Deterministic Uncertainty Propagation for Improved Model-Based Offline Reinforcement Learning

Abdullah Akgül **Manuel Haußmann** **Melih Kandemir**
Department of Mathematics and Computer Science
University of Southern Denmark
Odense, Denmark
{akgul,haussmann,kandemir}@imada.sdu.dk

Abstract

Current approaches to model-based offline reinforcement learning often incorporate uncertainty-based reward penalization to address the distributional shift problem. These approaches, commonly known as pessimistic value iteration, use Monte Carlo sampling to estimate the Bellman target to perform temporal difference-based policy evaluation. We find out that the randomness caused by this sampling step significantly delays convergence. We present a theoretical result demonstrating the strong dependency of suboptimality on the number of Monte Carlo samples taken per Bellman target calculation. Our main contribution is a deterministic approximation to the Bellman target that uses progressive moment matching, a method developed originally for deterministic variational inference. The resulting algorithm, which we call Moment Matching Offline Model-Based Policy Optimization (MOMBO), propagates the uncertainty of the next state through a nonlinear Q-network in a deterministic fashion by approximating the distributions of hidden layer activations by a normal distribution. We show that it is possible to provide tighter guarantees for the suboptimality of MOMBO than the existing Monte Carlo sampling approaches. We also observe MOMBO to converge faster than these approaches in a large set of benchmark tasks.

1 Introduction

Offline reinforcement learning (Lange et al., 2012; Levine et al., 2020) aims to solve a control task using an offline dataset without interacting with the target environment. Such an approach is essential in cases where environment interactions are expensive or risky (Shortreed et al., 2011; Singh et al., 2020; Nie et al., 2021; Micheli et al., 2022). Directly adapting traditional online off-policy reinforcement learning methods to offline settings often results in poor performance due to the adverse effects of the distributional shift caused by the policy updates (Fujimoto et al., 2019; Kumar et al., 2019). The main reason for the incompatibility is the rapid accumulation of overestimated action-values during policy improvement. Pessimistic Value Iteration (PEVI) (Jin et al., 2021) offers a theoretically justified solution to this problem that penalizes the estimated rewards of synthetic state transitions proportionally to the uncertainty of the predicted next state. The framework encompasses many state-of-the-art offline reinforcement learning algorithms as special cases (Yu et al., 2020; Sun et al., 2023).

Model-based offline reinforcement learning approaches first fit a probabilistic model on the real state transitions and then supplement the real data with synthetic samples generated from this model throughout policy search (Janner et al., 2019). One line of work addresses distributional shift by constraining policy learning (Kumar et al., 2019; Fujimoto and Gu, 2021) where the policy is trained to mimic the behavior policy and is penalized based on the discrepancy between its actions and those

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-2247

of the behavior policy, similarly to behavioral cloning. A second strategy introduces conservatism to training by *(i)* perturbing the training objective with a high-entropy behavior policy (Kumar et al., 2020; Yu et al., 2021), *(ii)* penalizing state-action pairs proportionally to their estimated variance (Yu et al., 2020; An et al., 2021; Bai et al., 2022; Sun et al., 2023), or *(iii)* adversarially training the environment model to minimize the value function (Rigter et al., 2022) to prevent overestimation in policy evaluation for out-of-domain state-action pairs.

In the offline reinforcement learning literature, uncertainty-driven approaches exist for both model-free (An et al., 2021; Bai et al., 2022) and model-based settings (Yu et al., 2020; Sun et al., 2023), all aimed at learning a pessimistic value function (Jin et al., 2021) by penalizing it with an uncertainty estimator. The impact of uncertainty quantification has been investigated in both online (Abbas et al., 2020) and offline (Lu et al., 2021) scenarios, particularly in model-based approaches, which is the focus of our work.

Despite the strong theoretical evidence suggesting that the quality of the Bellman target uncertainties has significant influence on model quality (O'Donoghue et al., 2018; Luis et al., 2023; Jin et al., 2021), the existing implementations rely on Monte Carlo samples and heuristics-driven uncertainty quantifiers being used as reward penalizers (Yu et al., 2020; Sun et al., 2023) which results in a high degree of randomness that manifests itself as significant delays in model convergence. Figure 1 demonstrates why this is the case by a toy example. The uncertainty around the estimated action-value of the next state does not shrink even after taking 10000 samples on the next state and passing them through a Q-network. Our moment matching-based approach predicts a similar mean with significantly smaller variance at the cost of only two Monte Carlo samples.

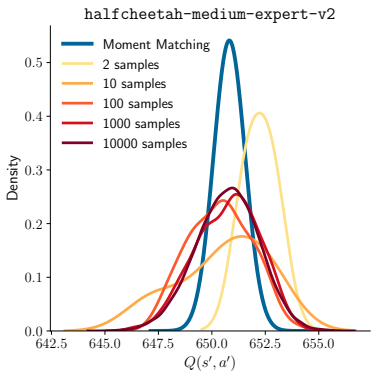

Figure 1: *Moment Matching versus Monte Carlo Sampling.* Moment matching offers sharp estimates of the action-value of the next state at the cost of only two forward passes through a critic network. A similar sharpness cannot be reached even with 10000 Monte Carlo samples, which is 5000 times more costly. See Appendix C.1.1 for details.

We identify the randomness introduced by Bellman target approximation via Monte Carlo sampling as the primary limiting factor for the performance of the PEVI approaches used in offline reinforcement learning. We have three main contributions:

- *(i)* We present a new algorithm that employs progressive moment matching, an idea originally developed for deterministic variational inference of Bayesian neural networks (Wu et al., 2019a), for the first time to propagate deterministically environment model uncertainties through Q-function estimates. We refer to this algorithm as *Moment Matching Offline Model-Based Policy Optimization* (MOMBO).
- *(ii)* We develop novel suboptimality guarantees for both MOMBO and existing sampling-based approaches highlighting that MOMBO is provably more efficient.
- *(iii)* We present comprehensive experiment results showing that MOMBO significantly accelerates training convergence while maintaining asymptotic performance.

2 Preliminaries

Reinforcement learning. We define a Markov Decision Process (MDP) as a tuple $\mathcal{M} \triangleq (\mathcal{S}, \mathcal{A}, r, \mathrm{P}, \mathrm{P}_1, \gamma, H)$ where $\mathcal{S}$ represents the state space and $\mathcal{A}$ denotes the action space. We have $r : \mathcal{S} \times \mathcal{A} \to [0, R_{\max}]$ as a bounded deterministic reward function and $\mathrm{P} : \mathcal{S} \times \mathcal{A} \times \Delta(\mathcal{S}) \to [0, 1]$ as the probabilistic state transition kernel, where $\Delta(\mathcal{S})$ denotes the probability simplex defined over the state space. The MDP has an initial state distribution $\mathrm{P}_1 \in \Delta(\mathcal{S})$, a discount factor $\gamma \in (0, 1)$, and an episode length H. We use deterministic policies $\pi : \mathcal{S} \to \mathcal{A}$ and deterministic rewards in analytical demonstrations for simplicity and without loss of generality. Our results extend straightforwardly to probabilistic policies and reward functions. The randomness on the next state may result from the system stochasticity or the estimation error of a probabilistic model trained on the data collected from a deterministic environment. We define the action-value of a policy

π for a state-action pair (s, a) as

$$Q_\pi(s,a) \triangleq \mathbb{E}_\pi \left[\sum_{i=h}^{H} \gamma^{i-h} r(s_i, a_i) \middle| s_h = s, a_h = a \right].$$

This function maps to the range $[0, {}^{R_{\max}}/_{1-\gamma}]$.

Offline reinforcement learning algorithms perform policy search using an offline collected dataset $\mathcal{D} = \{(s, a, r, s')\}$ from the target environment via a behavior policy π_β. The goal is to find a policy π that minimizes the suboptimality, defined as $\texttt{SubOpt}(\pi; s) \triangleq Q_{\pi^*}(s, \pi^*(s)) - Q_\pi(s, \pi(s))$ for an initial state s and optimal policy π^*. For brevity, we omit the dependency on s in $\texttt{SubOpt}(\cdot)$. Model-based reinforcement learning approaches often train state transition probabilities and reward functions on the offline dataset by maximum likelihood estimation with an assumed density, modeling these as an ensemble of heteroscedastic neural networks (Lakshminarayanan et al., 2017). Mainstream model-based offline reinforcement learning methods adopt the Dyna-style (Sutton, 1990; Janner et al., 2019), which suggests training an off-the-shelf model-free reinforcement learning algorithm on synthetic data generated from the learned environment model $\widehat{\mathcal{D}}$ using initial states from $\mathcal{D}$. It has been observed that mixing minibatches collected from synthetic and real datasets improves performance in both online (Janner et al., 2019) and offline (Yu et al., 2020, 2021) settings.

Scope and problem statement. We focus on model-based offline reinforcement learning due to its superior performance over model-free methods (Yu et al., 2020; Kidambi et al., 2020; Yu et al., 2021; Rigter et al., 2022; Sun et al., 2023). The primary challenge in offline reinforcement learning is *distributional shift* caused by a limited coverage of the state-action space. When the policy search algorithm probes unobserved state-action pairs during training, significant value approximation errors arise. In standard supervised learning, such errors diminish as new observations are collected. However, in reinforcement learning, overestimation errors are exploited by the policy improvement step and accumulate, resulting in a phenomenon known as the *overestimation bias* (Thrun and Schwartz, 1993). Techniques developed to overcome this problem in the online setting such as min-clipping (Fujimoto et al., 2018) are insufficient for offline reinforcement learning. Algorithms that use reward penalization proportional to the estimated uncertainty of an unobserved next state are commonly referred to as *Pessimistic Value Iteration* (Yu et al., 2020; Sun et al., 2023). These algorithms are grounded in a theory that provides performance improvement guarantees by bounding the variance of next state predictions (Jin et al., 2021; Uehara and Sun, 2022). We demonstrate the theoretical and practical limitations of PEVI-based approaches and address them with a new solution.

Bellman operators. Both our analysis and the algorithmic solution build on a problem that arises from the inaccurate estimation of the Bellman targets in critic training. The error stems from the fact that during training the agent has access only to the *sample Bellman operator*,

$$\widehat{\mathbb{B}}_\pi Q(s,a,s') \triangleq r(s,a) + \gamma Q(s', \pi(s')),$$

where s' is a Monte Carlo sample of the next state. However, the actual quantity of interest is the *exact Bellman operator* that is equivalent to the expectation of the sample Bellman operator,

$$\mathbb{B}_\pi Q(s,a) \triangleq \mathbb{E}_{s' \sim \mathrm{P}(\cdot|s,a)} \left[\widehat{\mathbb{B}}_\pi Q(s,a,s') \right].$$

3 Pessimistic value iteration algorithms

A pessimistic value iteration algorithm (Jin et al., 2021), denoted as $\mathbb{A}_{\texttt{PEVI}}(\widehat{\mathbb{B}}^\Gamma_\pi Q, \widehat{\mathrm{P}})$, performs Dyna-style model-based learning using the following *pessimistic sample Bellman target* for temporal difference updates during critic training:

$$\widehat{\mathbb{B}}^\Gamma_\pi Q(s,a,s') \triangleq \widehat{\mathbb{B}}_\pi Q(s,a,s') - \Gamma^Q_{\widehat{\mathrm{P}}}(s,a),$$

where $\Gamma^Q_{\widehat{\mathrm{P}}}(s,a)$ admits a learned state transition kernel $\widehat{\mathrm{P}}$ and a value function Q to map a state-action pair to a penalty score. Notably, this penalty term does not depend on an observed or previously sampled s' as it quantifies the uncertainty around s' using $\widehat{\mathrm{P}}$. The rest follows as running an off-the-shelf model-free online reinforcement learning algorithm, for instance Soft Actor-Critic (SAC)

https://doi.org/10.52202/079017-2247

(Haarnoja et al., 2018), on a replay buffer that comprises a blend of real observations and synthetic data generated from $\widehat{\mathrm{P}}$ using the recent policy of an intermediate training stage. We study the following two PEVI variants in this paper due to their representative value:

i) *Model-based Offline Policy Optimization (MOPO)* (Yu et al., 2020) directly penalizes the uncertainty of a state as inferred by the learned environment model $\Gamma^Q_{\widehat{\mathrm{P}}}(s,a) \triangleq \mathrm{var}_{\widehat{\mathrm{P}}}[s']$. MOPO belongs to the family of methods where penalties are based on the uncertainty of the next state.

ii) *MOdel-Bellman Inconsistency penalized offLinE Policy Optimization (MOBILE)* (Sun et al., 2023) propagates the uncertainty of the next state to the Bellman target through Monte Carlo sampling and uses it as a penalty:

$$\Gamma^Q_{\widehat{\mathrm{P}}}(s,a) \triangleq \widehat{\mathrm{var}}^N_{s'\sim\widehat{\mathrm{P}}}[\widehat{\mathbb{B}}_\pi Q(s,a,s')]$$

where $\widehat{\mathrm{var}}^N_{s'\sim\widehat{\mathrm{P}}}$ denotes the empirical variance of the quantity in brackets with respect to N samples drawn i.i.d. from $\widehat{\mathrm{P}}$. MOBILE represents the family of methods that penalize rewards based on the uncertainty of the Bellman target.

Both approaches approximate the mean of the Bellman target by evaluating the sample Bellman operator $\widehat{\mathbb{B}}_\pi Q(s,a,s')$ with a single s' available either from real environment interaction within the dataset or a single sample taken from $\widehat{\mathrm{P}}$. The following theorem establishes the critical role the penalty term plays in the model performance.

Theorem 1 (*Suboptimality of PEVI (Jin et al., 2021)*)**.** *For any π derived with $\mathbb{A}_{\mathit{PEVI}}(\widehat{\mathbb{B}}^\Gamma_\pi Q, \widehat{\mathrm{P}})$ that satisfies*

$$|\mathbb{B}_\pi Q(s,a) - \widehat{\mathbb{B}}_\pi Q(s,a,s')| \le \Gamma^Q_{\widehat{\mathrm{P}}}(s,a), \qquad \forall (s,a) \in \mathcal{S}\times\mathcal{A},$$

with probability at least $1-\delta$ for some error tolerance $\delta \in (0,1)$ where $s' \sim \mathrm{P}(\cdot|s,a)$, the following inequality holds for any initial state $s_1 \in \mathcal{S}$:

$$\mathtt{SubOpt}(\pi) \le 2\sum_{i=1}^{H} \mathbb{E}_{\pi^*}\left[\Gamma^Q_{\widehat{\mathrm{P}}}(s_i,a_i)\middle| s_1\right].$$

When deep neural networks are used as value function approximators, calculating their variance becomes analytically intractable even for normally distributed inputs. Consequently, reward penalties $\Gamma^Q_{\widehat{\mathrm{P}}}$ are typically derived from Monte Carlo samples, which are prone to high estimator variance (see Figure 1). Our key finding is that using high-variance estimates for reward penalization introduces three major practical issues in the training process of offline reinforcement learning algorithms:

(i) The information content of the distorted gradient signal shrinks, causing delayed convergence to the asymptotic performance.

(ii) The first two moments of the Bellman target are poorly approximated for feasible sample counts.

(iii) Larger confidence radii need to be used to counter the instability caused by *(i)* and the high overestimation risk caused by the inaccuracy described in *(ii)*, which unnecessarily restricts model capacity.

