A Theory

A.1 H-step lookahead with approximation error

We aim to show that H-step model-based lookahead policies are more robust to certain types of approximation errors than 1-step greedy policies given an approximate value function. We restate Theorem 1 here for convenience and then provide a proof.

Theorem 1. *(H-step lookahead policy) Suppose* $\hat{M}$ *is an approximate dynamics model such that* $\max_{s,a} D_{TV}\left(M(.|s,a), \hat{M}(.|s,a)\right) \leq \epsilon_m$. *Let* $\hat{V}$ *be an approximate value function such that* $\max_s|V^*(s) - \hat{V}(s)| \leq \epsilon_v$. *Let the reward function by bounded in [0,*R_{max}*] and* $\hat{V}$ *be bounded in [0,*V_{max}*]. Let* ϵ_p *be the suboptimality incurred in H-step lookahead optimization (Eqn. 3) such that* $J^* - \hat{J} \leq \epsilon_p$, *where* J^* *is the optimal return for the H-step optimization and* $\hat{J}$ *is the result of the suboptimal H-step optimization. Then the performance of the H-step lookahead policy* $\pi_{H,\hat{V}}$ *can be bounded as:*

$$J^{\pi^*} - J^{\pi_{H,\hat{V}}} \leq \frac{2}{1-\gamma^H}[C(\epsilon_m, H, \gamma) + \frac{\epsilon_p}{2} + \gamma^H \epsilon_v]$$

where

$$C(\epsilon_m, H, \gamma) = R_{\max} \sum_{t=0}^{H-1} \gamma^t t \epsilon_m + \gamma^H H \epsilon_m V_{max}$$

Proof. Assume we have an ϵ_v-approximate value function i.e $\|\hat{V} - V^*\|_\infty < \epsilon_v$ and we have an approximate transition model which satisfies $D_{TV}\left(M(.|s,a), \hat{M}(.|s,a)\right) \leq \epsilon_m$, similar to assumptions in [19, 18]. We analyze the optimality gap of the policy which uses an H-step lookahead optimization (Eqn. 3) with this approximate model and value function. First, we define some useful notations: let $\mathcal{M}$ be the MDP defined by $(\mathcal{S}, \mathcal{A}, M, r, s_0)$ which uses the ground truth dynamics M, state space $\mathcal{S}$, action space $\mathcal{A}$, reward function r and starting state s_0, and let $\hat{\mathcal{M}}$ be the MDP defined by $(\mathcal{S}, \mathcal{A}, \hat{M}, r, s_0)$ which uses the approximate dynamics model $\hat{M}$. Correspondingly, let $\mathcal{H}$ be an H-step finite horizon MDP given by $(\mathcal{S}, \mathcal{A}, M, r_{\text{mix}}, s_0)$ and let $\hat{\mathcal{H}}$ be an H-step finite horizon MDP given by $(\mathcal{S}, \mathcal{A}, \hat{M}, r_{\text{mix}}, s_0)$ where

$$r_{\text{mix}}(s_t, a_t) = \begin{cases} r(s,a) & \text{if } t < H \\ \hat{V}(s_H) & \text{if } t = H \end{cases} \tag{11}$$

We redefine $\pi_{H,\hat{V}}$ to be the policy obtained by repeatedly optimizing for the H-step lookahead objective (Eqn. 3) in $\hat{\mathcal{H}}$ and acting for H steps in $\mathcal{M}$. We do not consider the MPC setting for simplicity in proof i.e. the policy does not perform any replanning after taking its initial actions. We will use $\pi^*_{\mathcal{K}}$ denote the optimal policy for some MDP $\mathcal{K}$. Let $\hat{\tau}$ denote an H-step trajectory sampled by running $\pi^*_{\hat{\mathcal{H}}}$ in $\mathcal{M}$ and similarly τ is used to denote an H-step trajectory sampled by running $\pi^*_{\mathcal{H}}$ in $\mathcal{M}$. Let τ^* denote the H-step trajectory sampled by running $\pi^*_{\mathcal{M}}$ in $\mathcal{M}$. Let $p_{\hat{\tau}}$, p_τ and p_{τ^*} be the corresponding trajectory distributions. The performance gap we want to upper bound is given by:

