Co-Adaptation of Algorithmic and Implementational Innovations in Inference-based Deep Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recently many algorithms were devised for reinforcement learning (RL) with function approximation. While they have clear algorithmic distinctions, they also have many implementation differences that are algorithm-independent and sometimes under-emphasized. Such mixing of algorithmic novelty and implementation craftsmanship makes rigorous analyses of the sources of performance improvements across algorithms difficult. In this work, we focus on a series of off-policy inference-based actor-critic algorithms – MPO, AWR, and SAC – to decouple their algorithmic innovations and implementation decisions. We present unified derivations through a single control-as-inference objective, where we can categorize each algorithm as based on either Expectation-Maximization (EM) or direct Kullback-Leibler (KL) divergence minimization and treat the rest of specifications as implementation details. We performed extensive ablation studies, and identified substantial performance drops whenever implementation details are *mismatched* for algorithmic choices. These results show which implementation details are co-adapted and co-evolved with algorithms, and which are transferable across algorithms: as examples, we identified that tanh Gaussian policy and network sizes are highly adapted to algorithmic types, while layer normalization and ELU are critical for MPO's performances but also transfer to noticeable gains in SAC. We hope our work can inspire future work to further demystify sources of performance improvements across multiple algorithms and allow researchers to build on one another's both algorithmic and implementational innovations.

1 Introduction

Deep reinforcement learning (RL) has achieved huge empirical successes in both continuous [35, 17] and discrete [38, 25] problem settings with on-policy [51, 53] and off-policy [12, 21, 22] algorithms. Especially in the continuous control domain, interpreting RL as probabilistic inference [59, 60] has yielded many kinds of algorithms with strong empirical performances [33, 49, 50, 11, 28, 20, 15, 23].

Recently, there have been a series of off-policy algorithms derived from this perspective for learning policies with function approximations [2, 21, 47]. Notably, Soft Actor Critic (SAC) [21, 22], based on maximum entropy objective and soft Q-function, significantly outperforms on-policy [51, 53] and off-policy [35, 17, 12] methods. Maximum a posteriori Policy Optimisation (MPO) [2], inspired by REPS [49], employs a pseudo-likelihood objective, and achieves high sample-efficiency and fast convergence compared to the variety of policy gradient methods [35, 53, 6]. Similar to MPO, Advantage Weighted Regression (AWR) [47], and its variant, Advantage Weighted Actor-Critic (AWAC), also employ the pseudo-likelihood objective weighted by the exponential of advantages, and reports more stable performance than baselines both in online and offline [34] settings.

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

While these inference-based algorithms have similar derivations to each other, their empirical performances have large gaps among them when evaluated on the standard continuous control benchmarks, such as OpenAI Gym [5] or DeepMind Control Suite [57]. Critically, as each algorithm has unique low-level implementation design decisions – such as value estimation techniques, distribution of the policy, and network architectures – aside from high-level algorithmic choices, it is difficult to exactly identify the causes of these performance gaps as algorithmic or implementational.

In this paper, we first derive MPO, AWR, and SAC from a single objective function, through either Expectation-Maximization (EM) or KL minimization, mathematically clarifying the algorithmic connections among the recent state-of-the-art off-policy actor-critic algorithms. This unified derivation allows us to precisely identify implementation techniques for each algorithm, which are residual design choices in each method that are generally transferable to other algorithms. To reveal the sources of the performance gaps, we experiment with carefully-selected ablations of these identified implementation techniques, such as tanh-squashed Gaussian policy, network architectures, and clipped double Q-learning [12]. Specifically, we keep the high-level algorithmic designs while normalizing the implementation designs, enabling proper algorithmic comparisons. Our empirical results successfully distinguish between highly co-adapted design choices and no-co-adapted ones. We identified that clipped double Q-learning and tanh-squashed policies, the sources of SoTA performance of SAC, are highly co-adapted to KL-minimization-based method, SAC, and difficult to transfer and benefit in EM-based methods, MPO or AWR. In contrast, we discover that ELU [7] and layer normalization [4], the sources of SoTA performance of MPO, are transferable choices from MPO that also significantly benefit SAC. We hope our work can inspire more future works to precisely decouple algorithmic innovations from implementation details, which allows exact sources of performance gains to be identified and algorithmic researches to better build on one another.

2 Related Work

Inference-based RL algorithms RL as probabilistic inference has been studied in several prior contexts [59, 60, 33, 10, 56, 44], but many recently-proposed algorithms [2, 21, 47] are derived separately and their exact relationships are difficult to get out directly, due to mixing of algorithmic and implementational details, inconsistent implementation choices, environment-specific tunings, and benchmark differences. Our work organizes them as a unified policy iteration method, to clarify their exact mathematical algorithmic connections and tease out subtle, but important, implementation differences. We center our analyses around MPO, AWR, and SAC, because they are representative algorithms that span both EM-based [48, 49, 42, 43, 1, 55, 41] and KL-control-based [58, 50, 11, 28, 20, 32, 31] RL and achieve some of the most competitive performances on popular benchmarks [5, 57]. REPS [49], an EM approach, inspired MPO, AWR, and our unified objective in Eq. 1, while Soft Q-learning [20], a practical extension of KL control to continuous action space through Liu and Wang [37], directly led to the development of SAC.

Meta analyses of RL algorithms While many papers propose novel algorithms, some recent works focused on meta analyses of some of the popular algorithms, which attracted significant attention due to these algorithms' high-variance evaluation performances, reproducibility difficulty [8, 27, 64], and frequent code-level optimizations [24, 62, 9, 3]. Henderson et al. [24] empirically showed how these RL algorithms have inconsistent results across different official implementations and high variances even across runs with the same hyper-parameters, and recommended a concrete action item for the community – use more random seeds. Tucker et al. [62] show that high performances of action-dependent baselines [18, 16, 36, 19, 66] were more directly due to different subtle implementation choices. Engstrom et al. [9] focus solely on PPO and TRPO, two on-policy algorithms, and discuss how code-level optimizations, instead of the claimed algorithmic differences, actually led more to PPO's superior performances. Andrychowicz et al. [3] describes low-level (e.g. hyper-parameter choice, and regularization) and high-level (e.g. policy loss) design choices in on-policy algorithms affects the performance of PPO by showing results of large-scale evaluations.