3.1 Theoretical analysis of sampling-based PEVI

We motivate the benefit of our deterministic uncertainty propagation scheme by demonstrating the prohibitive relationship of the sample Bellman operator when used in the PEVI context. The analysis of the distribution around the Bellman operator can be characterized as follows. We are interested in the distribution of a deterministic map $y = f(x)$ for a normally distributed input $x \sim \mathcal{N}(\mu,\sigma^2)$. We analyze this complex object via a surrogate of it. For a sample set $S_N = \{x_i\}_{i=1}^N$ obtained i.i.d. from $\mathcal{N}(\mu,\sigma^2)$, let μ_N and σ^2_N be its empirical mean and variance. Now consider the following two random variables $y_N = \frac{1}{N}\sum_{i=1}^N f(x_i)$ and $y'_N = f(x')$ for $x' \sim \mathcal{N}(\mu_N, \sigma^2_N)$. Note that y_1

and y'_1 are equal in distribution. Furthermore, both y_N and y'_N converge to the true y as N tends to infinity. We perform our analysis on y'_N where analytical guarantees are easier to build. Furthermore, y'_N is a tighter approximation of both y_1 and y'_1. We construct the following suboptimality proof for the PEVI algorithmic family that approximates the uncertainty around the Bellman operator in the way y'_N is built. In this theorem and elsewhere, A_l stands for the weights of the l-th layer of a Multi-Layer Perceptron (MLP) which has 1-Lipschitz activation functions and $\|\cdot\|$ denotes $L1$-norm. See Appendix B.2 for the proof.

Theorem 2 (*Suboptimality of sampling-based PEVI*). *For any policy π_{MC} learned by $\mathbb{A}_{PEVI}(\widehat{\mathbb{B}}^{\Gamma}_{\pi}Q, \widehat{\mathrm{P}})$ using N Monte Carlo samples to approximate the Bellman target with respect to an action-value network defined as an L-layer MLP with 1-Lipschitz activation functions, the following inequality holds for any error tolerance $\delta \in (0,1)$ with probability at least $1-\delta$*

$$\texttt{SubOpt}(\pi_{MC}) \leq 2H \prod_{l=1}^{L} \|A_l\| \sqrt{-\frac{8\log(\delta/4)R_{\max}^2}{\lfloor N/2 \rfloor (1-\gamma)^2}}.$$

The bound in Theorem 2 is prohibitively loose since $R_{\max}^2/(1-\gamma)^2$ is a large number in practice. For instance, the maximum realizable step reward in the `HalfCheetah-v4` environment of the MuJoCo physics engine (Todorov et al., 2012) is at least around 11.7 according to Hui et al. (2023). Choosing the usual $\gamma = 0.99$, we obtain $R_{\max}^2/(1-\gamma)^2 = 1.37 \times 10^6$. Furthermore, as the bound depends on the number of samples for the next state N, it becomes looser as $N \to 1$. The bound is not defined for $N = 1$, but the loosening trend is clear.

4 MOMBO: Moment matching offline model-based policy optimization

Our key contribution is the finding that quantifying the uncertainty around the Bellman target brings significant benefits to the model-based offline reinforcement learning setting. We obtain a deterministic approximation using a moment matching technique originally developed for Bayesian inference. This method propagates the estimated uncertainty of the next state s' obtained from a learned environment model $\widehat{\mathrm{P}}$ through an action-value network by alternating between an affine transformation of a normally distributed input to another normally distributed linear activation and projecting the output of a nonlinear activation to a normal distribution by analytically calculating its first two moments. The resulting normal distributed output is then used to build a lower confidence bound on the Bellman target, which is an equivalent interpretation of reward penalization. Such a deterministic approximation is both a more accurate uncertainty quantifier and a more robust quantity to be used during training in a deep reinforcement learning setting. Its contribution to robust and sample-efficient training has been observed in other domains (Wang and Manning, 2013; Wu et al., 2019a). Prior work attempted to propagate full covariances through deep neural nets (see, e.g. Wu et al., 2019a; Look et al., 2023; Wright et al., 2024) at a prohibitive computational cost (quadratic in the number of neurons) that does not bring a commensurate empirical benefit. Therefore, we choose to propagate only means and variances.

Assuming the input of a fully-connected layer to be $X \sim \mathcal{N}(X|\mu, \sigma^2)$, the pre-activation Y for a neuron associated with weights θ is given by $Y = \theta^\top X$, i.e., $Y \sim \mathcal{N}(Y|\theta^\top\mu, (\theta^2)^\top\sigma^2)$, where we absorb the bias into θ for simplicity and the square on θ is applied element-wise. For a ReLU activation function $\mathrm{r}(x) \triangleq \max(0,x)$,[1] mean $\widetilde{\mu}$ and variance $\widetilde{\sigma}^2$ of $Y = \max(0,X)$ are analytically tractable (Frey and Hinton, 1999). We approximate the output with a normal distribution $\widetilde{X} \sim \mathcal{N}(\widetilde{X}|\widetilde{\mu}, \widetilde{\sigma}^2)$ and summarize several properties regarding $\widetilde{\mu}$ and $\widetilde{\sigma}^2$ below. See Appendix B.1 for proofs of all results in this subsection.

Lemma 1 (*Moment matching*). *For $X \sim \mathcal{N}(X|\mu, \sigma^2)$ and $Y = \max(0, X)$, we have*

$$\textit{(i)} \quad \widetilde{\mu} \triangleq \mathbb{E}[Y] = \mu\Phi(\alpha) + \sigma\phi(\alpha), \qquad \widetilde{\sigma}^2 \triangleq \mathrm{var}[Y] = (\mu^2+\sigma^2)\Phi(\alpha) + \mu\sigma\phi(\alpha) - \widetilde{\mu}^2,$$

where $\alpha = \mu/\sigma$, and $\phi(\cdot)$, $\Phi(\cdot)$ are the probability density function (pdf) and cumulative distribution function (cdf) of the standard normal distribution, respectively. Additionally, we have that

$$\textit{(ii)} \quad \widetilde{\mu} \geq \mu \quad \textit{and} \quad \textit{(iii)} \quad \widetilde{\sigma}^2 \leq \sigma^2.$$

[1] Although we build our algorithm with the ReLU activation function, it is also applicable to other activation functions whose first and second moments are tractable or can be approximated with sufficient accuracy, including all commonly used activation functions.

https://doi.org/10.52202/079017-2247

Algorithm 1 outlines the moment matching process. This process involves two passes through the linear layer: one for the first moment and one for the second, along with additional operations that take negligible computation time. In contrast, sampling-based methods require N forward passes, where N is the number of samples. Thus, for any $N > 2$, sampling-based methods introduce computational overhead. MOBILE (Sun et al., 2023), e.g., uses $N = 10$. See Figure 1 and Figure 3 for illustrations on the effect of varying N.

Algorithm 1 Deterministic uncertainty propagation through moment matching

function MOMENTMATCHINGTHROUGHLINEAR(θ, b, X)
  θ: Weights of the layer, b: bias of the layer
  $X = \mathcal{N}(X|\mu_X, \sigma_X^2)$ ▷ input a normal distribution
  $(\mu_Y, \sigma_Y^2) \leftarrow (\theta^\top \mu_X + b, \theta^{2\top} \sigma_X^2)$ ▷ transform mean and variance
  return $Y = \mathcal{N}(Y|\mu_Y, \sigma_Y^2)$ ▷ output the transformed distribution
end function
function MOMENTMATCHINGTHROUGHRELU(X)
  $X = \mathcal{N}(X|\mu_X, \sigma_X^2)$ ▷ input a normal distribution
  $\alpha \leftarrow \mu_X / \sigma_X$
  $\mu_Y \leftarrow \mu_X \Phi(\alpha) + \sigma_X \phi(\alpha)$ ▷ compute the first two moments of $\mathrm{ReLU}(X)$
  $\sigma_Y^2 \leftarrow \left(\mu_X^2 + \sigma_X^2\right) \Phi(\alpha) + \mu_X \sigma_X \phi(\alpha) - \mu_Y^2$ ▷ ϕ/Φ are the normal pdf/cdf
  return $Y = \mathcal{N}(Y|\mu_Y, \sigma_Y^2)$ ▷ output a normal distribution with these moments
end function

We propagate the distribution of the next state predicted by the environment model through the action-value function network using moment matching. We define a *moment matching Bellman target distribution* as:

$$\widetilde{\mathbb{B}}_\pi Q(s, a, s') \stackrel{d}{=} r(s, a) + \gamma Q_{MM}(s', \pi(s'))$$

for $s' \sim \widehat{\mathrm{P}}(\cdot|s, a)$ and $Q_{MM}(s', a')$ a normal distribution with mean $\mu_{MM}(s', a')$ and variance $\sigma_{MM}^2(s', a')$ obtained as the outcome of a progressive application of Algorithm 1 through the layers of a critic network Q. The sign $\stackrel{d}{=}$ denotes equality in distribution, i.e., the expressions on the two sides share the same probability law. We perform pessimistic value iteration using a lower confidence bound on $\widetilde{\mathbb{B}}_\pi Q(s, a, s')$ as the Bellman target

$$\widetilde{\mathbb{B}}_\pi^\Gamma Q(s, a, s') \triangleq r(s, a) + \gamma \mu_{MM}(s', \pi(s')) - \beta \gamma \sigma_{MM}(s', \pi(s')) \tag{1}$$

for some radius $\beta > 0$.

4.1 Theoretical analysis of moment matching-based uncertainty propagation

The following lemma provides a bound on the 1-Wasserstein distance W_1 between the true distribution ρ_Y of a random variable after being transformed by a ReLU activation function and its moment matched approximation $\rho_{\widetilde{X}}$. See Appendix B.3 for the proofs of all results presented in this subsection.

Lemma 2 (*Moment matching bound*)**.** *For the following three random variables*

$$X \sim \rho_X, \qquad Y = \max(0, X), \qquad \widetilde{X} \sim \mathcal{N}(\widetilde{X}|\widetilde{\mu}, \widetilde{\sigma}^2)$$

with $\widetilde{\mu} = \mathbb{E}[Y]$ *and* $\widetilde{\sigma}^2 = \mathrm{var}[Y]$ *the following inequality holds*

$$W_1(\rho_Y, \rho_{\widetilde{X}}) \leq \int_{-\infty}^{0} F_{\widetilde{X}}(u) du + W_1(\rho_X, \rho_{\widetilde{X}})$$

where $F_{\widetilde{X}}(\cdot)$ *is cdf of* $\widetilde{X}$ *. If* $\rho_X = \mathcal{N}(X|\mu, \sigma^2)$*, it can be further simplified to*

$$W_1(\rho_Y, \rho_{\widetilde{X}}) \leq \widetilde{\sigma} \phi\left(\frac{\widetilde{\mu}}{\widetilde{\sigma}}\right) - \widetilde{\mu} \Phi\left(-\frac{\widetilde{\mu}}{\widetilde{\sigma}}\right) + |\mu - \widetilde{\mu}| + |\sigma - \widetilde{\sigma}|.$$

Applying this to a moment matching L-layer MLP yields the following deterministic bound.

Lemma 3 (*Moment matching MLP bound*). *Let $f(X)$ be an L-layer MLP with ReLU activation $\mathrm{r}(x) = \max(0, x)$. For $l = 1, \ldots, L-1$, the sampling-based forward-pass is*

$$Y_0 = X_s, \qquad Y_l = \mathrm{r}(f_l(Y_{l-1})), \qquad Y_L = f_L(Y_{L-1})$$

where $f_l(\cdot)$ is the l-th layer and X_s a sample of $\mathcal{N}(X|\mu_X, \sigma_X^2)$. Its moment matching pendant is

$$\widetilde{X}_0 \sim \mathcal{N}(\widetilde{X}_0|\mu_X, \sigma_X^2), \qquad \widetilde{X}_l \sim \mathcal{N}\left(\widetilde{X}_l|\mathbb{E}\left[r(f_l(\widetilde{X}_{l-1}))\right], \mathrm{var}\left[r(f_l(\widetilde{X}_{l-1}))\right]\right).$$

The following inequality holds for $\widetilde{\rho}_Y = \rho_{\widetilde{X}_L} = \mathcal{N}(\widetilde{X}_L|\mathbb{E}[f(\widetilde{X}_{L-1})], \mathrm{var}[f(\widetilde{X}_{L-1})])$.

$$W_1(\rho_Y, \widetilde{\rho}_Y) \leq \sum_{l=2}^{L} \left(G(\widetilde{X}_{l-1}) + C_{l-1}\right) \prod_{j=l}^{L} \|A_j\|,$$

$$\text{with} \quad G(\widetilde{X}_l) = \widetilde{\sigma}_l \phi\left(\frac{\widetilde{\mu}_l}{\widetilde{\sigma}_l}\right) - \widetilde{\mu}_l \Phi\left(-\frac{\widetilde{\mu}_l}{\widetilde{\sigma}_l}\right) \leq 1, \qquad C_l \leq |A_l \widetilde{\mu}_{l-1} - \widetilde{\mu}_l| + \left|\sqrt{A_l^2 \widetilde{\sigma}_{l-1}^2} - \widetilde{\sigma}_l\right|$$

where $\sqrt{\cdot}$ and $(\cdot)^2$ are applied elementwise.

Relying on Lemma 6 again, this result provides an upper bound on the suboptimality of our moment matching approach.

Theorem 3 (*Suboptimality of moment matching-based PEVI algorithms*). *For any policy $\pi_{\texttt{MM}}$ derived with $\mathbb{A}_{\texttt{PEVI}}(\widetilde{\mathbb{B}}_\pi^\Gamma Q, \widehat{\mathrm{P}})$ learned by a penalization algorithm that uses moment matching to approximate the Bellman target with respect to an action-value network defined as an L-layer MLP with 1-Lipschitz activation functions, the following inequality holds*

$$\texttt{SubOpt}(\pi_{\texttt{MM}}) \leq 2H \sum_{l=2}^{L} \left(G(\widetilde{X}_{l-1}) + C_{l-1}\right) \prod_{j=l}^{L} \|A_j\|.$$

The bound in Theorem 3 is much tighter than Theorem 2 in practice as $R_{\max}^2/(1-\gamma)^2$ is very large while the Lipschitz continuities $\|A_j\|$ of the two-layer MLPs used in the experiments are in the order of low hundreds according to empirical investigations (Khromov and Singh, 2024). Another favorable property of Theorem 3 is that its statement is exact, as opposed to the probabilistic statement made in Theorem 2. The provable efficiency of MOMBO could be of independent interest for safety-critical use cases of offline reinforcement learning where the availability of analytical performance guarantees is a fundamental requirement.

4.2 Implementation details of MOMBO

We adopt the model learning scheme from Model-Based Policy Optimization (MBPO) (Janner et al., 2019) and use SAC (Haarnoja et al., 2018) as the policy search algorithm. These approaches represent the best practices in many recent model-based offline reinforcement learning methods (Yu et al., 2020, 2021; Sun et al., 2023). However, we note that most of our findings are more broadly applicable. Following MBPO, we train an independent heteroscedastic neural ensemble of transition kernels modeled as Gaussian distributions over the next state and reward. We denote each ensemble element as $\widehat{\mathrm{P}}_n(s', r|s, a)$ for $n \in \{1, \ldots, N_{\text{ens}}\}$. After evaluating the performance of each model on a validation set, we select the N_{elite} best-performing ensemble elements for further processing. We use these elite models to generate k-step rollouts with the current policy and to create the synthetic dataset $\widehat{\mathcal{D}}$, which we then combine with the real observations $\mathcal{D}$. We retain the mean and variance of the predictive distribution for the next state to propagate it through the action-value function while assigning zero variance to the real tuples.