$$J^{\pi^*} - J^{\pi_{H,\hat{V}}} = V^*(s_0) - V^{\pi_{H,\hat{V}}}(s_0) \tag{12}$$

$$= \mathbb{E}_{\tau^* \sim p_{\tau^*}}\left[\sum \gamma^t r(s_t,a_t) + \gamma^H V^*(s_H)\right] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}}\left[\sum \gamma^t r(s_t,a_t) + \gamma^H V^{\pi_{H,\hat{V}}}(s_H)\right] \tag{13}$$

$$= \mathbb{E}_{\tau^* \sim p_{\tau^*}}\left[\sum \gamma^t r(s_t,a_t) + \gamma^H V^*(s_H)\right] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}}\left[\sum \gamma^t r(s_t,a_t) + \gamma^H V^*(s_H)\right] \tag{14}$$

$$+ \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}}\left[\sum \gamma^t r(s_t,a_t) + \gamma^H V^*(s_H)\right] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}}\left[\sum \gamma^t r(s_t,a_t) + \gamma^H V^{\pi_{H,\hat{V}}}(s_H)\right] \tag{15}$$

$$= \mathbb{E}_{\tau^* \sim p_{\tau^*}}\left[\sum \gamma^t r(s_t,a_t) + \gamma^H V^*(s_H)\right] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}}\left[\sum \gamma^t r(s_t,a_t) + \gamma^H V^*(s_H)\right] \tag{16}$$

$$+ \gamma^H \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}}[V^*(s_H) - V^{\pi_{H,\hat{V}}}(s_H)] \tag{17}$$

Since we have $|V^*(s) - \hat{V}(s)| \leq \epsilon_v \; \forall s$, we can bound the following expressions:

$$\mathbb{E}_{\tau^* \sim p_{\tau^*}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H V^*(s_H) \Big] \leq \mathbb{E}_{\tau^* \sim p_{\tau^*}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] + \gamma^H \epsilon_v \tag{18}$$

$$\mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H V^*(s_H) \Big] \geq \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] - \gamma^H \epsilon_v \tag{19}$$

Subtracting these two inequalities (18 and 19), we get:

$$\begin{aligned} &\mathbb{E}_{\tau^* \sim p_{\tau^*}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H V^*(s_H) \Big] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H V^*(s_H) \Big] \\ \leq\; &\mathbb{E}_{\tau^* \sim p_{\tau^*}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] + 2\gamma^H \epsilon_v \end{aligned} \tag{20}$$

Substituting Eqn. 20 into Eqn. 16 we can bound the performance gap as follows:

$$\begin{aligned} &J^{\pi^*} - J^{\pi_{H,\hat{V}}} = V^*(s_0) - V^{\pi_{H,\hat{V}}}(s_0) \\ &\leq \mathbb{E}_{\tau^* \sim p_{\tau^*}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] && (21) \\ &+ 2\gamma^H \epsilon_v + \gamma^H \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} [V^*(s_H) - V^{\pi_{H,\hat{V}}}(s_H)] \\ &= \mathbb{E}_{\tau^* \sim p_{\tau^*}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] - \mathbb{E}_{\tau \sim p_{\tau}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] && (22) \\ &+ \mathbb{E}_{\tau \sim p_{\tau}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] \\ &+ 2\gamma^H \epsilon_v + \gamma^H \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} [V^*(s_H) - V^{\pi_{H,\hat{V}}}(s_H)] && (23) \\ &\leq \mathbb{E}_{\tau \sim p_{\tau}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] && (24) \\ &+ 2\gamma^H \epsilon_v + \gamma^H \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} [V^*(s_H) - V^{\pi_{H,\hat{V}}}(s_H)] && (25) \end{aligned}$$

The last step is due to the fact that τ is generated by the optimal action sequence in the *ground-truth* H-step MDP $\mathcal{H}$ as defined earlier which implies that $\mathbb{E}_{\tau^* \sim p_{\tau^*}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] \leq \mathbb{E}_{\tau \sim p_{\tau}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big]$

