuval 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.

[67] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd international conference on Machine learning*, pages 945–952. ACM, 2006.

[68] Hado Van Hasselt. Double q-learning. In *Advances in neural information processing systems*, pages 2613–2621, 2010.

[69] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[70] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

[71] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We demonstrate performance improvements and novel benefits of iterative amortization in Section 4. Each claim is supported by empirical evidence.
   (b) Did you describe the limitations of your work? [Yes] We discuss the additional computation requirements and the necessity of mitigating value overestimation.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not see any immediate societal impacts of this work beyond the general impacts of improvements in machine learning.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We have included code in the supplementary material with an accompanying README file.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All details and hyperparameters are provided in the Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars on each of our main quantitative results comparing performance.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We provide these details in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the creators of the environments and software packages used in this paper.
   (b) Did you mention the license of the assets? [Yes] We mention or cite the license for each asset.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]