In contrast to those prior works that mainly focus on a single family of on-policy algorithms, PPO and TRPO, and evaluating their implementation details alone, our work focuses on two distinct families of off-policy algorithms, and more importantly, presents unifying mathematical connections among independently-proposed state-of-the-art algorithms. Our experiments prove how *some* implementation choices in Table 1 are co-evolved with algorithmic innovations and/or have non-trivial effects on the performance of off-policy inference-based methods.

Method	Algorithm		Implementation				
	π_q	π_p	π_q update	π_p update	$\mathcal{G}$	$\mathcal{G}$ estimate	π_θ
MPO	Non-parametric	π_θ	Analytic + TR	SG + TR	Q^{π_p}	Retrace(1)	$\mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$
AWR	Non-parametric	π_θ	Analytic	Mixture + SG	$A^{\boldsymbol{\pi_p}}$	TD(λ)	$\mathcal{N}(\mu_\theta(s), \Sigma)$
AWAC	Non-parametric	π_θ	Analytic	Mixture + SG	Q^{π_p}	TD(0)	$\mathcal{N}(\mu_\theta(s), \Sigma_\theta)$
SAC	π_θ	Fixed	SG	(Fixed to Unif.)	$Q^{\boldsymbol{\pi_q}}_{\mathbf{soft}}$	TD(0) + TD3	Tanh($\mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$)
PoWER	Non-parametric	π_θ	Analytic	Analytic	$\eta \log Q^{\pi_p}$	TD(1)	$\mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$
RWR	Non-parametric	π_θ	Analytic	SG	$\eta \log r$	–	$\mathcal{N}(\mu_\theta(s), \Sigma)$
REPS	Non-parametric	π_q	Analytic	π_q	A^{π_p}	TD(0)	Softmax
UREX	Non-parametric	π_θ	Analytic	SG	Q^{π_p}	TD(1)	Softmax
V-MPO	Non-parametric	π_θ	Analytic + TR	SG + TR	A^{π_p}	n-step TD	$\mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$
TRPO	π_θ	π_q	TR	π_q	A^{π_p}	TD(1)	$\mathcal{N}(\mu_\theta(s), \Sigma_\theta)$
PPO	π_θ	π_q	SG + TR	π_q	A^{π_p}	GAE	$\mathcal{N}(\mu_\theta(s), \Sigma_\theta)$
DDPG*	π_θ	Fixed	SG	(Fixed)	Q^{π_q}	TD(0)	$\mu_\theta(s)$
TD3*	π_θ	Fixed	SG	(Fixed)	Q^{π_q}	TD(0) + TD3	$\mu_\theta(s)$

Table 1: Taxonomy based on the components of inference-based off-policy algorithms: MPO [2], AWR [47], AWAC [41], SAC [21], and other algorithms (see Appendix B). We follow the notation of Sec. 3 & 4. We characterize them with the parameterization of the policy (π_q and π_p), how policies are updated (π_q and π_p; SG stands for stochastic-gradient-based, and TR stands for trust-region-based), the choice of $\mathcal{G}(\cdot)$, how $\mathcal{G}$ is estimated, and the distribution of the policy. While the advantage function just can be interpreted as Q-function with baseline subtraction, we explicitly write A^π when the state-value function is parameterized instead of Q-function. (*Note that DDPG and TD3 are not "inference-based", but can be classified as KL control variants.)

3 Preliminaries

We consider a Markov Decision Process (MDP) defined by state space $\mathcal{S}$, action space $\mathcal{A}$, state transition probability distribution $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$, initial state distribution $p_1 : \mathcal{S} \to [0, \infty)$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1)$. Let R_t denote a discounted return $\sum_{u=t}^{\infty} \gamma^{u-t} r(s_u, a_u)$. We assume the standard RL setting, where the agent chooses actions based on a parametric policy π_θ and seeks for its parameter that maximizes the expected return $\mathbb{E}_{\pi_\theta}[R_1]$. Value functions for a policy π are the expected return conditioned by a state-action pair or a state, that is, $Q^\pi(s_t, a_t) := \mathbb{E}_\pi[R_t|s_t, a_t]$, and $V^\pi(s_t) := \mathbb{E}_\pi[R_t|s_t]$. They are called the state-action-value function (Q-function), and state-value function, respectively. The advantage function is defined as $A^\pi(s, a) := Q^\pi(s, a) - V^\pi(s)$. The Q-function for a policy π is known to be a unique fixed point of the Bellman operator $\mathcal{T}^\pi$ defined by $\mathcal{T}^\pi Q(s, a) = r(s, a) + \int \pi(a'|s')p(s'|s, a)Q(s', a')\, ds'da'$ for any function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. We denote a trajectory or successive state-action sequence as $\tau := (s_1, a_1, s_2, a_2, \dots)$. We also define an unnormalized state distribution under the policy π by $d_\pi(s) = \sum_{t=1}^{\infty} \gamma^t p(s_t = s|\pi)$.

Inference-based Methods For simplicity, we consider a finite-horizon MDP with a time horizon T for the time being. As a result, a trajectory τ becomes a finite length: $\tau := (s_1, a_1, \dots, s_{T-1}, a_{T-1}, s_T)$.

As in previous works motivated by probabilistic inference [33, 2], we introduce to the standard graphical model of an MDP a binary event variable $\mathcal{O}_t \in \{0, 1\}$, which represents whether the action in time step t is *optimal* or not. To derive the RL objective, we consider the marginal log-likelihood $\log \Pr(\mathcal{O} = 1|\pi_p)$, where π_p is a policy. Note that $\mathcal{O} = 1$ means $\mathcal{O}_t = 1$ for every time step. As is well known, we can decompose this using a variational distribution q of a trajectory as follows:

$$\begin{aligned}\log \Pr(\mathcal{O} = 1|\pi_p) &= \mathbb{E}_q \left[\log \Pr(\mathcal{O} = 1|\tau) - \log \frac{q(\tau)}{p(\tau)} + \log \frac{q(\tau)}{p(\tau|\mathcal{O} = 1)}\right] \\ &= \mathcal{J}(p, q) + D_{KL}(q(\tau) \,||\, p(\tau|\mathcal{O} = 1)),\end{aligned}$$

where $\mathcal{J}(p, q) := \mathbb{E}_q [\log \Pr(\mathcal{O} = 1|\tau)] - D_{KL}(q(\tau) \,||\, p(\tau))$ is the evidence lower bound (ELBO). Inference-based methods aim to find the parametric policy which maximizes the ELBO $\mathcal{J}(p, q)$.