Now we aim to characterize the performance gap between an optimal policy of MDP $\hat{\mathcal{H}}$, $\pi^*_{\hat{\mathcal{H}}}$, with the optimal policy of MDP $\mathcal{H}$, $\pi^*_{\mathcal{H}}$, evaluating both in the ground truth MDP $\mathcal{H}$. We wish to characterize this performance gap as a function of model errors and value errors $f(\epsilon_m, \epsilon_v, \gamma, H)$:.

$$\mathbb{E}_{\tau \sim p_{\tau}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] \leq f(\epsilon_m, \epsilon_v, \gamma, H)$$

Let $J^{\pi}_{\mathcal{H}}$ denote the performance of policy π when evaluated in MDP $\mathcal{H}$ starting from same initial state s_0. Then we can write this performance gap as

$$\begin{aligned} &\mathbb{E}_{\tau \sim p_{\tau}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] - \mathbb{E}_{\hat{\tau} \sim p_{\hat{\tau}}} \Big[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H) \Big] && (26) \\ &= J^{\pi^*_{\mathcal{H}}}_{\mathcal{H}} - J^{\pi^*_{\hat{\mathcal{H}}}}_{\mathcal{H}} && (27) \\ &= J^{\pi^*_{\mathcal{H}}}_{\mathcal{H}} - J^{\pi^*_{\mathcal{H}}}_{\hat{\mathcal{H}}} + J^{\pi^*_{\mathcal{H}}}_{\hat{\mathcal{H}}} - J^{\pi^*_{\hat{\mathcal{H}}}}_{\hat{\mathcal{H}}} + J^{\pi^*_{\hat{\mathcal{H}}}}_{\hat{\mathcal{H}}} - J^{\pi^*_{\hat{\mathcal{H}}}}_{\mathcal{H}} && (28) \\ &= \Big(J^{\pi^*_{\mathcal{H}}}_{\mathcal{H}} - J^{\pi^*_{\mathcal{H}}}_{\hat{\mathcal{H}}} \Big) - \Big(J^{\pi^*_{\hat{\mathcal{H}}}}_{\mathcal{H}} - J^{\pi^*_{\hat{\mathcal{H}}}}_{\hat{\mathcal{H}}} \Big) + \Big(J^{\pi^*_{\mathcal{H}}}_{\hat{\mathcal{H}}} - J^{\pi^*_{\hat{\mathcal{H}}}}_{\hat{\mathcal{H}}} \Big) && (29) \\ &\leq \Big(J^{\pi^*_{\mathcal{H}}}_{\mathcal{H}} - J^{\pi^*_{\mathcal{H}}}_{\hat{\mathcal{H}}} \Big) - \Big(J^{\pi^*_{\hat{\mathcal{H}}}}_{\mathcal{H}} - J^{\pi^*_{\hat{\mathcal{H}}}}_{\hat{\mathcal{H}}} \Big) + \epsilon_p && (30) \\ &\leq 2 \max_{\pi \in \{\pi^*_{\mathcal{H}}, \pi^*_{\hat{\mathcal{H}}}\}} \Big| \Big(J^{\pi}_{\mathcal{H}} - J^{\pi}_{\hat{\mathcal{H}}} \Big) \Big| + \epsilon_p && (31) \end{aligned}$$

The second-to-last equation is due to the assumed suboptimality of H-step lookahead planner where we have $\forall$ policies π, $J^{\pi^*_{\hat{\mathcal{H}}}}_{\hat{\mathcal{H}}} + \epsilon_p \geq J^{\pi}_{\hat{\mathcal{H}}}$. Since the total variation between M and $\hat{M}$ is at most ϵ_m, i.e $D_{TV}\Big(M(.|s,a), \hat{M}(.|s,a)\Big) \leq \epsilon_m$, we have that $|\rho^t_1(s,a) - \rho^t_2(s,a)| \leq t\epsilon_m$, where $\rho_1(s,a)$

is the discounted state-action visitation induced by π on $\mathcal{H}$, $\rho_2(s,a)$ is the discounted state-action visitation induced by the same policy on $\hat{\mathcal{H}}$ and superscript t indicates the state-action marginal at the t^{th} timestep (for proof see Lemma B.2 Markov Chain TVD Bound [13]). Then we can write the performance of policy π in terms of its induced state marginal and the reward function, i.e $J^{\pi}_{\mathcal{H}} = \sum_{s,a}\rho_1(s,a)r_{\text{mix}}(s,a) = \sum_{s,a}\sum_{t=0}^{H}\gamma^t\rho_1^t(s,a)r_{\text{mix}}(s,a)$ and use the Markov chain TVD bound:

$$
\begin{aligned}
J^{\pi}_{\mathcal{H}} - J^{\pi}_{\hat{\mathcal{H}}} &= \sum_{s,a}(\rho_1(s,a)-\rho_2(s,a))r_{\text{mix}}(s,a) && (32)\\
|J^{\pi}_{\mathcal{H}} - J^{\pi}_{\hat{\mathcal{H}}}| &= |\sum_{s,a}(\rho_1(s,a)-\rho_2(s,a))r_{\text{mix}}(s,a)| && (33)\\
&= |\sum_{s,a}\sum_{t=0}^{H}\gamma^t(\rho_1^t(s,a)-\rho_2^t(s,a))r^t_{\text{mix}}(s,a)| && (34)\\
&\leq \sum_{s,a}\sum_{t=0}^{H}\gamma^t|(\rho_1^t(s,a)-\rho_2^t(s,a))|r^t_{\text{mix}}(s,a) && (35)\\
&\leq R_{\max}\sum_{t=0}^{H-1}\gamma^t t\epsilon_m + \gamma^H H\epsilon_m V_{\max} && (36)\\
&= C(\epsilon_m, H, \gamma) && (37)
\end{aligned}
$$

Combining Eqn. 31 and Eqn. 37 we have:

$$
\mathbb{E}_{\tau\sim p_\tau}\left[\sum\gamma^t r(s_t,a_t)+\gamma^H\hat{V}(s_H)\right] - \mathbb{E}_{\hat{\tau}\sim p_{\hat{\tau}}}\left[\sum\gamma^t r(s_t,a_t)+\gamma^H\hat{V}(s_H)\right] \leq 2C(\epsilon_m,H,\gamma)+\epsilon_p \tag{38}
$$

We substitute Eqn. 38 in Eqn. 24. Also observe that the last term in Eqn. 24 $\gamma^H\mathbb{E}_{\hat{\tau}}[V^*(s_H) - V^{\pi_{H,\hat{V}}}(s_H)]$ can be bounded recursively. Then, we will have the following optimality gap for the H-step lookahead policy $\pi_{H,\hat{V}}$:

$$
J^{\pi^*} - J^{\pi_{H,\hat{V}}} \leq \frac{2}{1-\gamma^H}[C(\epsilon_m,H,\gamma)+\frac{\epsilon_p}{2}+\gamma^H\epsilon_v] \tag{39}
$$

The H-step lookahead policy $\pi_{H,\hat{V}}$ reduces the dependency on ϵ_v (the maximum error of the value function) by a factor of γ^H and introduces an additional dependency on ϵ_m (the maximum error of the model). In contrast, when we use 1-step greedy policy, the performance gap is bounded by (Lemma 1):

$$
J^{\pi^*} - J^{\pi_{H,\hat{V}}} \leq \frac{\gamma}{1-\gamma}[2\epsilon_v] \tag{40}
$$

Lemma 1 can be seen as a special case of our bound when ϵ_m is set to 0 and H is set to 1. □

A.2 H-step lookahead with model generalization error

In this section, we derive a similar proof as the previous section with a weaker assumption on model error. We consider a model trained by supervised learning where the sample error can be computed by PAC generalization bounds which bounds the expected loss and empirical loss under a dataset with high probability.

We define D to be the dataset of transitions and π_D to be the data collecting policy.

Corollary 1. *(H-step lookahead with function approximation) Suppose* $\hat{M}$ *is an approximate dynamics model such that* $\max_t \mathbb{E}_{s\sim\pi_{D,t}}\left[D_{TV}(M(.|s,a)\|\hat{M}(.|s,a))\right] \leq \tilde{\epsilon}_m$. *Let* $\hat{V}$ *be an approximate value function such that* $\max_s|V^*(s)-\hat{V}(s)|\leq \epsilon_v$. *Let the maximum TV distance of state distribution visited by lookahead policy* $\pi_{H,\hat{V}}$ *be bounded wrt state visitation of data generating policy by* $\max_t \mathbb{E}_{s\sim\pi_{H,\hat{V}}}\left[D_{TV}(\rho^t_{\pi_{H,\hat{V}}}\|\rho^t_{\pi_D})\right] \leq \epsilon_i$ *and* $\max\Big(D_{TV}(\pi_D(a|s)||\pi^*_H(a|s)), D_{TV}(\pi_D(a|s)||\pi^*_{\hat{H}}(a|s))\Big) \leq \tilde{\epsilon_\pi}$ $\forall s$. *Let the reward function by* *bounded in [0,R_{max}] and $\hat{V}$ be bounded in [0,V_{max}]. Then the performance of the H-step lookahead policy $\pi_{H,\hat{V}}$ can be bounded as:*