The lower confidence bound given in Equation (1) builds on our MDP definition, which assumes a deterministic reward function and a deterministic policy for illustrative purposes. In our implementation, the reward model also follows a heteroscedastic ensemble of normal distributions. We also incorporate the uncertainty around the predicted reward of a synthetic sample into the Bellman target calculation. Furthermore, our policy function is a squashed Gaussian distribution trained by SAC in the maximum entropy reinforcement learning setting. For further details, refer to the Appendix C.2.

https://doi.org/10.52202/079017-2247

Table 1: *Performance evaluation on the D4RL dataset.* Normalized reward at 3M gradient steps and Area Under the Learning Curve (AULC) (mean$\pm$std) scores are averaged across four repetitions. The highest means are highlighted in bold and are underlined if they fall within one standard deviation of the best score. The average normalized score is the average across all tasks. The average ranking is based on the rank of the mean.

Dataset Type	Environment	Normalized Reward (↑)			AULC (↑)		
		MOPO	MOBILE	MOMBO	MOPO	MOBILE	MOMBO
random	halfcheetah	37.2±1.6	41.2±1.1	**43.6±1.1**	36.3±1.0	39.5±1.2	**41.4±1.0**
	hopper	**31.7±0.1**	31.3±0.1	25.4±10.2†	**28.6±1.4**	23.6±3.7	17.3±1.3
	walker2d	8.2±5.6	**22.1±0.9**	21.5±0.1	5.4±3.2	18.0±0.4	**19.2±0.5**
	Average on random	25.7	**31.5**	30.2	23.4	**27.1**	26.0
medium	halfcheetah	72.4±4.2	75.8±0.8	**76.1±0.8**	70.9±2.0	72.1±1.0	**73.0±0.9**
	hopper	62.8±38.1	103.6±1.0	**104.2±0.5**	37.0±15.3	82.2±7.3	**95.9±2.5**
	walker2d	85.4±2.9	**88.3±2.5**	86.4±1.2	77.6±1.3	79.0±1.3	**84.0±1.1**
	Average on medium	73.6	**89.3**	88.9	61.8	77.8	**84.3**
medium-replay	halfcheetah	**72.1±3.8**	71.9±3.2	72.0±4.3	68.4±4.7	67.9±2.8	**68.7±3.9**
	hopper	92.7±20.7	**105.1±1.3**	104.8±1.0	81.7±4.6	78.7±4.0	**87.3±2.0**
	walker2d	85.9±5.3	**90.5±1.7**	89.6±3.8	65.3±12.7	79.9±4.3	**80.8±5.6**
	Average on medium-replay	83.4	**89.2**	88.8	72.4	75.5	**78.9**
medium-expert	halfcheetah	83.6±12.5	100.9±1.5	**103.3±0.8**	77.1±4.0	94.5±1.8	**95.2±0.7**
	hopper	74.9±44.2	112.5±0.2	**112.6±0.3**	55.6±17.3	82.7±7.3	**84.3±4.7**
	walker2d	108.2±4.3	**114.5±2.2**	113.9±0.9	88.3±6.3	94.3±0.9	**98.9±3.3**
	Average on medium-expert	88.9	109.3	**109.9**	73.6	90.5	**92.8**
	Average Score	67.6	**79.8**	79.5	57.5	67.7	**70.5**
	Average Ranking	2.7	**1.7**	**1.7**	2.7	2.2	**1.2**

† *High standard deviation due to failure in one repetition, which can be mitigated by increasing β. Median result:* 31.3

5 Experiments

We compare MOMBO against MOPO and MOBILE, the two representative PEVI variants, across twelve tasks from the D4RL dataset (Fu et al., 2020), which consists of data collected from three MuJoCo environments (`halfcheetah`, `hopper`, `walker2d`) with behavior policies exhibiting four degrees of expertise (`random`, `medium`, `medium-replay`, and `medium-expert`). We use the datasets collected with 'v2' versions of the MuJoCo environments to be commensurate with the state of the art. We evaluate model performance using two scores:

(i) Normalized Reward: Total episode reward collected in evaluation mode after offline training ends, normalized by the performances of random and expert policies.

(ii) Area Under Learning Curve (AULC): Average normalized reward computed at the intermediate steps of training.

AULC indicates how fast a model converges to its final performance. A higher AULC reflects more efficient learning other things being equal. We report the main results in Table 1 and provide the learning curves of the `halfcheetah` environment in Figure 2 for visual investigation. We present the learning curves for the remaining tasks in Figure 4. We obtain the results for the baseline models from the log files provided by the OfflineRL library (Sun, 2023). We implement MOMBO into the MOBILE (Sun et al., 2023) code base shared by its authors and use their experiment configurations wherever applicable. See Appendix C for details. The source code of our algorithm is available at `https://github.com/adinlab/MOMBO`. The OfflineRL library does not contain the log files for the random datasets for MOPO at the time of writing. We replicate these experiments using the MOBILE code based on the prescribed configurations.

Theorem 1 indicates that the performance of a PEVI algorithm depends on how tightly its reward penalizer $\Gamma^Q_{\widehat{\mathrm{P}}}(s,a)$ upper bounds the Bellman operator error $|\mathbb{B}_\pi Q(s,a) - \widehat{\mathbb{B}}_\pi Q(s,a)|$. We use this theoretical result to compare the quality of the reward penalizers of the three models based on average values of the following two performance scores calculated across data points observed during ten evaluation episodes:

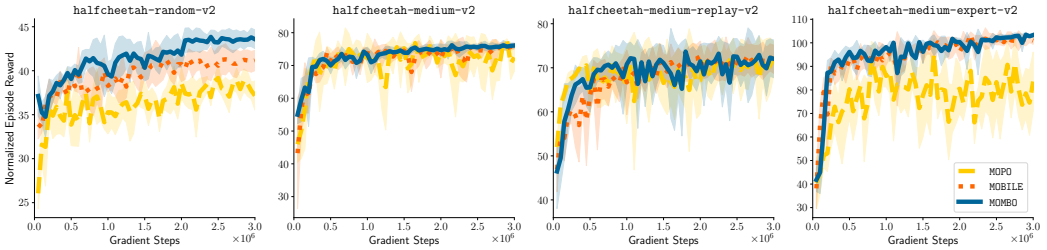

Figure 2: Evaluation results on `halfcheetah` for four settings. The dashed, dotted, and solid curves represent the mean of the normalized rewards across ten evaluation episodes and four random seeds. The shaded area indicates one standard deviation from the mean.

Table 2: *Uncertainty quantification on the D4RL dataset.* Accuracy and tightness (mean$\pm$std) scores are averaged across four repetitions. The highest means are highlighted in bold and are underlined if they fall within one standard deviation of the best score.

Dataset Type	Environment	Accuracy (↑)			Tightness (↑)		
		MOPO	MOBILE	MOMBO	MOPO	MOBILE	MOMBO
random	halfcheetah	$0.02_{\pm 0.0}$	$\mathbf{0.25}_{\pm \mathbf{0.02}}$	$\underline{0.24_{\pm 0.07}}$	$-14.58_{\pm 0.63}$	$-6.88_{\pm 0.61}$	$\mathbf{-6.21}_{\pm \mathbf{0.28}}$
	hopper	$0.14_{\pm 0.04}$	$\mathbf{0.85}_{\pm \mathbf{0.03}}$	$0.61_{\pm 0.07}$	$-1.68_{\pm 0.22}$	$\underline{-0.64_{\pm 0.2}}$	$\mathbf{-0.31}_{\pm \mathbf{0.65}}$
	walker2d	$0.01_{\pm 0.01}$	$0.55_{\pm 0.04}$	$\mathbf{0.74}_{\pm \mathbf{0.06}}$	$-17 \cdot 10^{7}_{\pm 16 \cdot 10^{7}}$	$-0.27_{\pm 0.08}$	$\mathbf{-0.14}_{\pm \mathbf{0.02}}$
medium	halfcheetah	$0.06_{\pm 0.01}$	$0.25_{\pm 0.0}$	$\mathbf{0.34}_{\pm \mathbf{0.04}}$	$-15.71_{\pm 0.68}$	$-10.03_{\pm 0.53}$	$\mathbf{-9.06}_{\pm \mathbf{0.26}}$
	hopper	$0.04_{\pm 0.01}$	$0.41_{\pm 0.01}$	$\mathbf{0.55}_{\pm \mathbf{0.03}}$	$-4.16_{\pm 1.24}$	$-3.14_{\pm 0.08}$	$\mathbf{-1.3}_{\pm \mathbf{0.21}}$
	walker2d	$0.02_{\pm 0.0}$	$0.16_{\pm 0.01}$	$\mathbf{0.38}_{\pm \mathbf{0.02}}$	$-8.91_{\pm 0.61}$	$-5.02_{\pm 0.52}$	$\mathbf{-4.03}_{\pm \mathbf{0.24}}$
medium-replay	halfcheetah	$0.09_{\pm 0.01}$	$0.04_{\pm 0.0}$	$\mathbf{0.16}_{\pm \mathbf{0.0}}$	$-11.67_{\pm 0.94}$	$-11.85_{\pm 0.41}$	$\mathbf{-10.4}_{\pm \mathbf{0.66}}$
	hopper	$0.02_{\pm 0.01}$	$0.03_{\pm 0.01}$	$\mathbf{0.17}_{\pm \mathbf{0.01}}$	$-5.35_{\pm 0.71}$	$-3.4_{\pm 0.04}$	$\mathbf{-3.17}_{\pm \mathbf{0.04}}$
	walker2d	$0.08_{\pm 0.01}$	$0.14_{\pm 0.01}$	$\mathbf{0.36}_{\pm \mathbf{0.02}}$	$-4.47_{\pm 0.43}$	$-4.56_{\pm 0.13}$	$\mathbf{-3.78}_{\pm \mathbf{0.39}}$
medium-expert	halfcheetah	$0.13_{\pm 0.02}$	$0.36_{\pm 0.03}$	$\mathbf{0.44}_{\pm \mathbf{0.03}}$	$-21.25_{\pm 2.87}$	$-13.22_{\pm 0.47}$	$\mathbf{-11.88}_{\pm \mathbf{0.5}}$
	hopper	$0.04_{\pm 0.02}$	$0.43_{\pm 0.02}$	$\mathbf{0.51}_{\pm \mathbf{0.04}}$	$-9.77_{\pm 7.24}$	$-3.5_{\pm 0.03}$	$\mathbf{-3.38}_{\pm \mathbf{0.02}}$
	walker2d	$0.05_{\pm 0.01}$	$\mathbf{0.47}_{\pm \mathbf{0.02}}$	$\underline{0.45_{\pm 0.02}}$	$-9.77_{\pm 0.02}$	$-5.52_{\pm 0.36}$	$\mathbf{-5.29}_{\pm \mathbf{0.14}}$

(i) Accuracy: $\mathbb{1}(\Gamma^{Q}_{\widehat{\mathrm{P}}}(s,a) \geq |\mathbb{B}_\pi Q(s,a) - \widehat{\mathbb{B}}_\pi Q(s,a)|)$ for the indicator function $\mathbb{1}$, i.e., how often the reward penalizer is a valid $\xi-$uncertainty quantifier as assumed by Theorem 1.

(ii) Tightness: $\Gamma^{Q}_{\widehat{\mathrm{P}}}(s,a) - |\mathbb{B}_\pi Q(s,a) - \widehat{\mathbb{B}}_\pi Q(s,a)|$, i.e., how sharp a $\xi-$uncertainty quantifier the reward penalizer is.

Table 2 shows that MOMBO provides more precise uncertainty estimates compared to the sampling-based approaches. It also indicates that MOMBO provides tighter estimates of the Bellman operator error, meaning that the sampling-based approaches use larger confidence radii. See Appendix C.1.2 for further details.

The D4RL dataset is a heavily studied benchmark database where many hyperparameters, such as penalty coefficients, are tuned by trial and error. We argue that this is not feasible in a realistic offline reinforcement learning setting where the central assumption is that policy search needs to be performed without real environment interactions. Furthermore, it is more realistic to assume that collecting data from expert policies is more expensive than random exploration. One would then need to perform offline reinforcement learning on a dataset that comprises observations obtained at different expertise levels. The mixed offline reinforcement learning dataset (Hong et al., 2023) satisfies both desiderata, as the baseline models are not engineered for its setting and its tasks comprise data from two policies: an expert or medium policy and a random policy, presented in varying proportions. We compare MOMBO against MOBILE and MOPO on this dataset for a fixed and shared penalty coefficient for all models. We find that MOMBO outperforms both baselines in final performance and learning speed in this more realistic setup. See Appendix C.1.3 and Table 3 for further details and results.

Our key experimental findings can be summarized as follows:

(i) MOMBO converges faster and trains more stably. It learns more rapidly and reaches its final performance earlier as visible from the AULC scores. Its moment matching approach

https://doi.org/10.52202/079017-2247

provides more informative gradient signals and better estimates of the first two moments of the Bellman target compared to sampling-based approaches, which suffer from high estimator variance caused by Monte Carlo sampling.

(ii) *MOMBO delivers a competitive final training performance.* It ranks highest across all tasks and outperforms other model-based offline reinforcement learning approaches, including COMBO (Yu et al., 2021) with an average normalized reward of 66.8, TT (Janner et al., 2021) with 62.3, and RAMBO (Rigter et al., 2022) with 67.7, all evaluated on the same twelve tasks in the D4RL dataset. We took these scores from Sun et al. (2023).

(iii) *MOMBO delivers superior final training performance when expert data is limited.* MOMBO outperforms the baselines (at 1% in the mixed dataset) in both AULC and normalized reward. For normalized reward, MOMBO achieves 37.0, compared to 32.4 for MOBILE and 27.9 for MOPO, averaged across all tasks at 1%. In terms of AULC, MOMBO attains 29.5, outperforming the 28.0 of MOBILE and the 23.5 of MOPO. See Table 3 for details.

(iv) *MOMBO provides more precise estimates of Bellman target uncertainty.* It outperforms both baselines in accuracy for 9 out of 12 tasks and in tightness all 12 tasks in the D4RL dataset.

6 Conclusion

The main objective of MOMBO is to accurately quantify the uncertainty surrounding a Bellman target estimate caused by the error in predicting the next state. We achieve this by deterministically propagating uncertainties through the value function with moment matching and introducing pessimism by constructing a lower confidence bound on the target Q-values. We analyze our model theoretically and evaluate its performance in various environments. Our findings may lay the groundwork for further theoretical analysis of model-based reinforcement learning algorithms in continuous state-action spaces. Our algorithmic contributions can also be instrumental in both model-based online and model-free offline reinforcement learning setups (An et al., 2021; Bai et al., 2022).

Limitations. The accuracy of the learned environment models sets a bottleneck on the performance of MOMBO. The choice of the confidence radius β is another decisive factor in model performance. MOMBO shares these weaknesses with the other state-of-the-art model-based offline reinforcement learning methods (Yu et al., 2020; Lu et al., 2021; Sun et al., 2023) and does not contribute to their mitigation. Furthermore, our theoretical analyses rely on the assumption of normally distributed Q-values, which may be improved with heavy-tailed assumed densities. Considering the bounds, a limitation is that we claim a tighter bound compared to a baseline bound that we derived ourselves. However, the most important factor in that bound, $R_{\max}^2/(1-\gamma)^2$, arises from Hoeffding's inequality, i.e., a classical statement. As such, we assume our bound to be rigorous and not inadvertently biased. Finally, our moment matching method is limited to activation functions for which the first two moments are either analytically tractable or can be approximated with sufficient accuracy, which includes all commonly used activation functions. Extensions to transformations such as, e.g., BatchNorm (Ioffe and Szegedy, 2015), or other activation functions remove the analytical tractability for moment matching. However, these are usually not used in our specific applications.