There are several algorithmic design choices for q and $\Pr(\mathcal{O} = 1|\tau)$. Since any q is valid, a popular choice is the one that factorizes in the same way as p, that is,

$$p(\tau) = p(s_1) \prod_t p(s_{t+1}|s_t, a_t)\pi_p(a_t|s_t), \qquad q(\tau) = p(s_1) \prod_t p(s_{t+1}|s_t, a_t)\pi_q(a_t|s_t),$$

where π_p is a prior policy, and π_q is a variational posterior policy. We may also choose which policy (π_p or π_q) to parameterize. As for $\Pr(\mathcal{O} = 1|\tau)$, the most popular choice is the following one [20, 21, 2, 33, 47]:

$$\Pr(\mathcal{O} = 1|\tau) \propto \exp\left(\sum_{t=1}^{T} \eta^{-1} \mathcal{G}(s_t, a_t)\right),$$

where $\eta > 0$ is a temperature, and $\mathcal{G}$ is a function over $\mathcal{S} \times \mathcal{A}$, such as an immediate reward function r, Q-function Q^π, and advantage function A^π. While we employ $\exp(\cdot)$ in the present paper, there are alternatives [45, 46, 54, 65]: for example, Siegel et al. [54] and Wang et al. [65] consider an indicator function $f : x \in \mathbb{R} \mapsto \mathbb{1}[x \geq 0]$, whereas Oh et al. [45] considers the rectified linear unit $f : x \in \mathbb{R} \mapsto \max\{x, 0\}$.

Incorporating these design choices, we can rewrite the ELBO $\mathcal{J}(p, q)$ in a more explicit form as;

$$\mathcal{J}(\pi_p, \pi_q) = \sum_{t=1}^{T} \mathbb{E}_q \left[\eta^{-1} \mathcal{G}(s_t, a_t) - D_{KL}(\pi_q(\cdot|s_t) \mid\mid \pi_p(\cdot|s_t)) \right].$$

In the following section, we adopt this ELBO and consider its relaxation to the infinite-horizon setting. Then, starting from it, we derive MPO, AWR, and SAC.

4 A Unified View of Inference-based Off-Policy Actor-Critic Algorithms

In Sec. 3, we provided the explicit form of the ELBO $\mathcal{J}(\pi_p, \pi_q)$. However, in practice, it is difficult to maximize it as the expectation $\mathbb{E}_q$ depends on π_q. Furthermore, since we are interested in a finite-horizon setting, we replace $\sum_{t \in [T]} \mathbb{E}_q$ with $\mathbb{E}_{d_\pi(s)}$. Note that the latter expectation $\mathbb{E}_{d_\pi(s)}$ is taken with the unnormalized state distribution d_π under an arbitrary policy π. This is commonly assumed in previous works [2, 21]. This relaxation leads to the following optimization:

$$\max_{\pi_p, \pi_q} \mathcal{J}(\pi_p, \pi_q) \;\text{ s.t. } \int d_\pi(s) \int \pi_p(a|s)\, dads = 1 \text{ and } \int d_\pi(s) \int \pi_q(a|s)\, dads = 1\,, \quad (1)$$

where $\mathcal{J}(\pi_p, \pi_q) = \mathbb{E}_{d_\pi(s)}[\eta^{-1}\mathcal{G}(s, a) - D_{KL}(\pi_q \mid\mid \pi_p)]$. With this objective, we can regard recent popular SoTA off-policy algorithms, MPO [2], AWR [47], and SAC [21] as variants of a unified policy iteration method. The components are summarized in Table 1. We first explain how these algorithms can be grouped into two categories of approaches for solving Eq. 1, and then we elaborate additional implementation details each algorithm makes.

4.1 Unified Policy Iteration: Algorithmic Perspective

Eq. 1 allows the following algorithmic choices: how or if to parameterize π_p and π_q; what optimizer to use for them; and if to optimize them jointly, or individually while holding the other fixed. We show that the algorithms in Table 1 can be classified into two categories: Expectation-Maximization control (EM control) and direct Kullback-Leibler divergence minimization control (KL control).

4.1.1 Expectation-Maximization (EM) Control

This category subsumes MPO [2] (similarly REPS [49]), AWR [47], and AWAC [41], and we term it *EM control*. At high level, the algorithm non-parametrically solves for the variational posterior π_q while holding the parametric prior $\pi_p = \pi_{\theta_p}$ fixed (E-Step), and then optimize π_p holding the new π_q fixed (M-Step). This can be viewed as either a generic EM algorithm and as performing coordinate ascent on Eq. 1. We denote θ_p and π_q *after* iteration k of EM steps by $\theta_p^{(k)}$ and $\pi_q^{(k)}$, respectively.

In **E-step** at iteration k, we force the variational posterior policy $\pi_q^{(k)}$ to be close to the optimal posterior policy, i.e., the maximizer of the ELBO $\mathcal{J}(\pi_{\theta_p^{(k-1)}}, \pi_q)$ with respect to π_q. EM control converts the hard-constraint optimization problem in Eq. 1 to solving the following Lagrangian,

$$\begin{aligned}\mathcal{J}(\pi_q, \beta) = &\int d_\pi(s) \int \pi_q(a|s) \frac{\mathcal{G}(s,a)}{\eta}\, dads \\ &- \int d_\pi(s) \int \pi_q(a|s) \log \frac{\pi_q(a|s)}{\pi_{\theta_p^{(k-1)}}(a|s)}\, dads + \beta \left(1 - \int d_\pi(s) \int \pi_q(a|s)\, dads\right). \quad (2)\end{aligned}$$

We analytically obtain the solution of Eq. 2,

$$\pi_q^{(k)}(a|s) = Z(s)^{-1} \pi_{\theta_p^{(k-1)}}(a|s) \exp\left(\eta^{-1}\mathcal{G}(s,a)\right),$$

where $Z(s)$ is the partition function.