$$J^{\pi^*} - J^{\pi_{H,\hat{V}}} \le \frac{2}{1-\gamma^H}[C(\tilde{\epsilon_m}, \tilde{\epsilon_\pi}, \epsilon_i, H, \gamma) + \gamma^H \epsilon_v]$$

where

$$C(\tilde{\epsilon_m}, \tilde{\epsilon_\pi}, H, \gamma) = R_{\max} \sum_{t=0}^{H-1} \gamma^t t(\tilde{\epsilon_m} + \tilde{\epsilon_\pi}) + R_{\max}\epsilon_i + \gamma^H H(\tilde{\epsilon_m} + \tilde{\epsilon_\pi}) V_{max}$$

Proof. In the function approximation setting, a more realistic perfomance bound depends on the generalization error of model and distribution shift for the new policy under the collected dataset of transitions D. Let π_D be the data collecting policy. Let $\mathbb{E}_{s\sim\pi_{D,t}}\Big[D_{TV}(M(.|s,a)||\hat{M}(.|s,a))\Big] \le \tilde{\epsilon}_m \ \ \forall s$ and $\max\Big(D_{TV}(\pi_D(a|s)||\pi^*_H(a|s)), D_{TV}(\pi_D(a|s)||\pi^*_{\hat{H}}(a|s))\Big) \le \tilde{\epsilon_\pi} \ \ \forall s$. Following Lemma B.2 Markov Chain TVD Bound [13] with model generalization error $\tilde{\epsilon}_m$, policy distribution shift $\tilde{\epsilon}_\pi$ and bounded state visitation of lookahead policy by ϵ_i, we have: $|\rho_1^t(s,a) - \rho_2^t(s,a)| \le t(\tilde{\epsilon}_m + \tilde{\epsilon}_\pi) + \epsilon_i$ Substituting the new state-action divergence bound in Eqn. 35 from Theorem 1 we get the following performance bound:

$$J^{\pi^*} - J^{\pi_{H,\hat{V}}} \le \frac{2}{1-\gamma^H}[C(\tilde{\epsilon_m}, \tilde{\epsilon_\pi}, \epsilon_i, H, \gamma) + \gamma^H \epsilon_v] \tag{41}$$

where $C(\tilde{\epsilon_m}, \tilde{\epsilon_\pi}, H, \gamma) = R_{\max} \sum_{t=0}^{H-1} \gamma^t t(\tilde{\epsilon_m} + \tilde{\epsilon_\pi}) + R_{\max}\epsilon_i + \gamma^H H(\tilde{\epsilon_m} + \tilde{\epsilon_\pi}) V_{\max}$.

Intuitively this bound highlights the tradeoff between model error and value error reasonably when the dataset is sufficiently exploratory to cover π^*_H and H-step lookahead policy has visitation close to the dataset.

□

A.3 H-step lookahead with Empirical Dataset Distribution using Fitted-Q Iteration

In this section, we take a look at the analysis of H-step lookahead under a set of different assumptions. In particular, we assume a form of model generalization error and that the optimal H-step trajectory is obtained via fitted-Q iteration in the H-step MDP at every timestep during policy deployment. This analysis largely follows the fitted-Q iteration analysis from [60, 61, 41] but we adapt it to H-step lookahead in a simplified form.