Acknowledgments and Disclosure of Funding

AA and MH thank the Carlsberg Foundation for supporting their research under the grant number CF21-0250. We are grateful to Birgit Debrabant for her valuable inputs in the development of some theoretical aspects. We also thank all reviewers and the area chair for improving the quality of our work by their thorough reviews and active discussion during the rebuttal phase.

References

Abbas, Z., Sokota, S., Talvitie, E., and White, M. (2020). Selective Dyna-style planning under limited model capacity. In *International Conference on Machine Learning*.

An, G., Moon, S., Kim, J.-H., and Song, H. O. (2021). Uncertainty-based offline reinforcement learning with diversified Q-ensemble. In *Advances in Neural Information Processing Systems*.

Bai, C., Wang, L., Yang, Z., Deng, Z., Garg, A., Liu, P., and Wang, Z. (2022). Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*.

Casella, G. and Berger, R. (2024). *Statistical inference*. CRC Press.

Cetin, E., Tirinzoni, A., Pirotta, M., Lazaric, A., Ollivier, Y., and Touati, A. (2024). Simple ingredients for offline reinforcement learning. In *International Conference on Machine Learning*.

Cheng, C.-A., Xie, T., Jiang, N., and Agarwal, A. (2022). Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*.

Chhachhi, S. and Teng, F. (2023). On the 1-wasserstein distance between location-scale distributions and the effect of differential privacy. *arXiv preprint arXiv:2304.14869*.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. (2018). Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*.

Frey, B. J. and Hinton, G. E. (1999). Variational learning in nonlinear Gaussian belief networks. *Neural Computation*.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2020). D4RL: Datasets for deep data-driven reinforcement learning.

Fujimoto, S. and Gu, S. S. (2021). A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*.

Fujimoto, S., Meger, D., and Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*.

Gast, J. and Roth, S. (2018). Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Haußmann, M., Hamprecht, F. A., and Kandemir, M. (2019). Deep active learning with adaptive acquisition. In *International Joint Conference on Artificial Intelligence*.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*.

Hong, Z.-W., Agrawal, P., Combes, R. T. d., and Laroche, R. (2023). Harnessing mixed offline reinforcement learning datasets via trajectory weighting. In *International Conference on Learning Representations*.

Hui, D. Y.-T., Courville, A. C., and Bacon, P.-L. (2023). Double Gumbel Q-learning. In *Advances in Neural Information Processing Systems*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*.

Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*.

Janner, M., Li, Q., and Levine, S. (2021). Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*.

Jeong, J., Wang, X., Gimelfarb, M., Kim, H., Abdulhai, B., and Sanner, S. (2023). Conservative Bayesian model-based value expansion for offline policy optimization. In *International Conference on Learning Representations*.

 https://doi.org/10.52202/079017-2247

Jin, Y., Yang, Z., and Wang, Z. (2021). Is pessimism provably efficient for offline RL? In *International Conference on Machine Learning*.

Khromov, G. and Singh, S. P. (2024). Some fundamental aspects about Lipschitz continuity of neural networks. In *International Conference on Learning Representations*.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. (2020). MOReL : Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. (2021). Offline reinforcement learning with Fisher divergence critic regularization. In *International Conference on Machine Learning*.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. (2019). Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*.

Lange, S., Gabel, T., and Riedmiller, M. (2012). Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*. Springer.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Look, A., Kandemir, M., Rakitsch, B., and Peters, J. (2023). A deterministic approximation to neural SDEs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Lu, C., Ball, P. J., Parker-Holder, J., Osborne, M. A., and Roberts, S. J. (2021). Revisiting design choices in offline model-based reinforcement learning. In *International Conference on Learning Representations*.

Luis, C. E., Bottero, A. G., Vinogradska, J., Berkenkamp, F., and Peters, J. (2023). Model-based uncertainty in value functions. In *International Conference on Artificial Intelligence and Statistics*.

Micheli, V., Alonso, E., and Fleuret, F. (2022). Transformers are sample-efficient world models. In *International Conference on Learning Representations*.

Nie, X., Brunskill, E., and Wager, S. (2021). Learning when-to-treat policies. *Journal of the American Statistical Association*.

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. (2018). The uncertainty Bellman equation and exploration. In *International Conference on Machine Learning*.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. (2019). Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*.

Pitcan, Y. (2017). A note on concentration inequalities for U-statistics. *arXiv preprint arXiv:1712.06160*.

Rigter, M., Lacerda, B., and Hawes, N. (2022). RAMBO-RL: Robust adversarial model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*.

Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., and Murphy, S. A. (2011). Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*.

Siegel, N. Y., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., Heess, N., and Riedmiller, M. (2020). Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*.

Singh, A., Yu, A., Yang, J., Zhang, J., Kumar, A., and Levine, S. (2020). COG: Connecting new skills to past experience with offline reinforcement learning. In *Conference on Robot Learning*.

Sun, Y. (2023). Offlinerl-kit: An elegant pytorch offline reinforcement learning library. https://github.com/yihaosun1124/OfflineRL-Kit.

Sun, Y., Zhang, J., Jia, C., Lin, H., Ye, J., and Yu, Y. (2023). Model-Bellman inconsistency for model-based offline reinforcement learning. In *International Conference on Machine Learning*.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings 1990*. Morgan Kaufmann.

Thrun, S. and Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School*.

Todorov, E., Erez, T., and Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Uehara, M. and Sun, W. (2022). Pessimistic model-based offline reinforcement learning under partial coverage. In *International Conference on Learning Representations*.

Villani, C. et al. (2009). *Optimal transport: old and new*. Springer.

Wang, S. and Manning, C. (2013). Fast dropout training. In *International Conference on Machine Learning*.

Wasserman, L. (2004). *All of Statistics*. Springer New York.

Wright, O., Nakahira, Y., and Moura, J. M. (2024). An analytic solution to covariance propagation in neural networks. In *International Conference on Artificial Intelligence and Statistics*.

Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernandez-Lobato, J. M., and Gaunt, A. L. (2019a). Deterministic variational inference for robust Bayesian neural networks. In *International Conference on Learning Representations*.

Wu, Y., Tucker, G., and Nachum, O. (2019b). Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*.

Xie, T., Cheng, C.-A., Jiang, N., Mineiro, P., and Agarwal, A. (2021). Bellman-consistent pessimism for offline reinforcement learning. In *Advances in Neural Information Processing Systems*.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. (2021). COMBO: Conservative offline model-based policy optimization. In *Advances in Neural Information Processing Systems*.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. (2020). MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*.

 https://doi.org/10.52202/079017-2247

APPENDIX

A Related work

Offline reinforcement learning. The goal of offline reinforcement learning is to derive an optimal policy from fixed datasets collected through interactions with a behavior policy in the environment. This approach is particularly relevant in scenarios where direct interaction with the environment is costly or poses potential risks. The applications of offline reinforcement learning span various domains, including robotics (Singh et al., 2020; Micheli et al., 2022) and healthcare (Shortreed et al., 2011; Nie et al., 2021). Notably, applying off-policy reinforcement learning algorithms directly to offline reinforcement learning settings often fails due to issues such as overestimation bias and distributional shift. Research in offline reinforcement learning has evolved along two main avenues: model-free and model-based approaches.

Model-based offline reinforcement learning. Dyna-style model-based reinforcement learning (Sutton, 1990; Janner et al., 2019) algorithms employ a environment model to simulate the true transition model, generating a synthetic dataset that enhances sample efficiency and serves as a surrogate environment for interaction. Model-based offline reinforcement learning methods vary in their approaches to applying conservatism using the environment model. For instance, MOPO (Yu et al., 2020) and MORel (Kidambi et al., 2020) learn a pessimistic value function by penalizing the MDP generated by the environment model, with rewards being penalized based on different uncertainty measures of the environment model. COMBO (Yu et al., 2021) is a Dyna-style model-based approach derived from conservative Q-learning (Kumar et al., 2020). RAMBO (Rigter et al., 2022) introduces conservatism by adversarially training the environment model to minimize the value function, thus ensuring accurate predictions. MOBILE (Sun et al., 2023) penalizes the value function based on the uncertainty of the Bellman operator through sampling. CBOP (Jeong et al., 2023) introduces conservatism by applying a lower confidence bound to the value function with an adaptive weighting of h-step returns in the model-based value expansion framework (Feinberg et al., 2018).

Uncertainty-driven offline reinforcement learning approaches apply penalties in various ways. As model-free methods, EDAC (An et al., 2021) applies penalties based on ensemble similarities, while PBRL (Bai et al., 2022) penalizes based on the disagreement among bootstrapped Q-functions. Uncertainty-driven model-based offline reinforcement learning algorithms adhere to the meta-algorithm known as pessimistic value iteration (Jin et al., 2021). This meta-algorithm introduces pessimism through a ξ-uncertainty quantifier to minimize the Bellman approximation error. MOPO (Yu et al., 2020) and MOBILE (Sun et al., 2023) can be seen as practical implementations of this meta-algorithm. MOPO leverages prediction uncertainty from the environment model in various ways, as explored by Lu et al. (2021), while MOBILE uses uncertainty in the value function for the synthetic dataset through sampling. MOMBO is also based on PEVI, and our ξ-uncertainty quantifier is derived from the uncertainty of the Bellman target value, which does not rely on sampling. Instead, it directly propagates uncertainty from the environment models to the value function, thanks to moment matching.

Model-free offline reinforcement learning algorithms can be broadly categorized into two groups: policy regularization and value regularization. Policy regularization methods aim to constrain the learned policy to prevent deviations from the behavior policy (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019b; Peng et al., 2019; Siegel et al., 2020). For instance, during training, TD3-BC (Fujimoto and Gu, 2021) incorporates a behavior cloning term to regulate its policy. On the other hand, value regularization methods (Kostrikov et al., 2021; Xie et al., 2021; Cheng et al., 2022) introduce conservatism during the value optimization phase using various approaches. For example, CQL (Kumar et al., 2020) penalizes Q-values associated with out-of-distribution actions to avoid overestimation. Similarly, EDAC (An et al., 2021), which employs ensemble value networks, applies a similar penalization strategy based on uncertainty measures over Q-values.

Uncertainty propagation via various moment matching-based approaches has been well-researched, primarily in the area of Bayesian deep learning (Frey and Hinton, 1999; Wang and

Manning, 2013; Wu et al., 2019a), with applications, e.g., in active learning (Haußmann et al., 2019) and computer vision (Gast and Roth, 2018).

B Theoretical analysis

This section contains proofs for all the theoretical statements made in the main text.

B.1 Prerequisites

We first restate and extend the following result on moment matching with the ReLU activation function. Our focus on the ReLU activation throughout this work is due to the fact that it is currently the most commonly used activation function in deep reinforcement learning. One could derive similar results for any piecewise linear activation function and several others as well.

Lemma 1 *(Moment matching). For $X \sim \mathcal{N}(X|\mu, \sigma^2)$ and $Y = \max(0, X)$, we have*

$$\textit{(i)} \quad \widetilde{\mu} \triangleq \mathbb{E}[Y] = \mu\Phi(\alpha) + \sigma\phi(\alpha), \qquad \widetilde{\sigma}^2 \triangleq \operatorname{var}[Y] = (\mu^2 + \sigma^2)\Phi(\alpha) + \mu\sigma\phi(\alpha) - \widetilde{\mu}^2$$

where $\alpha = \mu/\sigma$, and $\phi(\cdot)$, $\Phi(\cdot)$ are the probability density function (pdf) and cumulative distribution function (cdf) of the standard normal distribution, respectively. Additionally, we have that

$$\textit{(ii)} \quad \widetilde{\mu} \geq \mu \quad \textit{and} \quad \textit{(iii)} \quad \widetilde{\sigma}^2 \leq \sigma^2.$$

Proof of Lemma 1. See Frey and Hinton (1999) for the derivation of *(i)*, i.e., $\widetilde{\mu}$ and $\widetilde{\sigma}^2$.

Statement *(ii)* holds as

$$\widetilde{\mu} = \mathbb{E}[Y] = \int_0^\infty xp(x)dx \geq \int_{-\infty}^0 xp(x)dx + \int_0^\infty xp(x)dx = \mu.$$

Note that

$$\mathbb{E}\left[Y^2\right] = \int_0^\infty x^2p(x)dx \leq \int_{-\infty}^0 x^2p(x)dx + \int_0^\infty x^2p(x)dx = \mathbb{E}\left[X^2\right].$$

Therefore, we get *(iii)* by

$$\widetilde{\sigma}^2 = \operatorname{var}[Y] = \mathbb{E}\left[Y^2\right] - \widetilde{\mu}^2 \leq \mathbb{E}\left[X^2\right] - \mu^2 = \sigma^2$$

concluding the proof. □

Although we do not require the following lemma in our final results, a helpful result is the following inequality between the cdfs of two normally distributed random variables X and its matched counterpart $\widetilde{X}$.

Lemma 4 (*An inequality between normal cdfs*)**.** *For two normally distributed random variables $X \sim \mathcal{N}(X|\mu, \sigma^2)$ and $\widetilde{X} \sim \mathcal{N}(\widetilde{X}|\widetilde{\mu}, \widetilde{\sigma}^2)$ with $\widetilde{\mu}\sigma \geq \mu\widetilde{\sigma}$, the following inequality holds:*

$$F_{\widetilde{X}}(u) \leq F_X(u) \qquad \textit{for } u \leq 0.$$

Given $\widetilde{\mu}$ and $\widetilde{\sigma}^2$ as in Lemma 1, the assumptions of this lemma hold and it is applicable to our moment matching propagation.

Proof of Lemma 4. Assume $u \leq 0$. We have that

$$F_{\widetilde{X}}(u) \leq F_X(u) \quad \Leftrightarrow \quad \Phi\Big(\frac{u - \widetilde{\mu}}{\widetilde{\sigma}}\Big) \leq \Phi\Big(\frac{u - \mu}{\sigma}\Big).$$

As it is a monotonically increasing function, this in turn is equivalent to

$$\begin{aligned} \frac{u - \widetilde{\mu}}{\widetilde{\sigma}} \leq \frac{u - \mu}{\sigma} \quad &\Leftrightarrow u\sigma - \widetilde{\mu}\sigma \leq u\widetilde{\sigma} - \mu\widetilde{\sigma} \\ &\Leftrightarrow u(\sigma - \widetilde{\sigma}) \leq \widetilde{\mu}\sigma - \mu\widetilde{\sigma} \end{aligned}$$

which holds as $u(\sigma - \tilde{\sigma}) < 0$ via Lemma 1 for $u < 0$, and $\widetilde{\mu}\sigma \geq \mu\widetilde{\sigma}$ by assumption. □

https://doi.org/10.52202/079017-2247

We provide the following definition, as we derive our bounds using Wasserstein distances.

Definition 1 (*Wasserstein distance*)**.** Let (M, d) denote a metric space, and let $p \in [1, \infty]$. The p-*Wasserstein distance* between two probability measures ρ_1 and ρ_2 on M, with finite p-moments, is defined as

$$W_p(\rho_1, \rho_2) \triangleq \inf_{\gamma \in \chi(\rho_1, \rho_2)} \left(\mathbb{E}_{(x,y)\sim\gamma}\left[d(x,y)^p\right]\right)^{1/p}$$

where $\chi(\cdot, \cdot)$ represents the set of all couplings of two measures, ρ_1 and ρ_2.