In **M-Step** at iteration k, we maximize the ELBO $\mathcal{J}(\pi_q^{(k)}, \pi_p)$ with respect to π_p. Considering the optimization with respect to π_p in Eq. 1 results in *forward* KL minimization, which is often referred to as a pseudo-likelihood or policy projection objective,

$$\max_{\theta_p} \mathbb{E}_{d_\pi(s)\pi_q^{(k)}(a|s)} \left[\log \pi_{\theta_p}(a|s)\right] = \max_{\theta_p} \mathbb{E}_{d_\pi(s)\pi_{\theta_p^{(k-1)}}(a|s)} \left[\frac{\log \pi_{\theta_p}(a|s)}{Z(s)} \exp\left(\eta^{-1}\mathcal{G}(s,a)\right)\right],$$

where we approximate $Z(s) \approx \frac{1}{M}\sum_{j=1}^{M} \exp(\eta^{-1}\mathcal{G}(s,a_j))$ with $a_j \sim \pi_{\theta_p^{(k-1)}}(\cdot|s)$ in practice.

4.1.2 Direct Kullback-Leibler (KL) Divergence Minimization Control

In contrast to the EM control in Sec. 4.1.1, we only optimize the variational posterior π_q while holding the prior π_p fixed. In this scheme, π_q is parameterized, so we denote it as π_{θ_q}. This leads to *KL control* [58, 61, 50, 11, 28, 20].

Equivalent to E-step in Sec. 4.1.1, we force the variational posterior policy π_{θ_q} to be close to the optimal posterior policy, i.e, the maximizer of $\mathcal{J}(\pi_p, \pi_q)$ with respect to π_q. The difference is that instead of analytically solving $\mathcal{J}(\pi_p, \pi_q)$ for π_q, we optimize $\mathcal{J}(\pi_p, \pi_{\theta_q})$ with respect to θ_q, which results in the following objective,

$$\max_{\theta_q} \mathbb{E}_{d_\pi(s)\pi_{\theta_q}(a|s)} \left[\eta^{-1}\mathcal{G}(s,a) - \log \frac{\pi_{\theta_q}(a|s)}{\pi_p(a|s)}\right].$$

4.1.3 "Optimal" Policies of EM and KL Control

While we formulate EM and KL control in a unified framework, we note that they have a fundamental difference in their definition of "optimal" policies. KL control fixes π_p and converges to a *regularized-optimal* policy in an exact case [14, 63]. In contrast, EM control continues updating both π_p and π_q, resulting in convergence to the *standard optimal* policy in an exact case [50]. We also note that EM control solves KL control as a sub-problem; for example, the first E-step exactly corresponds to a KL control problem, except the policy parameterization.

4.2 Unified Policy Iteration: Implementation Details

In this section, we explain the missing pieces of MPO, AWR, and SAC in Sec. 4.1. Additionally, we also describe the details of other algorithms [29, 48, 49, 40, 55, 35, 12, 51, 53] from the EM and KL control perspective. See Appendix B for the details.

4.2.1 MPO

Algorithm MPO closely follows the EM control scheme explained in Sec. 4.1.1, wherein $\pi_p = \pi_{\theta_p}$ is parametric, and π_q is non-parametric.

Implementation [π_q Update] This corresponds to the E-step. MPO uses a trust-region (TR) method. Concretely, it replaces the reverse KL penalty (second term) in the Lagrangian (Eq. 2) with a reverse KL constraint and analytically solves it for π_q. As a result, η is also optimized during the training by minimizing the dual of Eq. 2, which resembles REPS [49]:

$$g(\eta) = \eta\epsilon + \eta \log \mathbb{E}_{d_\pi(s)\pi_{\theta_p^{(k-1)}}(a|s)} \left[\exp\left(\eta^{-1} Q^{\pi_{\theta_p^{(k-1)}}}(s,a)\right)\right].$$

[π_p Update] This corresponds to the M-step. MPO uses a combination of Stochastic Gradient (SG) ascent and trust-region method based on a forward KL divergence similarly to TRPO [51]. Concretely, it obtains $\theta_p^{(k)}$ by maximizing the objective (Eq. 3) with respect to θ_p subject to $\mathbb{E}_{d_\pi(s)}[D_{KL}(\pi_{\theta_p^{(k-1)}}(\cdot|s) \;||\; \pi_{\theta_p}(\cdot|s))] \leq \varepsilon$. MPO further decompose this KL divergence to two terms, assuming Gaussian policies; one term includes only the mean vector of $\pi_{\theta_p}(\cdot|s)$, and the other includes only its covariance matrix. Abdolmaleki et al. [2] justify this as a log-prior of MAP estimation, which is assumed as a second-order approximation of KL divergence.

[$\mathcal{G}$ and $\mathcal{G}$ Estimate] MPO uses the Q-function of $\pi_{\theta_p^{(k-1)}}$ as $\mathcal{G}$ in the k-th E-step. It originally uses the Retrace update [39], while its practical implementation [26] uses a single-step Bellman update.

[π_θ] For a paramterized policy, MPO uses a Gaussian distribution with state-dependent mean vector and diagonal covariance matrix. The trust-region method in MPO's M-step heavily relies on this Gaussian assumption. Since a Gaussian distribution has an infinite support, MPO has a penalty term in its policy loss function that forces the mean of the policy to stay within the range of action space.

4.2.2 AWR and AWAC

Algorithm AWR slightly deviates from the EM control. In the M-step, AWR simply set $Z(s)$ to 1.

Implementation [π_q Update] This corresponds the E-step. AWR and AWAC analytically solves the Lagrangian (Eq. 2). In contrast to MPO, they don't use trust-region method.

[π_p Update] This corresponds to the M-step. At iteration k, AWR uses an average of all previous policies $\widetilde{\pi}_{p^k} := \frac{1}{k}\sum_{j=0}^{k-1}\pi_{\theta_p^{(j)}}$ instead of $\pi_{\theta_p^{(k-1)}}$ (cf. Eq. 3). In practice, the average policy $\widetilde{\pi}_{p^k}$ is replaced by samples of actions from a replay buffer, which stores action samples of previous policies.