Assumption 1. *Let our replay buffer dataset be denoted by D and the data generating distribution be given by d^{π_D}, where π_D is the data generating policy. Let the Q-function class is given by $\mathcal{Q} \subset \mathbb{R}^{S\times A}$. The empirical bellman update $\hat{\mathcal{T}}Q$ under the learned model is given by:*

$$L_{d^{\pi_{\hat{M}}}}(Q, Q^k) = \mathbb{E}_{s,a,r,s'\sim d^{\pi_{\hat{M}}}}\left[(Q(s,a) - r - \gamma Q^k(s', \pi_Q(s')))^2\right] \tag{42}$$

where Q^k is the Q-function at k iteration, $d^{\pi_{\hat{M}}}$ is the state visitation under a learned model $\hat{M}$ from dataset D. Also we define:

$$L_{d^{\pi_D}}(Q, Q^k) = \mathbb{E}_{s,a,r,s'\sim d^{\pi_D}}\left[(Q(s,a) - r - \gamma Q^k(s', \pi_Q(s')))^2\right] \tag{43}$$

A form of model generalization error: We assume the following uniform deviation bound which holds with high probability ($\ge 1-\delta$):

$$\forall Q, Q^k,\ |L_D(Q, Q^k) - L_{d^{\pi_D}}(Q, Q^k)| \le \tilde{\epsilon}_m \tag{44}$$

This bound can be obtained by concentration inequality as in [41] using concentration inequality $\tilde{\epsilon}_m$ to be a function of size of dataset $|D|$, δ and size of function space for $\mathcal{Q}$.

Intuitively the assumption above states that the bellman error obtained in the data-generating distribution is close to the bellman error obtained via state-action distribution induced by the learned model, where the model is learned on a finite fixed dataset D sampled from data generating distribution.

In the following analysis, we assume that H-step lookahead policy is obtained by performing fitted-Q iteration in the H-step approximate MDP $\hat{\mathcal{H}}$ defined in Theorem 1.

Theorem 2. *Suppose $\hat{M}$ is an approximate dynamics model such that Assumption 1 holds. Let $\hat{V}$ be an approximate value function such that $\max_s |V^*(s) - \hat{V}(s)| \leq \epsilon_v$. Let the reward function by bounded in [0,R_{max}] and $\hat{V}$ be bounded in [0,V_{max}]. Let concentrability coefficient $\tilde{C}$ be such that $\forall s, a$ $\frac{\nu(s,a)}{d^{\pi_D}(s,a)} \leq \tilde{C}$ where $\nu(s,a)$ is state-action distribution induced by any non-stationary policy. Then the performance of the H-step lookahead policy $\pi_{H,\hat{V}}$ obtained by running fitted-Q iteration on the learned model to convergence can be bounded as:*

$$J^{\pi^*} - J^{\pi_{H,\hat{V}}} \leq \frac{2}{1-\gamma^H}[C(\tilde{\epsilon}_m, \tilde{C}, H, \gamma) + \gamma^H \epsilon_v]$$

where

$$C(\tilde{\epsilon}_m, \tilde{C}, H, \gamma) = \frac{2(1-\gamma^H)}{1-\gamma}\left(\frac{1}{1-\gamma}\sqrt{2\tilde{\epsilon}_m \tilde{C}}\right)$$

Proof. In this section we analyze the performance of H-step lookahead policies under the assumptions for Fitted Q Iteration [41]. This analysis extends the fitted-Q iteration analysis from greedy to H-step lookahead policies.

Let $\|g\|_{p,\nu}$ denote a weighted p-norm under distribution ν given by $\|g\|_{p,\nu} = \mathbb{E}_{s\sim\nu}[|g(s)|^p]^{\frac{1}{p}}$. We start by reusing the previous analysis in Theorem 1 under the new stated assumptions to replace the bound for Eqn. 26. Let $\pi^*_{\hat{\mathcal{H}}}$ be denoted by $\hat{\pi}_H$ and $\pi^*_{\mathcal{H}}$ by π^*_H for ease of notation. In this analysis $\hat{\pi}_H$ is the 1-step greedy policy obtained from Q_k the learned Q-function after k iterations of fitted-Q iteration on the H-step MDP $\hat{\mathcal{H}}$.