For $p = 1$, the 1-Wasserstein distance, W_1, can be simplified (see, e.g., Villani et al., 2009) to

$$W_1(\rho_1, \rho_2) = \int_{\mathbb{R}} \left|F_1(x) - F_2(x)\right| dx \tag{2}$$

where ρ_1, ρ_2 are two probability measures on $\mathbb{R}$, and $F_1(\cdot)$ and $F_2(\cdot)$ are their respective cdfs.

Given this definition, we prove the following two bounds on W_1.

Lemma 5 (*Wasserstein inequalities*)**.** *The following inequalities hold for the* 1*-Wasserstein metric*

(i) $W_1(\rho_U^1, \rho_U^2) \leq \|A\| W_1(\rho_V^1, \rho_V^2)$*, for* $U = AV + b$*.*

(ii) $W_1(\rho_Z^1, \rho_Z^2) \leq W_1(\rho_U^1, \rho_U^2)$*, for* $Z = g(U)$ *with* $g(\cdot)$ *locally* K*-Lipschitz with* $K \leq 1$

where ρ_x *is the density of* x*,* $U \in \mathbb{R}^{d_u}$*,* $V \in \mathbb{R}^{d_v}$*,* $A \in \mathbb{R}^{d_u \times d_v}$*, and* $b \in \mathbb{R}^{d_u}$*.*

Proof of Lemma 5. Given the Kantorovich-Rubinstein duality (Villani et al., 2009) for W_1 we have

$$W_1(\mu, \nu) = \frac{1}{K} \sup_{\text{lip}(f) \leq K} \mathbb{E}_{x\sim\mu}\left[f(x)\right] - \mathbb{E}_{y\sim\nu}\left[f(y)\right]$$

for any two probability measures μ, ν, and where $\text{lip}(f) < K$ specifies the family of K-Lipschitz functions.

We use this to first prove *(ii)*. For $Z = g(U)$ we have that

$$\begin{aligned}
W_1(\rho_Z^1, \rho_Z^2) &= \sup_{\text{lip}(f)\leq 1} \mathbb{E}_{z\sim\rho_Z^1}\left[f(z)\right] - \mathbb{E}_{z\sim\rho_Z^2}\left[f(z)\right] \\
&= \sup_{\text{lip}(f)\leq 1} \mathbb{E}_{u\sim\rho_U^1}\left[f(g(u))\right] - \mathbb{E}_{u\sim\rho_U^2}\left[f(g(u))\right], && \text{change of variables} \\
&\leq \sup_{\text{lip}(h)\leq 1} \mathbb{E}_{u\sim\rho_U^1}\left[h(u)\right] - \mathbb{E}_{u\sim\rho_U^2}\left[h(u)\right] = W_1(\rho_U^1, \rho_U^2)
\end{aligned}$$

where the inequality holds as $f \circ g$ has a Lipschitz constant $L \leq 1$, i.e., we end up with a supremum over a smaller class of Lipschitz functions which we revert in the inequality. This gives us the desired inequality.

To prove *(i)* we use that for $\|A\| \leq 1$, the transformation $g(V) \triangleq 1$ is 1-Lipschitz and the result follows via *(ii)*. For $\|A\| > 1$ we use that $g(\cdot)$ is K-Lipschitz and the result follows by adapting the proof accordingly. □

Finally, the following lemma allows us to relate the 1-Wasserstein distance of two transformed distributions to an absolute difference between the respective transforming functions.

Lemma 6 (*Wasserstein to suboptimality*)**.** *Consider two probability distributions* P_x, P_u *defined on a measurable space* $(\mathcal{S}, \sigma(\mathcal{S}))$ *and two transformed random variables* $y = f(x) \sim P_y$ *(with* $x \sim P_x$*), and* $z = g(u) \sim P_z$ *(with* $u \sim P_u$*) for functions* $f, g : \mathcal{S} \to \mathbb{R}$*. If*

$$W_1(P_y, P_z) \leq C \qquad \textit{then} \quad \left|\mathbb{E}_{x\sim P_x}\left[f(x)\right] - \mathbb{E}_{u\sim P_u}\left[g(u)\right]\right| \leq C$$

for any constant $C \geq 0$*.*

Proof of Lemma 6. Assume that $W_1(P_y, P_z) \leq C$ for some $C \geq 0$. The Kantorovich-Rubinstein duality (Villani et al., 2009), with $K = 1$ gives us

$$W_1(P_y, P_z) = \sup_{\text{lip}(h)\leq 1} \mathbb{E}_{y\sim P_y}\left[h(y)\right] - \mathbb{E}_{z\sim P_z}\left[h(z)\right] \leq C.$$

As the identity function $\lambda : x \mapsto x$ is 1-Lipschitz, we can lower bound the supremum further with

$$\mathbb{E}_{y\sim P_y}[y] - \mathbb{E}_{z\sim P_z}[z] \leq \sup_{\text{lip}(h)\leq 1} \mathbb{E}_{y\sim P_y}[h(y)] - \mathbb{E}_{z\sim P_z}[h(z)] \leq C.$$

Using what is colloquially known as the law of the unconscious statistician (Casella and Berger, 2024) , we have that

$$\mathbb{E}_{y\sim P_y}[y] = \mathbb{E}_{x\sim P_x}[f(x)]$$

and analogously for $\mathbb{E}_{z\sim P_z}[z]$. Therefore

$$\mathbb{E}_{y\sim P_y}[y] - \mathbb{E}_{z\sim P_z}[z] = \mathbb{E}_{x\sim P_x}[f(x)] - \mathbb{E}_{u\sim P_u}[g(u)]$$

which gives us

$$\mathbb{E}_{x\sim P_x}[f(x)] - \mathbb{E}_{u\sim P_u}[g(u)] \leq C.$$

As W_1 is symmetric, we can follow the same argument and additionally have

$$\mathbb{E}_{u\sim P_u}[g(u)] - \mathbb{E}_{x\sim P_x}[f(x)] \leq C.$$

Using that for $x \in \mathbb{R}$ with $x < C$ and $-x < C$ it follows that $|x| < C$, we can combine these two inequalities and get

$$\left|\mathbb{E}_{x\sim P_x}[f(x)] - \mathbb{E}_{u\sim P_u}[g(u)]\right| \leq C$$

as desired. □

B.2 Proof of theorem 2

Proving Theorem 2 requires the following upper W_1 bound for an MLP with a 1-Lipschitz activation function.

Lemma 7 (W_1 *bound on an MLP*)**.** *Consider an L-layer MLP* $f(\cdot)$

$$Y_l = A_l^\top X_{l-1} + b_l, \qquad X_l = \mathrm{r}(Y_l), \qquad l = 1, \ldots, L$$

where A_l, b_l *are the weight matrix and bias vector for layer* l*, respectively, and* $\mathrm{r}(\cdot)$ *denotes an elementwise* 1*-Lipschitz activation function. We write* $Y \triangleq Y_L = f(X_0)$*, where* X_0 *is the input. Assuming two measures* $X_1 \sim \rho_X^1$ *and* $X_2 \sim \rho_X^2$*, we have that*

$$W_1(\rho_Y^1, \rho_Y^2) \leq \prod_{l=1}^{L} \|A_l\| W_1(\rho_X^1, \rho_X^2).$$

Proof of Lemma 7. We have that

$$\begin{aligned} W_1(\rho_Y^1, \rho_Y^2) &\leq \|A_L\| W_1(\rho_{X_{L-1}}^1, \rho_{X_{L-1}}^2), && \text{via Lemma 5 (i)} \\ &= \|A_L\| W_1\big(\rho_{h(Y_{L-1})}^1, \rho_{h(Y_{L-1})}^2\big) \leq \|A_L\| W_1\big(\rho_{Y_{L-1}}^1, \rho_{Y_{L-1}}^2\big), && \text{via Lemma 5 (ii)} \\ &\leq \cdots \leq \prod_{l=1}^{L} \|A_l\| W_1(\rho_X^1, \rho_X^2). \end{aligned}$$

□

The W_1 distance between two measures on the output distribution is upper bounded by the W_1 distance between the corresponding input measures with a factor given by the product of the norms of the weight matrices A_l. This result allows us to provide a probabilistic bound on the W_1 distance between a normal ρ_Y and its empirical, sampling-based, estimate $\hat{\rho}_Y$.

The following lemma allows us to extend this result to a probabilistic sampling-based upper bound.

Lemma 8 (*Sampling-based MLP bound*)**.** *For an L-layer MLP* $f(\cdot)$*with* 1*-Lipschitz activation function, let* $Y \triangleq Y_L = f(X)$ *where* $X \sim \mathcal{N}(X|\mu_X, \sigma_X^2)$*, the following bound holds:*

$$\mathbb{P}\left(W_1(\rho_Y, \rho_{\hat{Y}}^N) \leq \prod_{l=1}^{L} \|A_l\| \sqrt{-\frac{8\log(\delta/4) R_{\max}^2}{\lfloor N/2 \rfloor (1-\gamma)^2}}\right) \geq 1 - \delta$$

where $\delta \in (0, 1)$ *and* $\rho_{\hat{Y}}^N$ *is the empirical estimate given* N *i.i.d. samples.*

https://doi.org/10.52202/079017-2247

Proof of Lemma 8. With Hoeffding's inequality (Wasserman, 2004), we have that

$$\mathbb{P}\left(|\hat{\mu}_N - \mu| \geq \varepsilon\right) \leq 2\exp\left(-2N^2\varepsilon^2 \Big/ \sum_{i=1}^{N}\left(\frac{R_{\max}}{1-\gamma} - 0\right)^2\right) = 2\exp\left(-\frac{2\varepsilon^2 N(1-\gamma)^2}{R_{\max}^2}\right)$$

for the empirical mean $\hat{\mu}_N$ of Y. As variances are U-statistics with order $m = 2$ and kernel $h = \frac{1}{2}(x-y)^2$ using Hoeffding (1963); Pitcan (2017), we have for the empirical covariance of Y, $\hat{\sigma}_N^2$, that

$$\begin{aligned} & \mathbb{P}(\hat{\sigma}_N^2 - \sigma^2 \geq \varepsilon) \leq \exp\left(-\frac{\varepsilon^2 \lfloor N/m \rfloor}{2\|h\|_\infty^2}\right) = \exp\left(-\frac{\varepsilon^2 \lfloor N/2 \rfloor (1-\gamma)^2}{2R_{\max}^2}\right) \\ \Leftrightarrow \quad & \mathbb{P}(\hat{\sigma}_N^2 - \sigma^2 \leq \varepsilon) \geq 1 - \exp\left(-\frac{\varepsilon^2 \lfloor N/2 \rfloor (1-\gamma)^2}{2R_{\max}^2}\right) \end{aligned}$$

where $\|\cdot\|_\infty$ denotes L^∞-norm. Applying the inequality $\hat{\sigma}_N \leq \sqrt{\sigma^2 + \varepsilon} \leq \sigma + \sqrt{\varepsilon}$, we get

$$\begin{aligned} & \mathbb{P}(\hat{\sigma}_N \leq \sigma + \varepsilon) \geq 1 - \exp\left(-\frac{\varepsilon^2 \lfloor N/2 \rfloor (1-\gamma)^2}{2R_{\max}^2}\right) \\ \Leftrightarrow \quad & \mathbb{P}(\hat{\sigma}_N - \sigma \geq \varepsilon) \leq \exp\left(-\frac{\varepsilon^2 \lfloor N/2 \rfloor (1-\gamma)^2}{2R_{\max}^2}\right). \end{aligned}$$

As the same inequality holds for $\sigma - \hat{\sigma}_N \geq \varepsilon$, a union bound give us that

$$\mathbb{P}(|\sigma - \hat{\sigma}_n| \geq \varepsilon) \leq 2\exp\left(-\frac{\varepsilon^2 \lfloor N/2 \rfloor (1-\gamma)^2}{2R_{\max}^2}\right).$$

Using an analytical upper bound (Chhachhi and Teng, 2023) on the 1-Wasserstein distance between two normal distributions,

$$W_1(\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2)) \leq |\mu_1 - \mu_2| + |\sigma_1 - \sigma_2|$$

we have that

$$\begin{aligned} & \mathbb{P}\big(W_1(\mathcal{N}(\mu,\sigma^2), \mathcal{N}(\hat{\mu}_N, \hat{\sigma}_N)) \geq 2\varepsilon\big) \\ & \quad \leq \mathbb{P}\left(|\mu - \hat{\mu}_N| + |\sigma - \hat{\sigma}_N| \geq 2\varepsilon\right) \\ & \quad \leq \mathbb{P}(|\hat{\mu}_N - \mu| \geq \varepsilon \text{ or } |\hat{\sigma}_N - \sigma| \geq \varepsilon) \\ & \quad \leq \mathbb{P}(|\hat{\mu}_N - \mu| \geq \varepsilon) + \mathbb{P}(|\hat{\sigma}_N - \sigma| \geq \varepsilon), \qquad \text{union bound} \\ & \quad \leq 2\exp(-2Nc\varepsilon^2) + 2\exp(-0.5\lfloor N/2 \rfloor c\varepsilon^2) \\ & \quad \leq 4\exp(-0.5\lfloor N/2 \rfloor c\varepsilon^2) \triangleq \delta, \end{aligned}$$

where we use the shorthand notation $c = (1-\gamma)^2/R_{\max}^2$.

Solving for $\bar{\varepsilon} = 2\varepsilon$ gives us

$$\bar{\varepsilon} = \sqrt{-\frac{8\log(\delta/4)}{\lfloor N/2 \rfloor c}}.$$

The bound is then given as

$$\mathbb{P}\left(W_1\left(\mathcal{N}(\mu,\sigma^2), \mathcal{N}(\hat{\mu}_N, \hat{\sigma}_N^2)\right) \leq \sqrt{-\frac{8\log(\delta/4)}{\lfloor N/2 \rfloor c}}\right) \geq 1 - \delta$$

which gives us with Lemma 7, that

$$\mathbb{P}\left(W_1(\rho_Y, \rho_{\hat{Y}}^N) \leq \prod_{l=1}^{L}\|A_l\| \sqrt{-\frac{8\log(\delta/4)R_{\max}^2}{\lfloor N/2 \rfloor (1-\gamma)^2}}\right) \geq 1 - \delta.$$

□

Finally, applying this result to the relation in Lemma 6 allows us to prove the desired non-asymptotic bound on the performance of the sampling-based model.