[$\mathcal{G}$ and $\mathcal{G}$ Estimate] AWR uses the advantage function of $\widetilde{\pi}_{p^k}$ as $\mathcal{G}$ in the k-th E-step, and learns the state-value function of $\pi_{\theta_p^{(k-1)}}$ with TD(λ) [52]. Due to its choice of $\tilde{\pi}_{p^k}$, this avoids importance corrections [39]. In contrast, AWAC estimates the advantage via the Q-function with TD(0).

[π_θ] For a parameterized policy, both AWR and AWAC use a Gaussian distribution with state-dependent mean vector and state-independent diagonal covariance matrix (a constant one for AWR). As in MPO, they uses a penalty term to keep the mean of the policy within the range of action space.

4.2.3 SAC

Algorithm Contrary to MPO and AWR, SAC follows the KL control scheme explained in Sec. 4.1.2, wherein the variational posterior policy $\pi_q = \pi_{\theta_q}$ is parameterized, and the prior policy π_p is fixed to the uniform distribution over the action space $\mathcal{A}$. SAC uses as $\mathcal{G}$ a soft Q-function:

$$Q_{\text{soft}}^{\pi_{\theta_q}}(s_t, a_t) := r(s_t, a_t) + \gamma\mathbb{E}_{\pi_{\theta_q}}\left[V_{\text{soft}}^{\pi_{\theta_q}}(s_{t+1})\right],$$
$$V_{\text{soft}}^{\pi_{\theta_q}}(s_t) := V^{\pi_{\theta_q}}(s_t) + \mathbb{E}_{\pi_{\theta_q}}\left[\sum_{u=t}^{\infty}\gamma^{u-t}\eta\mathcal{H}(\pi_{\theta_q}(\cdot|s_t))\big|s_t\right],$$

with $\mathcal{H}(\pi_{\theta_q}(\cdot|s_t))$ being $-\mathbb{E}_{\pi_{\theta_q}}[\log\pi_{\theta_q}(a_t|s_t)|s_t]$.

Implementation [π_q and π_p Update] SAC performs stochastic gradient (SG) ascent updates of only π_q, where the objective function is shown in Eq. 1. SAC has no operation equivalent to M-step in MPO and AWR, since it keeps the prior policy π_p to the uniform distribution.

Similar to the temperature tuning in MPO, Haarnoja et al. [22] consider the dual function of entropy constrained Lagrangian and treat the temperature η as a Lagrange multiplier, which results in,

$$g(\eta) = -\eta\bar{\mathcal{H}} - \eta\mathbb{E}_{d_\pi(s)\pi_{\theta_q}(a|s)}\left[\log\pi_{\theta_q}(a|s)\right],$$

where $\bar{\mathcal{H}}$ is the target entropy to ensure that the entropy of the policy should be larger than it.

[$\mathcal{G}$ and $\mathcal{G}$ Estimate] SAC uses $Q_{\text{soft}}^{\pi_{\theta_q}}$ as $\mathcal{G}$, and TD(0)-like algorithm based on this soft Q-function. In policy evaluation, clipped double Q-learning [12], which retains multiple Q-functions (typically two) and takes the minimum, is employed to suppress the overestimation of Q value.

[π_θ] The policy of SAC is a squashed Gaussian distribution: SAC first samples a random variable u from a Gaussian distribution $\mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$ with state-dependent mean vector and diagonal covariance; then, it applies the tanh function to u and obtain the action $a \in [-1, 1]^{|\mathcal{A}|}$. (If a dimension of the action space is not $[-1, 1]$, an appropriate scaling and shift by an affine function is applied after the squashing.) Tanh squashing prevents out-of-bounds action.

4.3 Empirical Comparison

To evaluate the empirical performance of these off-policy algorithms (MPO, AWR, AWAC, and SAC), we compare their performances on Open AI Gym MuJoCo environments, namely, Hopper, Walker2d, HalfCheetah, Ant, Humanoid, and Swimmer, following Haarnoja et al. [22]. Figure 1 shows that SAC outperforms others in four environments (Hopper, Walker2d, HalfCheetah, and Humanoid),

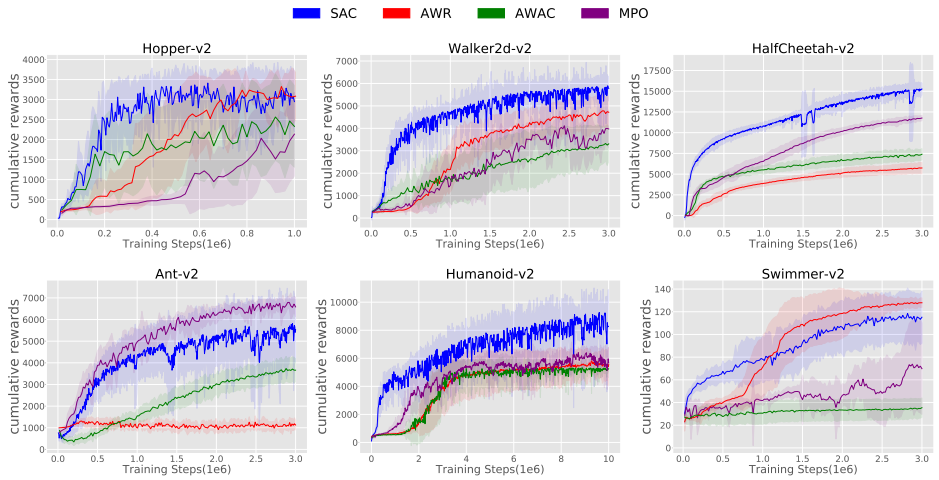

Figure 1: Benchmarking results on OpenAI Gym MuJoCo locomotion environments. All methods are run with 10 random seeds. SAC seems to perform consistently better.

	SAC (D)	SAC (S)	AWAC (D)	AWAC (S)	MPO (D)	MPO (S)
Hopper-v2	$\mathbf{3013 \pm 602}$	1601 ± 733	2329 ± 1020	2540 ± 755	2352 ± 959	2136 ± 1047
Walker2d-v2	$\mathbf{5820 \pm 411}$	1888 ± 922	3307 ± 780	3662 ± 712	4471 ± 281	3972 ± 849
HalfCheetah-v2	15254 ± 751	$\mathbf{15701 \pm 630}$	7396 ± 677	7226 ± 449	12028 ± 191	11769 ± 321
Ant-v2	5532 ± 1266	1163 ± 1326	3659 ± 523	3008 ± 375	$\mathbf{7179 \pm 190}$	6584 ± 455
Humanoid-v2	$\mathbf{8081 \pm 1149}$	768 ± 215	5243 ± 200	2738 ± 982	6858 ± 373	5709 ± 1081
Swimmer-v2	114 ± 21	$\mathbf{143 \pm 3}$	35 ± 8	38 ± 7	69 ± 29	70 ± 40

Table 2: Ablation of Clipped Double Q-Learning. (D) denotes algorithms with clipped double Q-learning, and (S) denotes without it. We test original SAC (D), AWAC (D), MPO (S) and some variants; SAC without clipped double Q (S), AWAC (S), and MPO with clipped double Q (D). SAC (S) beats SAC (D) in HalfCheetah and Swimmer, while it fails in Hopper, Walker, Ant and Humanoid, which implies that SAC (S) obtains a highly exploratory policy, since it fails in termination environments. The learning curves are shown in Appendix F.

while MPO in Ant and AWR in Swimmer achieves the best performance. Generally, SAC seems to perform consistently better. We extensively evaluate MPO, AWR, and SAC on the 28 tasks on DeepMind Control Suite and 3 MuJoCo manipulation tasks. See Appendix C and D for the details.

We reproduce all algorithms based on pytorch RL library [13], referring their original implementations [21, 47, 41, 26]. We share the codebase in supplemental materials, and will make it open-sourced in a de-anonymized version to facilitate further research.

5 Evaluation on Implementational Choices

In Sec. 4.3, SAC shows the notable results in most of environments, while we revealed the derivation and formulation of those methods resemble to each other. To specify the effect of each implementational detail on the performance, we experiment with extensive and careful one-by-one ablations on; (1) clipped double Q-learning, (2) distribution of the policy, (3) activation and normalization, and (4) network size. We also conclude several recommendations for the practitioners (Table 6).

5.1 Clipped Double Q-Learning

We hypothesize that clipped double Q-learning has a large effect on the notable performance of SAC, and could be transferable in EM control methods. To verify this, we test the effect of clipped double Q-learning. Instead of AWR, here we evaluate AWAC since it uses Q-function. Table 2 shows that single-Q SAC outperforms original one in HalfCheetah and Swimmer that do not have the termination of the episode, while struggles to learn in Hopper, Walker, Ant and Humanoid that have the episodic termination conditions. A hypothesis is that single-Q SAC obtains a more exploratory policy due to larger overestimation bias, which can help in environments where explorations are safe, but hurt in environments where reckless explorations lead to immediate terminations. In contrast, clipped double Q-learning does not help MPO or AWAC as significantly as SAC; most results do not change or lead to slight improvements over the originals. This suggests that clipped double Q-learning might be a co-dependent and indispensable choice to KL control methods.

Recommendation Use clipped double Q-learning as a default choice, but you can omit it in EM control methods or non-terminal environments, such as HalfCheetah or Swimmer.

	SAC (w/)	SAC (w/o)	AWR (w/)	AWR (w/o)	MPO (w/)	MPO (w/o)
Hopper-v2	3013 ± 602	6 ± 10	2709 ± 905	$\mathbf{3085 \pm 593}$	2149 ± 849	2136 ± 1047
Walker2d-v2	$\mathbf{5820 \pm 411}$	$-\infty$	3295 ± 335	4717 ± 678	3167 ± 815	3972 ± 849
HalfCheetah-v2	$\mathbf{15254 \pm 751}$	$-\infty$	3653 ± 652	5742 ± 667	9523 ± 312	11769 ± 321
Ant-v2	5532 ± 1266	$-\infty$	445 ± 106	1127 ± 224	2880 ± 306	$\mathbf{6584 \pm 455}$
Humanoid-v2	$\mathbf{8081 \pm 1149}$	$108 \pm 82^{\dagger}$	$2304 \pm 1629^{\dagger}$	5573 ± 1020	6688 ± 192	5709 ± 1081
Swimmer-v2	114 ± 21	28 ± 11	121 ± 3	$\mathbf{128 \pm 4}$	110 ± 42	70 ± 40

Table 3: Ablation of Tanh transformation (†numerical error happens during training). We test SAC without tanh squashing, AWR with tanh, and MPO with tanh. SAC without tanh transform results in drastic degradation of the performance, which can be caused by the maximum entropy objective that encourages the maximization of the covariance. In contrast, EM Control methods don't enjoy the performance gain from the tanh-squashed policy, which seems a less transferable choice. The learning curves are shown in Appendix F.

5.2 Distribution of the Policy

Another hypothesis is that the tanh-squashed policy (last column in Table 1) is an important and transferable design choice. We compare SAC without tanh transform (with MPO action penalty instead) to the original one, which results in drastic degradation (Table 3). This can be caused by the maximum entropy objective that encourages maximizing the covariance. These observations suggest that the high performance of SAC seems to highly depend on the implementational choice of the policy distribution. In contrast, MPO and AWR with tanh squashing don't enjoy such large performance gain, or archive worse cumulative rewards. This also suggests that tanh-squashed policy might be highly co-adapted in SAC and less transferable to EM control methods. Note that for EM control, we must clip the actions to keep them within the supports of distributions; $a \in [-1+\epsilon, 1-\epsilon]^{|\mathcal{A}|}$. We found that no clipping cause significant numerical instability. See Appendix G for the details.

Recommendation Use the original distributions for each algorithm. If you use the tanh-squashed Gaussian in EM control methods, clip the action to be careful for the numerical instability.

5.3 Activation and Normalization

The implementation of MPO [26] has some code-level detailed choices; ELU activation [7] and layer normalization [4]. We hypothesize that these, sometimes unfamiliar, implementational details stabilize the learning process, and contribute the performance of MPO most. Especially in Ant (Sec. 4.3), MPO significantly outperforms SAC. To investigate this much deeper, we add them into SAC or AWR and remove them from MPO, while maintaining the rest of the implementations. Table 4 shows that layer normalization can contribute to a significantly higher performance of SAC in Ant, and replacing ReLU with ELU also improves performance a lot in Swimmer, where AWR is the best in Figure 1. In contrast, the performance of MPO drastically collapsed when we just removed ELU and layer normalization. This observation suggests that these implementations are not only indispensable choices for MPO, but are transferable and beneficial to both KL and EM control methods. Additionally, we tested incorporating ELU and layer normalization to SAC in 12 DM Control tasks where MPO outperformed SAC, and observed that they again often benefit SAC performances substantially. See Appendix C for the details.