Theorem 2 *(Suboptimality of sampling-based PEVI algorithms). For any policy π_{MC} learned by $\mathbb{A}_{PEVI}(\widehat{\mathbb{B}}_\pi^\Gamma Q, \widehat{\mathrm{P}})$ using N Monte Carlo samples to approximate the Bellman target with respect to an action-value network defined as an L-layer MLP with 1-Lipschitz activation functions, the following inequality holds for any error tolerance $\delta \in (0,1)$ with probability at least $1-\delta$*

$$\texttt{SubOpt}(\pi_{MC}) \leq 2H \prod_{l=1}^{L} \|A_l\| \sqrt{-\frac{8\log(\delta/4) R_{\max}^2}{\lfloor N/2 \rfloor (1-\gamma)^2}}.$$

Proof of Theorem 2. Define $C = \prod_{l=1}^{L} \|A_l\| \sqrt{-\frac{8\log(\delta/4) R_{\max}^2}{\lfloor N/2 \rfloor (1-\gamma)^2}}$. Then, combining the bound from Lemma 8 with the result from Lemma 6, we get that with probability $1-\delta$

$$\begin{aligned}
\left| \mathbb{E}_{X\sim\mathcal{N}(\mu_X,\sigma_X^2)} \left[\widehat{\mathbb{B}}_\pi Q(X)\right] - \mathbb{E}_{X\sim\mathcal{N}(\widehat{\mu}_X,\widehat{\sigma}_X^2)} \left[\widehat{\mathbb{B}}_\pi Q(X)\right] \right| &\leq C \\
\left| \mathbb{B}_\pi Q(s,a) - \mathbb{E}_{X\sim\mathcal{N}(\widehat{\mu}_X,\widehat{\sigma}_X^2)} \left[\widehat{\mathbb{B}}_\pi Q(X)\right] \right| &\leq C \\
\left| \mathbb{B}_\pi Q(s,a) - \widehat{\mathbb{B}}_\pi Q(s,a,s') \right| &\leq C.
\end{aligned}$$

Note that $X = (s,a,s')$, i.e., the state, action, and next state tuple, is distributed as described in the main text. Therefore C is a ξ-uncertainty quantifier, with $\xi = \delta$. Applying Theorem 1 finalizes the proof. □

B.3 Proof of theorem 3

In order to prove Theorem 3, i.e., an upper bound on our proposed moment matching approach, we require the following two lemmata.

The first lemma allows us to upper bound the 1-Wasserstein distance between an input distribution and its output distribution after transformation by the ReLU activation function.

Lemma 2 *(Moment matching bound). For the following three random variables*

$$X \sim \rho_X, \qquad Y = \max(0, X), \qquad \widetilde{X} \sim \mathcal{N}(\widetilde{X} | \widetilde{\mu}, \widetilde{\sigma}^2)$$

with $\widetilde{\mu} = \mathbb{E}[Y]$ and $\widetilde{\sigma}^2 = \mathrm{var}[Y]$ the following inequality holds

$$W_1(\rho_Y, \rho_{\widetilde{X}}) \leq \int_{-\infty}^{0} F_{\widetilde{X}}(u) du + W_1(\rho_X, \rho_{\widetilde{X}})$$

where $F_{\widetilde{X}}(\cdot)$ is cdf of $\widetilde{X}$. If $\rho_X = \mathcal{N}(X|\mu,\sigma^2)$, it can be further simplified to

$$W_1(\rho_Y, \rho_{\widetilde{X}}) \leq \widetilde{\sigma}\phi\left(\frac{\widetilde{\mu}}{\widetilde{\sigma}}\right) - \widetilde{\mu}\Phi\left(-\frac{\widetilde{\mu}}{\widetilde{\sigma}}\right) + |\mu - \widetilde{\mu}| + |\sigma - \widetilde{\sigma}|.$$

Proof of Lemma 2. For a generic $X \sim \rho_X$, $Y = \max(0,X)$ and $\widetilde{X} \sim \mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2)$, we have with $F_Y(u) = \mathbb{1}_{u\geq 0} F_X(u)$[2] and (2) that

$$\begin{aligned}
W_1(\rho_Y, \rho_{\widetilde{X}}) &= \int_{\mathbb{R}} \left| F_Y(u) - F_{\widetilde{X}}(u) \right| du \\
&= \int_{-\infty}^{0} F_{\widetilde{X}}(u) du + \int_{0}^{\infty} \left| F_X(u) - F_{\widetilde{X}}(u) \right| du \\
&\leq \int_{-\infty}^{0} F_{\widetilde{X}}(u) du + \int_{0}^{\infty} \left| F_X(u) - F_{\widetilde{X}}(u) \right| du + \int_{-\infty}^{0} \left| F_X(u) - F_{\widetilde{X}}(u) \right| du
\end{aligned}$$

[2]The indicator function $\mathbb{1}$ is defined as $\mathbb{1}_S = 1$ if the statement S is true and 0 otherwise.

https://doi.org/10.52202/079017-2247

$$= \int_{-\infty}^{0} F_{\widetilde{X}}(u)du + W_1(\rho_X, \rho_{\widetilde{X}}).$$

If $\rho_X = \mathcal{N}(X|\mu, \sigma)$, we can upper bound the W_1 distance via Chhachhi and Teng (2023)

$$W_1(\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2)) \leq |\mu_1 - \mu_2| + |\sigma_1 - \sigma_2|$$

and evaluate the integral

$$\int_{-\infty}^{0} F_{\widetilde{X}}(u)du = \int_{-\infty}^{0} \Phi\left(\frac{u - \widetilde{\mu}}{\widetilde{\sigma}}\right) du = \widetilde{\sigma}\phi\left(\frac{\widetilde{\mu}}{\widetilde{\sigma}}\right) - \widetilde{\mu}\Phi\left(-\frac{\widetilde{\mu}}{\widetilde{\sigma}}\right)$$

where we used that

$$\int \Phi(a + bx)dx \stackrel{c}{=} \frac{1}{b}\big((a + bx)\Phi(a + bx) + \phi(a + bx)\big)$$

where $\stackrel{c}{=}$ signifies equality up to an additive constant. Together this gives us that

$$W_1(\rho_Y, \rho_{\tilde{x}}) \leq \sigma\phi\left(\frac{\widetilde{\mu}}{\widetilde{\sigma}}\right) - \widetilde{\mu}\Phi\left(-\frac{\widetilde{\mu}}{\widetilde{\sigma}}\right) + |\mu - \widetilde{\mu}| + |\sigma - \widetilde{\sigma}|.$$

□

Lemma 3 *(Moment matching MLP bound). Let $f(X)$ be an L-layer MLP with ReLU activation $\mathrm{r}(x) = \max(0, x)$. For $l = 1, \ldots, L-1$, the sampling-based forward-pass is*

$$Y_0 = X_s, \qquad Y_l = \mathrm{r}(f_l(Y_{l-1})), \qquad Y_L = f_L(Y_{L-1})$$

where $f_l(\cdot)$ is the l-th layer and X_s a sample of $\mathcal{N}(X|\mu_X, \sigma_X^2)$. Its moment matching pendant is

$$\widetilde{X}_0 \sim \mathcal{N}(\widetilde{X}_0|\mu_X, \sigma_X^2), \qquad \widetilde{X}_l \sim \mathcal{N}\left(\widetilde{X}_l \middle| \mathbb{E}\left[r(f_l(\widetilde{X}_{l-1}))\right], \mathrm{var}\left[r(f_l(\widetilde{X}_{l-1}))\right]\right).$$

The following inequality holds for $\widetilde{\rho}_Y = \rho_{\widetilde{X}_L} = \mathcal{N}(\widetilde{X}_L|\mathbb{E}[f(\widetilde{X}_{L-1})], \mathrm{var}[f(\widetilde{X}_{L-1})])$.

$$W_1(\rho_Y, \widetilde{\rho}_Y) \leq \sum_{l=2}^{L} \big(G(\widetilde{X}_{l-1}) + C_{l-1}\big) \prod_{j=l}^{L} \|A_j\|$$

$$\textit{with} \quad G(\widetilde{X}_l) = \widetilde{\sigma}_l\phi\left(\frac{\widetilde{\mu}_l}{\widetilde{\sigma}_l}\right) - \widetilde{\mu}_l\Phi\left(-\frac{\widetilde{\mu}_l}{\widetilde{\sigma}_l}\right) \leq 1, \qquad C_l \leq |A_l\widetilde{\mu}_{l-1} - \widetilde{\mu}_l| + \left|\sqrt{A_l^2\widetilde{\sigma}_{l-1}^2} - \widetilde{\sigma}_l\right|$$

where $\sqrt{\cdot}$ and $(\cdot)^2$ are applied elementwise.

Proof of Lemma 3. Using Lemma 2 we have that the W_1 distance between the deterministic forward pass ρ_{Y_l} and the matching-based forward pass $\rho_{\widetilde{X}_l}$ is given as

$$\begin{aligned}
W_1(\rho_{Y_l}, \rho_{\widetilde{X}_l}) &\leq G(\widetilde{X}_l) + W_1(\rho_{f_l(Y_{l-1})}, \rho_{\widetilde{X}_l}), && \text{where } G(X) \triangleq \int_{-\infty}^{0} F_X(u)du \\
&\leq G(\widetilde{X}_l) + W_1(\rho_{f_l(Y_{l-1})}, \rho_{f_l(\widetilde{X}_{l-1})}) \\
&\quad + W_1(\rho_{f_l(\widetilde{X}_{l-1})}, \rho_{\widetilde{X}_l}), && \text{via the triangle inequality} \\
&\leq G(\widetilde{X}_l) + \|A_l\| W_1(\rho_{Y_{l-1}}, \rho_{\widetilde{X}_{l-1}}) + C_l, && C_l \triangleq W_1(\rho_{f_l(\widetilde{X}_{l-1})}, \rho_{\widetilde{X}_l}) \\
&\leq G(\widetilde{X}_l) + \|A_l\| \Big(G(\widetilde{X}_{l-1}) \\
&\quad + \|A_{l-1}\| W_1(\rho_{Y_{l-2}}, \rho_{\widetilde{X}_{l-2}}) + C_{l-1} \Big) + C_l.
\end{aligned}$$

That is, for the entire MLP we have

$$\begin{aligned}
W_1(\rho_Y, \widetilde{\rho}_Y) = W_1(\rho_{Y_L}, \rho_{\widetilde{X}_L}) &= \|A_L\| W_1(\rho_{Y_{L-1}}, \rho_{\widetilde{X}_{L-1}}) \\
&\leq \|A_L\| \left(G(\widetilde{X}_{L-1}) + \|A_{L-1}\| W_1(\rho_{Y_{L-2}}, \rho_{\widetilde{X}_{L-2}}) + C_{L-1} \right)
\end{aligned}$$

$$\leq \sum_{l=3}^{L} \left(G(\widetilde{X}_{l-1}) + C_{l-1} \right) \prod_{j=l}^{L} \|A_j\| + W_1(\rho_{Y_1}, \rho_{\widetilde{X}_1}) \prod_{l=2}^{L} \|A_l\|$$

where, finally,

$$W_1(\rho_{Y_1}, \rho_{\widetilde{X}_1}) \leq \sigma \phi \left(\frac{\widetilde{\mu}}{\widetilde{\sigma}} \right) - \widetilde{\mu} \Phi \left(-\frac{\widetilde{\mu}}{\widetilde{\sigma}} \right) + |A_1 \mu_X - \widetilde{\mu}_1| + \left| \sqrt{A_1^2 \sigma_X^2} - \widetilde{\sigma}_1 \right|.$$

$G(\widetilde{X}_l)$ and C_l are given as

$$G(\widetilde{X}_l) = \widetilde{\sigma}_l \phi \left(\frac{\widetilde{\mu}_l}{\widetilde{\sigma}_l} \right) - \widetilde{\mu}_l \Phi \left(-\frac{\widetilde{\mu}_l}{\widetilde{\sigma}_l} \right), \qquad C_l \leq |A_l \widetilde{\mu}_{l-1} - \widetilde{\mu}_l| + \left| \sqrt{A^2 \widetilde{\sigma}_{l-1}^2} - \widetilde{\sigma}_l \right|$$

where $\sqrt{\cdot}$ and $(\cdot)^2$ are applied elementwise. □

Theorem 3 *(Suboptimality of moment matching-based PEVI algorithms). For any policy π_{MM} derived with $\mathbb{A}_{PEVI}(\widetilde{\mathbb{B}}_\pi^\Gamma Q, \widehat{\mathrm{P}})$ learned by a penalization algorithm that uses moment matching to approximate the Bellman target with respect to an action-value network defined as an L-layer MLP with 1-Lipschitz activation functions, the following inequality holds*

$$\mathtt{SubOpt}(\pi_{MM}) \leq 2H \sum_{l=2}^{L} \left(G(\widetilde{X}_{l-1}) + C_{l-1} \right) \prod_{j=l}^{L} \|A_j\|.$$

Proof of Theorem 3. The proof is analogous to the one of Theorem 2. Note that $X = (s, a, s')$, i.e., the state-action pair, is distributed as described in the main text.

Define $C = \sum_{l=2} \left(G(\tilde{X}_{l-1}) + C_{l-1} \right) \prod_{j=l}^{L} \|A_j\|$. By combining this theorem with the results from Theorem 1, we have that

$$\begin{aligned} \left| \mathbb{E}_{X \sim \mathcal{N}(\mu_X, \sigma_X^2)} \left[\widetilde{\mathbb{B}}_\pi Q(X) \right] - \mathbb{E}_{X \sim \mathcal{N}(\tilde{\mu}_X, \tilde{\sigma}_X^2)} \left[\widetilde{\mathbb{B}}_\pi Q(X) \right] \right| &\leq C \\ \left| \mathbb{B}_\pi Q(s,a) - \mathbb{E}_{X \sim \mathcal{N}(\tilde{\mu}_X, \tilde{\sigma}_X^2)} \left[\widetilde{\mathbb{B}}_\pi Q(X) \right] \right| &\leq C \\ \left| \mathbb{B}_\pi Q(s,a) - \widetilde{\mathbb{B}}_\pi Q(s,a,s') \right| &\leq C \end{aligned}$$

holds deterministically. Therefore, C is a ξ-uncertainty quantifier for $\xi = \delta = 0$. Applying Theorem 1 finalizes the proof. □

C Further details on experiments

C.1 Experiment procedures

C.1.1 Moment matching versus Monte Carlo sampling experiment

We sample an initial state s_1 from the environments. Next, we use the expert policies π^*, treated as optimal policies, provided[3] in the D4RL datasets (Fu et al., 2020) to sample action a_1 from the expert policy. Using the learned environment model $\widehat{\mathrm{P}}$, we predict the next state samples $s_2 \sim \widehat{\mathrm{P}}(\cdot|s_1, a_1)$. For each sample, we evaluate the next action-value $Q(s_2, \pi^*(s_2))$ using the previously learned critics as the value function. We then evaluate the next action-value using moment matching $Q_{MM}(s_2, \pi^*(s_2))$ and visualize the distributions. For this experiment, we use the learned critics from seed 0. We repeat this process for all tasks in the D4RL dataset and present the results in Figure 3.

[3] https://rail.eecs.berkeley.edu/datasets/offline_rl/ope_policies/onnx/

https://doi.org/10.52202/079017-2247

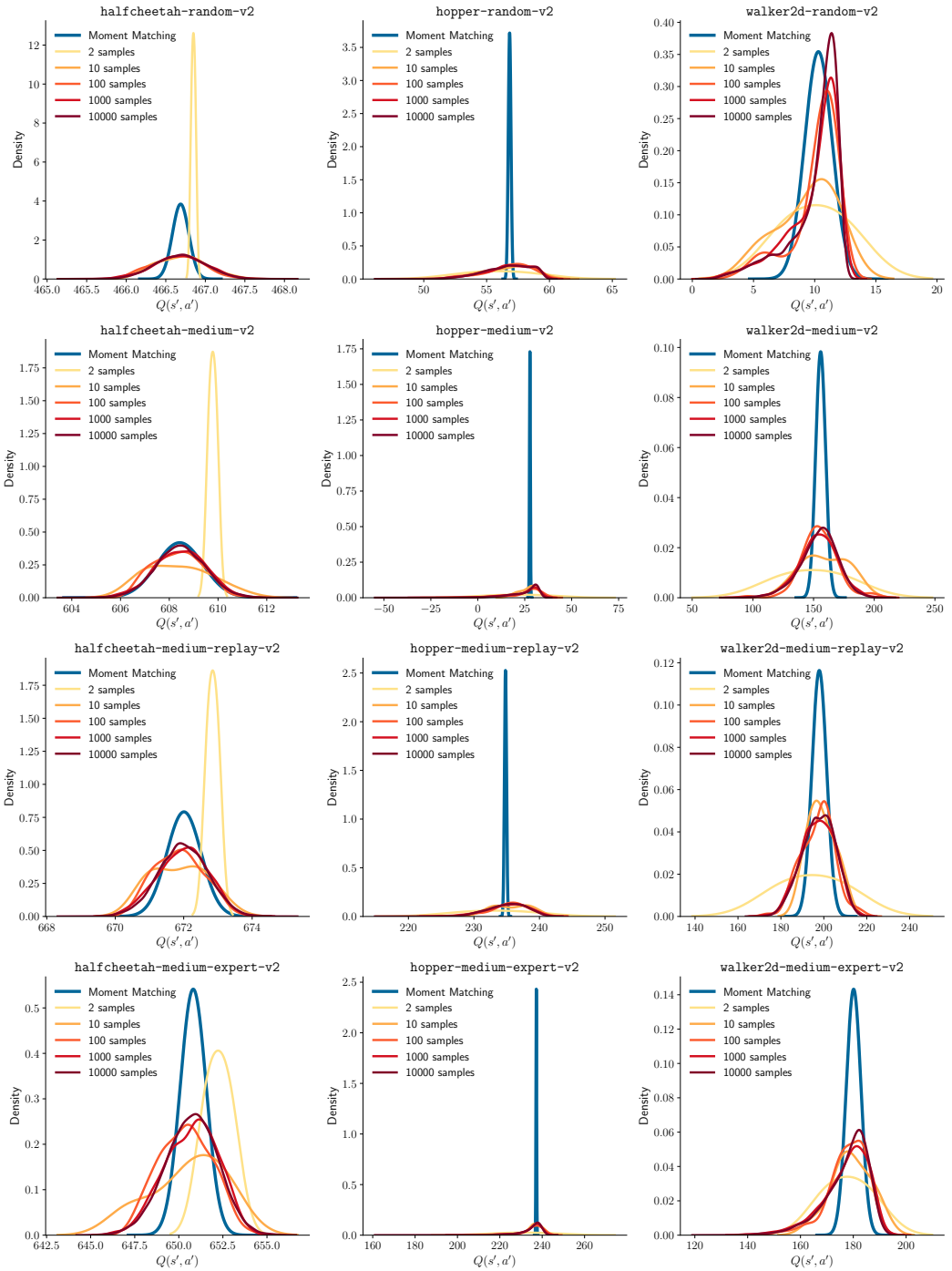

Figure 3: *Moment Matching versus Monte Carlo Sampling.* A comparison of moment matching and Monte Carlo sampling methods for estimating the next value for all tasks in the D4RL dataset.

Table 3: *Performance evaluation on the mixed dataset.* Normalized reward at 2M gradient steps and Area Under the Learning Curve (AULC) (mean$\pm$std) scores are averaged across four repetitions. The highest means are highlighted in bold and are underlined if they fall within one standard deviation of the best score. The average normalized score is the average across all tasks, and the average ranking is based on the rank of the mean.

Dataset Ratio (%)	Dataset Type	Environment	Normalized Reward (↑)			AULC (↑)		
			MOPO	MOBILE	MOMBO	MOPO	MOBILE	MOMBO
1	random-medium	halfcheetah	42.1±12.0	51.3±2.7	**55.4±2.1**	41.4±6.6	46.0±2.1	**48.5±1.3**
		hopper	**48.9±25.7**	22.9±4.0	30.4±2.0	**35.5±3.0**	21.8±2.2	27.3±2.8
		walker2d	0.1±0.0	15.9±6.3	**22.0±0.4**	0.1±0.0	13.1±5.1	**18.7±0.7**
	random-expert	halfcheetah	33.5±3.3	**51.8±2.0**	41.0±16.9	31.2±1.8	**38.2±3.0**	32.3±8.5
		hopper	42.8±32.9	30.7±16.9	**51.5±35.6**	32.3±2.4	**32.7±7.0**	32.0±7.2
		walker2d	−0.1±0.0	21.6±0.0	**21.8±0.4**	0.3±0.7	16.6±0.9	**18.2±1.0**
		Average on 1	27.9	32.4	**37.0**	23.5	28.0	**29.5**
5	random-medium	halfcheetah	62.1±2.7	**62.4±1.4**	62.2±3.5	54.3±1.5	56.4±0.2	**58.1±1.1**
		hopper	32.0±6.7	46.8±11.8	**60.0±27.3**	29.5±3.7	38.3±3.3	**50.1±8.2**
		walker2d	5.2±5.4	10.6±6.5	**21.7±0.1**	4.7±4.5	10.3±4.7	**18.5±0.7**
	random-expert	halfcheetah	33.8±10.2	**54.0±22.0**	39.1±19.7	33.6±2.3	**49.7±10.6**	30.3±9.7
		hopper	**43.6±30.2**	43.0±27.3	17.1±12.6	29.2±4.8	**42.1±4.7**	39.6±3.5
		walker2d	1.6±2.0	13.8±7.4	**24.0±2.6**	2.0±1.2	13.3±3.7	**18.4±1.8**
		Average on 5	29.7	**38.4**	37.4	25.5	35.0	**35.8**
10	random-medium	halfcheetah	66.2±0.7	65.4±0.7	**68.0±1.1**	59.5±1.0	59.9±0.4	**61.2±1.2**
		hopper	37.3±9.0	45.0±29.0	**58.4±14.8**	33.0±2.1	58.4±16.8	**60.1±8.7**
		walker2d	**44.5±5.5**	11.1±6.9	17.8±6.7	**24.5±4.4**	10.9±3.8	16.8±2.9
	random-expert	halfcheetah	29.5±9.8	**74.5±8.2**	58.1±16.8	32.9±3.8	**64.7±2.3**	62.4±6.6
		hopper	20.0±5.7	**80.3±21.4**	77.8±29.5	19.2±2.1	40.3±6.1	**45.5±6.2**
		walker2d	4.2±3.6	14.9±7.9	**21.3±4.0**	2.9±1.6	14.0±3.4	**17.2±1.0**
		Average on 10	33.6	48.5	**50.2**	28.7	41.3	**43.9**
50	random-medium	halfcheetah	73.4±1.4	72.0±0.8	**73.5±1.9**	68.4±0.2	67.3±0.8	**69.3±0.8**
		hopper	67.2±34.5	106.5±2.0	**108.0±0.6**	34.1±14.1	80.9±5.9	**82.9±4.8**
		walker2d	64.4±30.6	36.1±25.0	14.7±8.2	**58.9±16.5**	29.6±3.2	37.7±9.1
	random-expert	halfcheetah	42.1±10.5	**72.9±29.1**	33.4±12.0	30.7±3.6	**47.3±6.7**	44.1±9.6
		hopper	24.7±3.7	70.3±39.8	**88.6±37.9**	22.5±1.8	62.5±9.2	**65.4±8.5**
		walker2d	16.0±11.4	15.7±8.5	**16.4±9.7**	7.7±1.1	9.8±4.8	**11.8±4.1**
		Average on 50	48.0	**62.2**	55.8	37.1	49.6	**51.9**
		Average Score	34.8	**45.4**	45.1	28.7	38.5	**40.3**
		Average Ranking	2.5	2.0	**1.5**	2.6	1.9	**1.5**

C.1.2 Uncertainty quantifier experiment

We generate 10 episode rollouts using the real environment and previously trained policies in evaluation mode. From these rollouts, we select state, action, reward, and next state tuples at every 10^{th} step, including the final step. For each selected tuple, we calculate the mean accuracy and tightness scores using the learned environment models. We repeat this process for each of the 4 seeds across every task and report the mean and standard deviation of the accuracies and tightness scores. We estimate the exact Bellman target by taking samples from the policies, and the number of samples is 1000. Table 2 shows the results of this experiment.

C.1.3 Mixed offline reinforcement learning dataset experiment

The behavior policy of the mixed datasets consists of a combination of `random` and `medium` or `expert` policies, mixed at a specified demonstration ratio. Throughout the experiments, we use the configurations of Cetin et al. (2024). See Appendix C.2 for details regarding the hyperparameters. Results in Table 3 show that MOMBO converges faster with minimal expert input, indicating that it effectively leverages limited information while maintaining competitive performance. Compared to other PEVI approaches, MOMBO demonstrates faster learning, greater robustness, and provides strong asymptotic performance, particularly under the constraints of limited expert knowledge and fixed training budgets, where hyperparameter tuning is not feasible. See Figures 5 to 6 for visualizations of the learning curves.

https://doi.org/10.52202/079017-2247

Table 4: Common hyperparameters and sources used in the experimental pipeline.

	D4RL	Mixed
Sources		
Dataset	Fu et al. (2020)	Hong et al. (2023)
Configuration	Sun et al. (2023)	Hong et al. (2023)
Implementation	Sun et al. (2023)	
Expert policy	Fu et al. (2020)	—
Pretrained environment models	Sun (2023)	—
Policy learning		
Seeds	$[0, 1, 2, 3]$	
Learning rate actor	0.0001	
Learning rate critic	0.0003	
Hidden dimensions actor	$[256, 256]$	$[1024, 1024, 1024, 1024, 1024]$
Hidden dimensions critic	$[256, 256]$	$[256, 256, 256]$,
Discount factor (γ)	0.99	
Soft update (τ)	0.005	
Number of critics	2	
Batch size	256	
Learning rate scheduler	Cosine Annealing for Actor	
Epoch	3000	2000
Number of steps per epoch	1000	
Rollout frequency	1000 steps	
Rollout length (k)	See Table 5	5
Penalty coefficient (β)	See Table 5	2.0
Rollout batch size	50000	
Model retain epochs	5	10
Length of fake buffer $\lvert\widehat{\mathcal{D}}\rvert$	5×50000	10×50000
Real ratio (p)	0.05 (except `halfcheetah-medium-expert-v2` 0.5)	
Environment model learning		
Learning rate	0.001	
Early stop	5	
Hidden dimensions	$[200, 200, 200, 200]$	
L2 weight decay	$[2.5 \times 10^{-5}, 5 \times 10^{-5}, 7.5 \times 10^{-5}, 7.5 \times 10^{-5}, 1 \times 10^{-4}]$	
Number of ensembles N_{ens}	7	
Number of elites N_{elite}	5	

C.2 Hyperparameters and experimental setup

In this section, we provide all the necessary details to reproduce MOMBO. We evaluate MOMBO with four repetitions using the following seeds: $[0, 1, 2, 3]$. We run the experiments using the official repository of MOBILE (Sun et al., 2023),[4] as well as our own model implementation, available at `https://github.com/adinlab/MOMBO`. We list the hyperparameters related to the experimental pipeline in Table 4.

C.2.1 Environment model training

Following prior works (Yu et al., 2020, 2021; Sun et al., 2023), we model the transition model P as a neural network ensemble that predicts the next state and reward as a Gaussian distribution. We formulate this as $\widehat{\mathrm{P}}_\theta(s', r) = \mathcal{N}(\mu_\theta(s, a), \Sigma_\theta(s, a))$, where μ_θ and Σ_θ are neural networks that model the parameters of the Gaussian distribution. These neural networks are parameterized by θ and we learn a diagonal covariance $\Sigma_\theta(a, s)$.

[4] `https://github.com/yihaosun1124/mobile/tree/4882dce878f0792a337c0a95c27f4abf7e926101`

We use $N_{\text{ens}} = 7$ ensemble elements and select $N_{\text{elite}} = 5$ elite elements from the ensemble based on a validation dataset containing 1000 transitions. Each model in the ensemble consists of a 4-layer feedforward neural network with 200 hidden units, and we apply $L2$ weight decay with the following weights for each layer, starting from the first layer: $[2.5 \times 10^{-5}, 5 \times 10^{-5}, 7.5 \times 10^{-5}, 7.5 \times 10^{-5}, 1 \times 10^{-4}]$. We train the environment model using maximum likelihood estimation with a learning rate of 0.001 and a batch size of 256. We apply early stopping with 5 steps using the validation dataset. The only exception is for `walker2d-medium` dataset, which has a fixed number of 30 learning episodes. We use Adam optimizer (Kingma and Ba, 2015) for the environment model.

We use the pre-trained environment models provided in Sun (2023) to minimize differences and reduce training time. For the mixed dataset experiments, we train environment models using four seeds $(0, 1, 2, 3)$ and use them for each baseline.

C.2.2 Policy training

Architecture and optimization details. We train MOMBO for 3000 episodes on the D4RL dataset and 2000 episodes on the mixed dataset, performing updates to the policy and Q-function 1000 times per episode with a batch size of 256. We set the learning rate for the critics to 0.0003, while the learning rate for the actor is 0.0001. We use the Adam optimizer (Kingma and Ba, 2015) for both the actor and critics. We employ a cosine annealing learning rate scheduler for the actor. For the D4RL dataset, both the actor and critics architectures consist of 2-layer feedforward neural networks with 256 hidden units. For the mixed dataset, the actor uses a 5-layer feedforward neural network with 1024 hidden units, and the critics consist of a 3-layer feedforward neural network with 256 hidden units. Unlike other baselines, our critic network is capable of deterministically propagating uncertainties via moment matching. The critic ensemble consists of two networks. For policy optimization, we mostly follow the SAC (Haarnoja et al., 2018), with a difference in the policy evaluation phase. Using MOBILE's configuration, we apply the deterministic Bellman backup operator instead of the soft Bellman backup operator. We set the discount factor to $\gamma = 0.99$ and the soft update parameter to $\tau = 0.005$. The α parameter is learned during training with the learning rate of 0.0001, except for `hopper-medium` and `hopper-medium-replay`, where $\alpha = 0.2$. We set the target entropy to $-\dim(\mathcal{A})$, where dim is the number of dimensions. We train all networks using a random mixture of the real dataset $\mathcal{D}$ and synthetically generated rollouts $\widehat{\mathcal{D}}$ with real and synthetic data ratios of p and $1 - p$, respectively. We set $p = 0.05$, except for `halfcheetah-medium-expert`, where $p = 0.5$.

Synthetic dataset generation for policy optimization. We follow the same procedure for synthetic dataset generation as it is common in the literature. During this phase, we sample a batch of initial states from the real dataset $\mathcal{D}$ and sample the corresponding actions from the current policy. For each state-action pair in the batch, the environment models predict a Gaussian distribution over the next state and reward. We choose one of the elite elements from the ensemble uniformly at random and take a sample from the predicted distribution for the next state and reward. We store these rollout transitions in a synthetic dataset $\widehat{\mathcal{D}}$, where we also keep their variances in the buffer. We generate new rollouts at the beginning of each episode and append them to $\widehat{\mathcal{D}}$. We set the batch size for rollouts to 50000, and store only the rollouts from the last 5 and 10 episodes in $\widehat{\mathcal{D}}$ for the D4RL dataset and the mixed dataset, respectively. For the mixed dataset, the rollout length (k) is 5 for all tasks. Table 5 presents the rollout length for each task for the D4RL dataset.

Penalty coefficients. We adopt all configurations from MOBILE with the exception of β due to the differences in the scale of the uncertainty estimators. For the D4RL dataset, we provide our choices for β in Table 5. For the mixed dataset, we select penalty coefficient β as 2.0 for all tasks in order to apply a penalty corresponding to 95% confidence level. We provide all the other shared hyperparameters in Table 4.