Recommendation It is worth considering to replace the activation function from ReLU to ELU and incorporate the layer normalization, to archive the best performance in several tasks. Both of them are transferable between EM and KL control methods.

5.4 Network Size

While on-policy algorithms such as PPO and TRPO can use the common network architecture as in prior works [9, 24], the architecture that works well in all the off-policy inference-based methods is not obvious, and the RL community doesn't have an agreeable default choice. To validate the dependency on the networks among the inference-based methods, we exchange the size and number of hidden layers in the policy and value networks. We denote the network size of MPO ((256, 256, 256) for policy and (512, 512, 256) for value) as large (L), of SAC ((256, 256) for policy and value) as middle (M), and of AWR ((128, 64) for policy and value) as small (S) (see Appendix A for the details). Table 5 illustrates that SAC with large network seems to have better performance. However, as shown in Appendix F (Figure 8), the learning curve sometimes becomes unstable. While the certain degree of robustness to the network size is observed in SAC, EM control methods, MPO and AWR, seem fragile and more dependent on the specific network size. This trend is remarkable in MPO. AWR (M) or (L) also struggles to learn in high-dimensional state tasks, such as Ant (111 dim),

	Hopper-v2	Walker2d-v2	HalfCheetah-v2	Ant-v2	Humanoid-v2	Swimmer-v2
SAC	3013 ± 602	$\mathbf{5820 \pm 411}$	15254 ± 751	5532 ± 1266	8081 ± 1149	114 ± 21
SAC-E$^+$	2337 ± 903	5504 ± 431	$\mathbf{15350 \pm 594}$	6457 ± 828	$\mathbf{8196 \pm 892}$	$\mathbf{146 \pm 7}$
SAC-L$^+$	2368 ± 179	5613 ± 762	13074 ± 2218	$\mathbf{7349 \pm 176}$	8146 ± 470	99 ± 18
SAC-E$^+$L$^+$	1926 ± 417	5751 ± 400	12555 ± 1259	7017 ± 132	7687 ± 1385	143 ± 9
MPO	2136 ± 1047	3972 ± 849	11769 ± 321	6584 ± 455	5709 ± 1081	70 ± 40
MPO-E$^-$	2700 ± 879	3553 ± 1145	11638 ± 664	5917 ± 702	4870 ± 1917	108 ± 28
MPO-L$^-$	824 ± 250	2413 ± 1352	6064 ± 4596	2135 ± 2988	5039 ± 838	$-\infty$
MPO-E$^-$L$^-$	843 ± 168	1708 ± 663	-1363 ± 20965	807 ± 2351	5566 ± 787	$-\infty$
AWR	3085 ± 593	4717 ± 678	5742 ± 667	1127 ± 224	5573 ± 1020	128 ± 4
AWR-E$^+$	1793 ± 1305	4418 ± 319	5910 ± 754	2288 ± 715	6708 ± 226	128 ± 4
AWR-L$^+$	2525 ± 1130	4900 ± 671	5391 ± 232	639 ± 68	5962 ± 376	129 ± 2
AWR-E$^+$L$^+$	$\mathbf{3234 \pm 118}$	4906 ± 304	6081 ± 753	2283 ± 927	6041 ± 270	130 ± 1

Table 4: Incorporating ELU/layer normalization into SAC and AWR. E$^+$/L$^+$ indicates adding, and E$^-$/L$^-$ indicates removing ELU/layer normalization. Introducing layer normalization or ELU into SAC improves the performances in Ant (beating MPO), Swimmer (beating AWR), HalfCheetah, and Humanoid. AWR also shows the improvement in several tasks. MPO removing layer normalization largely drops its performance. Both implementation details seems transferable between EM and KL control methods.

	Hopper-v2	Walker2d-v2	HalfCheetah-v2	Ant-v2	Humanoid-v2	Swimmer-v2
SAC (L)	2486 ± 746	3188 ± 2115	$\mathbf{16528 \pm 183}$	$\mathbf{7495 \pm 405}$	$\mathbf{8255 \pm 578}$	118 ± 26
SAC (M)	3013 ± 602	$\mathbf{5820 \pm 411}$	15254 ± 751	5532 ± 1266	8081 ± 1149	114 ± 21
SAC (S)	$\mathbf{3456 \pm 81}$	4939 ± 284	12241 ± 400	3290 ± 691	7724 ± 497	59 ± 11
MPO (L)	2136 ± 1047	3972 ± 849	11769 ± 321	6584 ± 455	5709 ± 1081	70 ± 40
MPO (M)	661 ± 79	1965 ± 1426	$-\infty$	5192 ± 538	6015 ± 771	81 ± 28
MPO (S)	430 ± 99	2055 ± 990	5003 ± 1567	3587 ± 957	4745 ± 1428	59 ± 28
AWR (L)	3221 ± 193	4688 ± 648	4360 ± 542	35 ± 43	665 ± 54	$\mathbf{133 \pm 3}$
AWR (M)	2816 ± 910	4826 ± 547	5538 ± 720	413 ± 117	3849 ± 1647	$\mathbf{133 \pm 2}$
AWR (S)	3085 ± 593	4717 ± 678	5742 ± 667	1127 ± 224	5573 ± 1020	128 ± 4

Table 5: The performance of each algorithm with different network size. (S) stands for the small network size from AWR, (M) for the middle network size from SAC, and (L) for the large network size from MPO. Generally, SAC, a KL control method seems more robust to the network size than EM control methods; MPO and AWR.

or Humanoid (376 dim). The results also imply that, in contrast to on-policy methods [3], the network size in the off-policy inference-based methods seems a less transferable choice.

Recommendation For SAC, use medium size network. Large size will also work, but the learning curve might be unstable. For MPO, we strongly recommend to stick to large size, because it is very sensitive to the network size. For AWR, using small size is a better choice, especially in high-dimensional state tasks, such as Ant, or Humanoid.

6 Conclusion

In this work, we present a taxonomy of inference-based algorithms, and successfully identify algorithm-specific as well as algorithm-independent implementation details that cause substantial performance improvements. We first reformulated recent inference-based off-policy algorithms – such as MPO, AWR and SAC – into a unified mathematical objective and exhaustively clarified the algorithmic and implementational differences. Through precise ablation studies, we empirically show that implementation choices like tanh-squashed distribution and clipped double Q-learning are highly co-adapted to KL control methods (e.g. SAC), and difficult to benefit in EM control methods (e.g. MPO or AWR). As an example, the network architectures of inference-based off-policy algorithms, especially EM controls, seem more co-dependent than on-policy methods like PPO or TRPO, which therefore have significant impacts on the overall algorithm performances and need to be carefully tuned per algorithm. Such dependence of each algorithmic innovation on specific hand-tuned implementation details makes accurate performance gain attributions and cumulative build-up of research insights difficult. In contrast, we also find that some code-level implementation details, such as ELU and layer normalization, are not only indispensable choice to MPO, but also transferable and beneficial to SAC substantially. We hope our work can encourage more works that study precisely the impacts of algorithmic properties and empirical design choices, not only for one type of algorithms, but also across a broader spectrum of deep RL algorithms.

	MPO	**AWR**	**SAC**
Clipped Double Q [5.1]	△	△	◯
Tanh Gaussian [5.2]	△	△	◯
ELU & LayerNorm [5.3]	◯	○	○
Large Network [5.4]	◯	△	○
Medium Network [5.4]	×	△	○
Small Network [5.4]	×	○	△

Table 6: Intuitive summary of the ablations. ◯ stands for indispensable choice, ○ stands for recommended choice, △ stands for not much different or worse choice than expected, and × stands for un-recommended choice. See **Recommendation** for the details.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Jonas Degrave, Steven Bohez, Yuval Tassa, Dan Belov, Nicolas Heess, and Martin Riedmiller. Relative entropy regularized policy iteration. *arXiv preprint arXiv:1812.02256*, 2018.

[2] 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.

[3] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

[6] Kamil Ciosek and Shimon Whiteson. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 2020.

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*, 2016.

[8] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, 2016.

[9] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2019.

[10] Matthew Fellows, Anuj Mahajan, Tim GJ Rudner, and Shimon Whiteson. Virel: A variational inference framework for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

[11] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Conference on Uncertainty in Artificial Intelligence*, 2016.

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

[13] Yasuhiro Fujita, Prabhat Nagarajan, Toshiki Kataoka, and Takahiro Ishikawa. Chainerrl: A deep reinforcement learning library. *Journal of Machine Learning Research*, 2021.

[14] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized Markov decision processes. In *International Conference on Machine Learning*, 2019.

[15] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, 2020.

[16] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

[17] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, 2016.

[18] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-Prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations*, 2017.

[19] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

[20] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, 2017.

[21] 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*, 2018.

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

[23] Danijar Hafner, Pedro A Ortega, Jimmy Ba, Thomas Parr, Karl Friston, and Nicolas Heess. Action and perception as divergence minimization. *arXiv preprint arXiv:2009.01791*, 2020.

[24] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, 2017.

[25] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.

[26] Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

[27] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

[28] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, 2017.

[29] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, 2008.

[30] 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*, 2019.

[31] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, 2020.

[32] Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In *Advances in Neural Information Processing Systems*, 2020.

[33] Sergey Levine. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[34] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[35] 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.

[36] Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-depedent control variates for policy optimization via stein's identity. *arXiv preprint arXiv:1710.11198*, 2017.

[37] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *arXiv preprint arXiv:1608.04471*, 2016.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[39] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

[40] Ofir Nachum, Mohammad Norouzi, and Dale Schuurmans. Improving policy gradient by exploring under-appreciated rewards. In *International Conference on Learning Representations*, 2017.

[41] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[42] Gerhard Neumann et al. Variational inference for policy search in changing situations. In *International Conference on Machine Learning*, 2011.

[43] Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, 2016.

[44] Brendan O'Donoghue, Ian Osband, and Catalin Ionescu. Making sense of reinforcement learning and probabilistic inference. In *International Conference on Learning Representations*, 2020.

[45] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, 2018.

[46] Masashi Okada and Tadahiro Taniguchi. Variational inference mpc for bayesian model-based reinforcement learning. In *Conference on Robot Learning*, 2019.

[47] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-Weighted Regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[48] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, 2007.

[49] Jan Peters, Katharina Mülling, and Yasemin Altün. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

[50] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *International Joint Conference on Artificial Intelligence*, 2012.

[51] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

[52] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

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

[54] Noah Y. Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin A. Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

[55] H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Heess, Dan Belov, Martin Riedmiller, and Matthew M. Botvinick. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. In *International Conference on Learning Representations*, 2020.

[56] Yunhao Tang and Alp Kucukelbir. Hindsight expectation maximization for goal-conditioned reinforcement learning. *arXiv preprint arXiv:2006.07549*, 2020.

[57] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[58] Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in Neural Information Processing Systems*, 2006.

[59] Emanuel Todorov. General duality between optimal control and estimation. In *IEEE Conference on Decision and Control*, 2008.

[60] Marc Toussaint. Robot trajectory optimization using approximate inference. In *International Conference on Machine Learning*, 2009.

[61] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *International conference on Machine learning*, 2006.

[62] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. In *International Conference on Machine Learning*, 2018.

[63] Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of kl regularization in rl. In *Advances in Neural Information Processing Systems*, 2021.

[64] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[65] Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

[66] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. *arXiv preprint arXiv:1803.07246*, 2018.

[67] 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]
   (b) Did you describe the limitations of your work? [Yes] We added the failure cases of ablations that end up the insufficient coverage and the unclear insights in Appendix G.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Actually, we conducted exhaustive evaluations through the enormous experiments, which might lead to force the future research to spend much computing resources. We hope our empirical observations and recommendations help the practitioners to explore the explosive configuration space.
   (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 shared the codebase in supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Our experimental results were averaged among 10 random seeds.
   (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] See Appendix A.
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 cited the authors of the codebase in Sec. 4.3.
   (b) Did you mention the license of the assets? [N/A]
   (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]