Experiment compute resources. We perform our experiments on three computational resources: 1) Tesla V100 GPU, Intel(R) Xeon(R) Gold 6230 CPU at 2.10 GHz, and 46 GB of memory; 2) NVIDIA Tesla A100 GPU, AMD EPYC 7F72 CPU at 3.2 GHz, and 256 GB of memory; and 3) GeForce RTX 4090 GPU, Intel(R) Core(TM) i7-14700K CPU at 5.6 GHz, and 96 GB of memory. We measure the computation time for 1000 gradient steps to be approximately 8 seconds for the D4RL dataset and 11 seconds for the mixed dataset on the third device. Assuming the environment models

https://doi.org/10.52202/079017-2247

Table 5: Hyperparameters of MOMBO on the D4RL dataset.

Dataset Type	Environment	Rollout Length k	Penalty Coefficient β
random	halfcheetah	5	0.5
	hopper	5	4.5
	walker2d	5	2.5
medium	halfcheetah	5	0.75
	hopper	5	3.0
	walker2d	5	0.75
medium-replay	halfcheetah	5	0.2
	hopper	5	0.15
	walker2d	1	0.5
medium-expert	halfcheetah	5	1.0
	hopper	5	1.5
	walker2d	1	1.5

are provided and excluding evaluation time, the total time required to reproduce all MOMBO repetitions is approximately 330 hours for D4RL and 600 hours for the mixed dataset. The computational cost for other experiments is negligible.

C.3 Visualizations of learning curves

Figures 4 to 6 illustrate the learning curves of PEVI-based approaches, including ours, across gradient steps. In the figures, the thick (dashed/dotted/solid) curve represents the mean normalized rewards across ten evaluation episodes and four random seeds, with the shaded area indicating one standard deviation from the mean. The legend provides the mean and standard deviation of the normalized reward and AULC scores in this order. We show the results for halfcheetah in the left panel, hopper in the middle panel, and walker2d in the right panel. The horizontal axis represents gradient steps and the vertical axis shows normalized episode rewards.

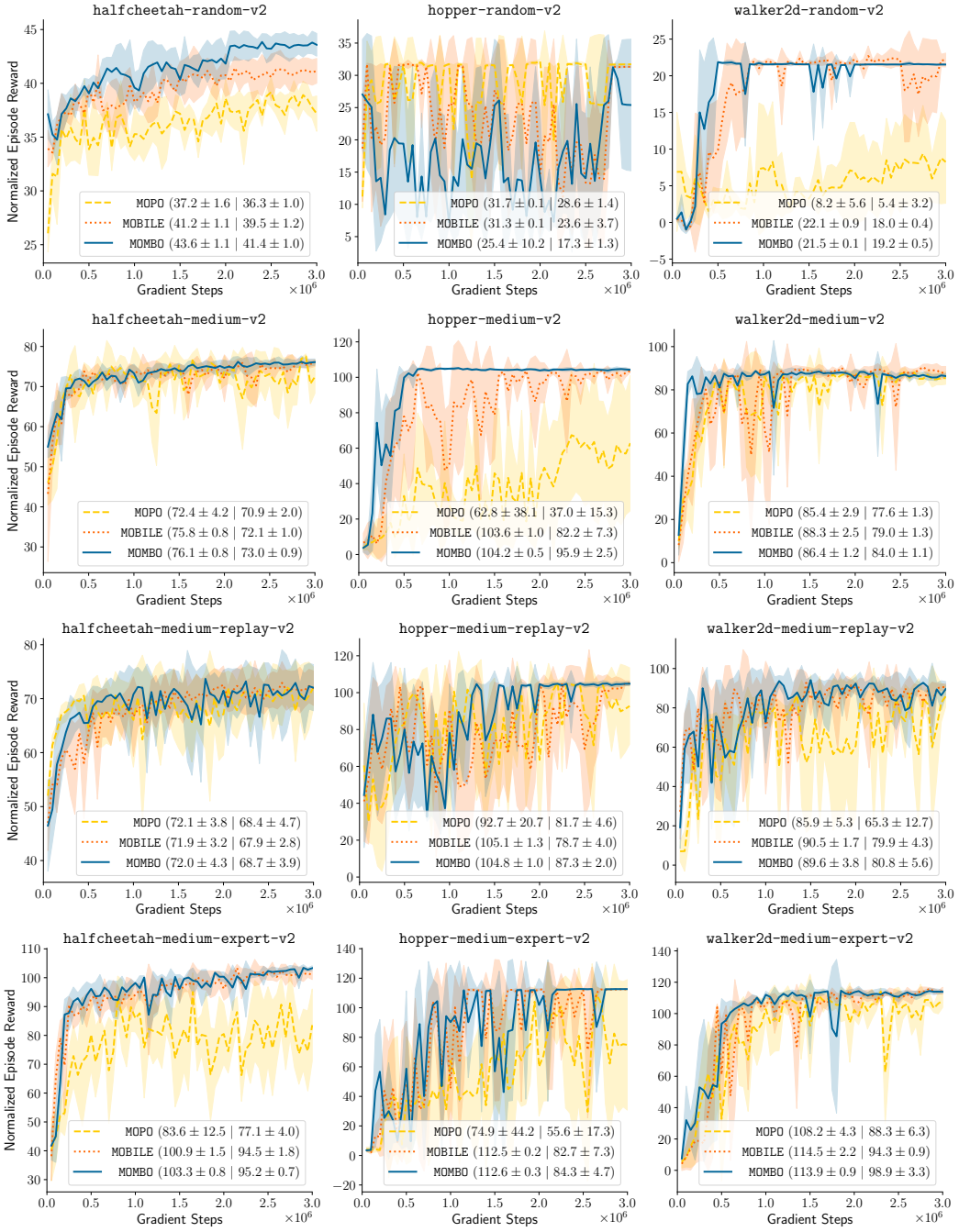

Figure 4: Learning curves for the D4RL dataset.

https://doi.org/10.52202/079017-2247

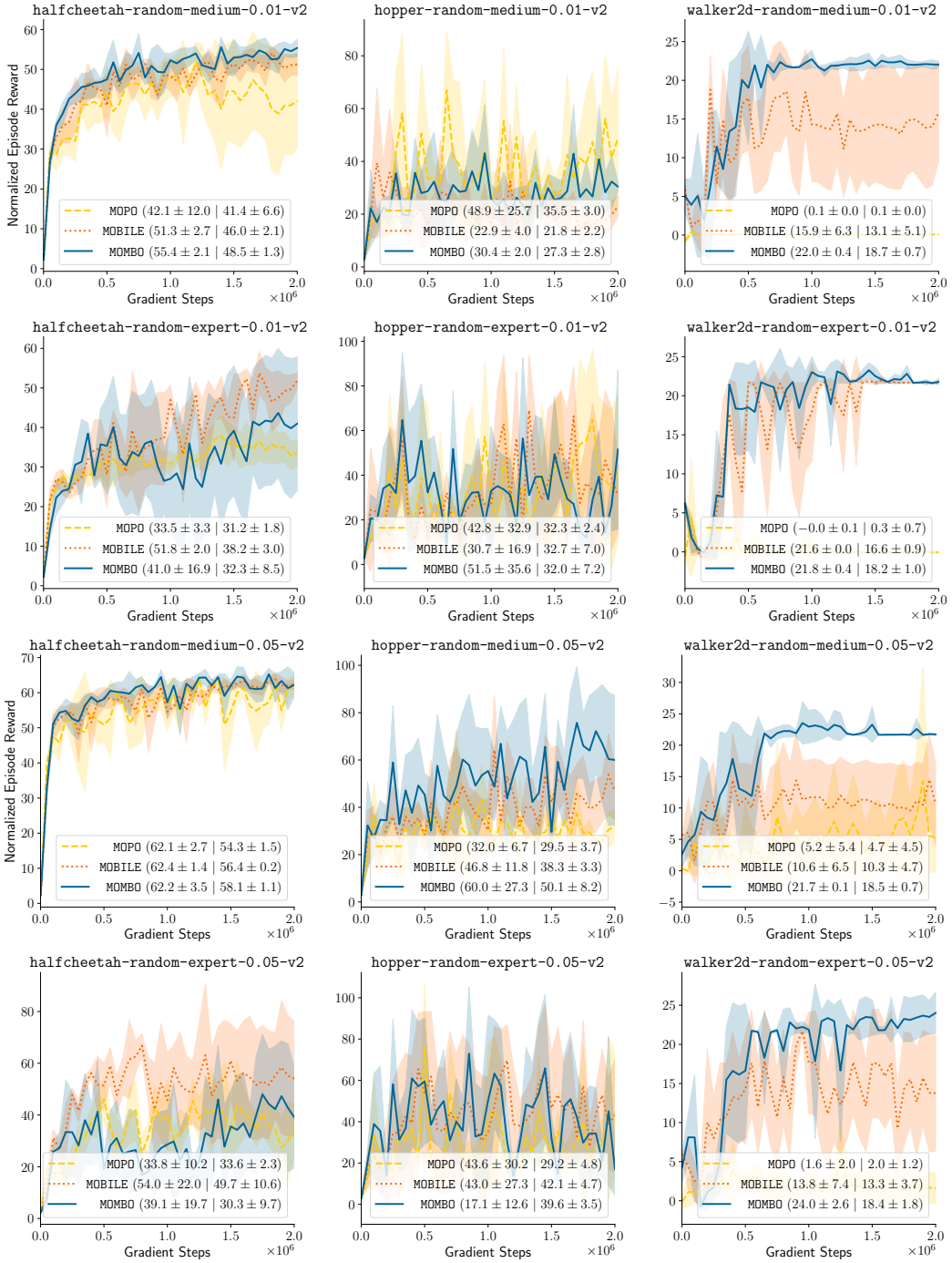

Figure 5: Learning curves for the mixed dataset with trained policy demonstration ratios of `0.01` and `0.05`.

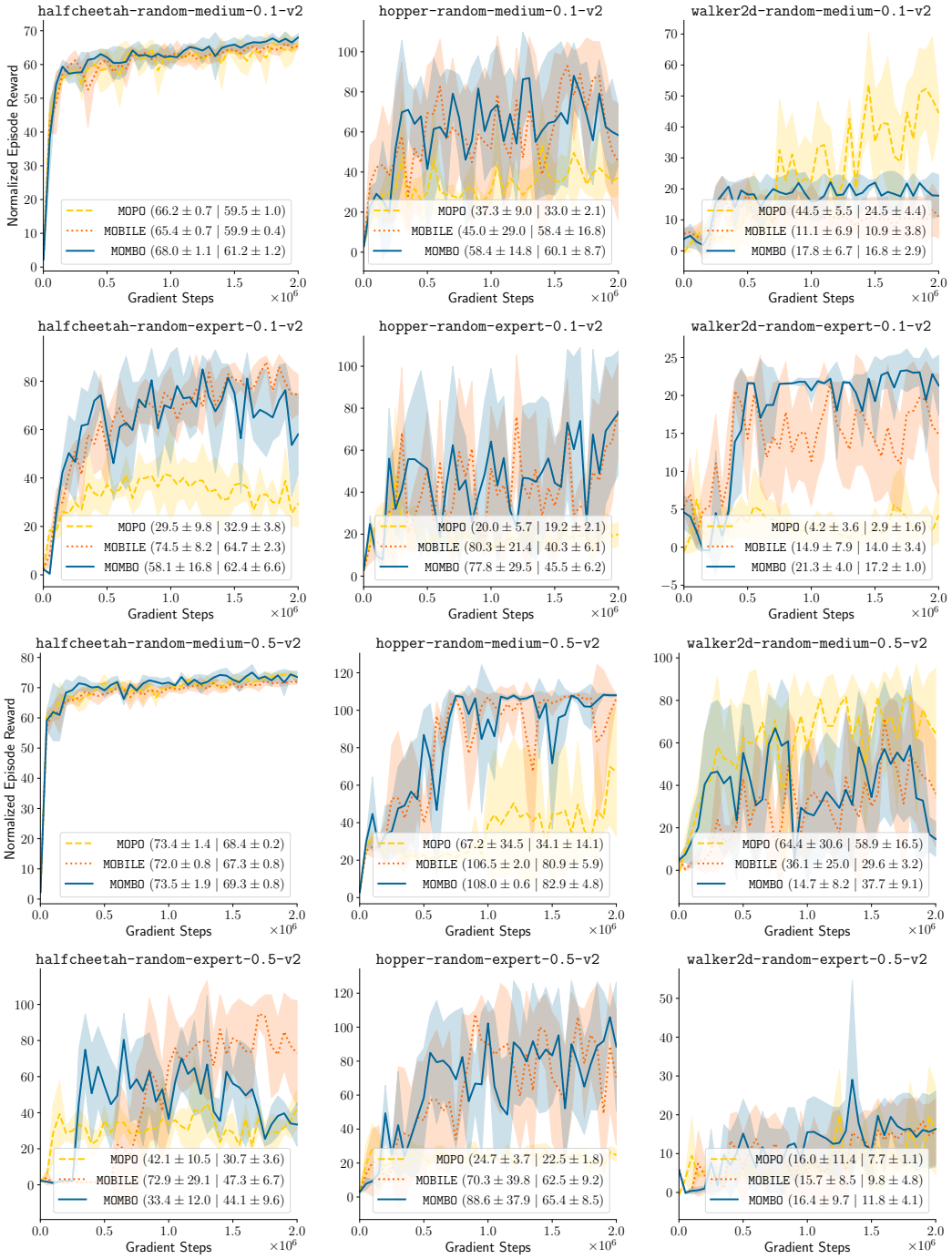

Figure 6: Learning curves for the mixed dataset with trained policy demonstration ratios of `0.1` and `0.5`.

https://doi.org/10.52202/079017-2247

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We summarize our contribution in the abstract and in greater detail in the last paragraph of the introduction. These are theoretical (discussed in Section 3 and Section 4) and empirical (discussed in Section 5).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of the approach as part of our concluding discussion in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide all assumptions in the respective theoretical lemmata and theorems. Appendix B includes all statements together with their respective proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We highlight all experimental details and hyperparameters to ensure reproducibility in Appendix C. The experimental benchmarks come from standard libraries and for the public repository we base part of our evaluation on we reference the specific commit we worked with. Additionally, we make an implementation of our model available at `https://github.com/adinlab/MOMBO`.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

https://doi.org/10.52202/079017-2247

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release a version of our model. The data is generated from standard public libraries for this specific task which are referenced in Appendix C.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the necessary details about the experimental pipeline in Appendix C, along with the source code needed to reproduce our results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars for each experiment together with details on what the respective numbers mean. We report mean $\pm$ one standard deviation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the computational resources and provide an approximate estimate of the computation time for all the experiments in Appendix C, in the paragraph titled "Experiment compute resources".

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We read the Code of Ethics carefully and ensured that our work complies with it. As the research is foundational we do not anticipate any ethical problems.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

https://doi.org/10.52202/079017-2247

Justification: The paper focuses on foundational research. As such there is no immediate short term social impact of the work to be expected, neither positive nor negative.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: As our research is foundational research there is no safeguard needed at the current level of research.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We do not rely on any fixed data sets. The public libraries and the repository that we rely on are referrenced and explicitly acknowledged.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: There are no new assests released with this work.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing or research with human subjects was performed as part of this project.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No crowdsourcing or research with human subjects was performed as part of this project. As such no IRB approval was necessary.

https://doi.org/10.52202/079017-2247

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.