Rewriting Eqn. 26:

$$\begin{aligned}
&\mathbb{E}_{\tau\sim p_\tau}\left[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H)\right] - \mathbb{E}_{\hat{\tau}\sim p_{\hat{\tau}}}\left[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}(s_H)\right] && (45)\\
&= J^{\pi^*_H}_{\mathcal{H}} - J^{\hat{\pi}_H}_{\mathcal{H}} && (46)
\end{aligned}$$

Using performance difference lemma we can write:

$$\begin{aligned}
J^{\pi^*_H}_{\mathcal{H}} - J^{\hat{\pi}_H}_{\mathcal{H}} &\leq \sum_{t=1}^{H} \gamma^{t-1} \mathbb{E}_{s\sim d^{\hat{\pi}_H}}\left[V^{\pi^*_H}(s) - Q^{\pi^*_H}(s, \hat{\pi}_H)\right] && (47)\\
&\leq \sum_{t=1}^{H} \gamma^{t-1} \mathbb{E}_{s\sim d^{\hat{\pi}_H}}\left[V^{\pi^*_H}(s) - Q_k(s, \pi^*_H) + Q_k(s, \hat{\pi}_H) - Q^{\pi^*_H}(s, \hat{\pi}_H)\right] && (48)\\
&\leq \sum_{t=1}^{H} \gamma^{t-1} \left(\|Q^{\pi^*_H} - Q_k\|_{1, d^{\hat{\pi}_H}\times\pi^*_H} + \|Q^{\pi^*_H} - Q_k\|_{1, d^{\hat{\pi}_H}\times\hat{\pi}_H}\right) && (49)\\
&\leq \sum_{t=1}^{H} \gamma^{t-1} \left(\|Q^{\pi^*_H} - Q_k\|_{d^{\hat{\pi}_H}\times\pi^*_H} + \|Q^{\pi^*_H} - Q_k\|_{d^{\hat{\pi}_H}\times\hat{\pi}_H}\right) && (50)
\end{aligned}$$

The second line follows from the fact that $Q_k(s, \hat{\pi}_H) \geq Q_k(s, \pi^*_H)$ since $\hat{\pi}_H$ maximizes Q_k.The concentrability assumptions allows us to compare weighted norms under state distribution induced by any policy $\nu(s,a)$ and $d^{\pi_D}(s,a)$ as follows: $\|.\|_\nu \leq \sqrt{\tilde{C}}\|.\|_{d^{\pi_D}}$. We can bound $\|Q^{\pi^*_H} - Q_k\|_{\mu,\pi}$ for arbitrary state distribution μ and policy π as:

$$\begin{aligned}
\|Q^{\pi^*_H} - Q_k\|_{\mu\times\pi} &= \|Q^{\pi^*_H} - \mathcal{T}Q_{k-1} + \mathcal{T}Q_{k-1} - Q_k\| && (51)\\
&\leq \|\mathcal{T}Q^{\pi^*_H} - \mathcal{T}Q_{k-1}\|_{\mu\times\pi} + \|\mathcal{T}Q_{k-1} - Q_k\|_{\mu\times\pi} && (52)\\
&\leq \|\mathcal{T}Q^{\pi^*_H} - \mathcal{T}Q_{k-1}\|_{\mu\times\pi} + \sqrt{\tilde{C}}\|\mathcal{T}Q_{k-1} - Q_k\|_{d^{\pi_D}} && (53)\\
&= \gamma\|Q_{k-1}(\cdot, \pi_{Q_{k-1}}) - Q^{\pi^*_H}(\cdot, \pi^*_H)\|_{P(\mu\times\pi)} + \sqrt{\tilde{C}}\|\mathcal{T}Q_{k-1} - Q_k\|_{d^{\pi_D}} && (54)
\end{aligned}$$

where $P(\mu \times \pi)$ as distribution over $\mathcal{S}$ where $s, a \sim \mu,\ s' \sim p(s,a)$. Define $\pi_{mix} = \text{argmax}_{a\in\mathcal{A}}(Q^{\pi^*_H}(s,a), Q_{k-1}(s,a))$. Then we have:

$$\|Q^{\pi^*_H} - Q_k\|_{\nu,\pi} = \gamma\|Q_{k-1}(.,\pi_{Q_{k-1}}) - Q^{\pi^*_H}(.,\pi^*_H)\|_{P(\mu\times\pi)} + \sqrt{|\mathcal{A}|\tilde{C}}\|\mathcal{T}Q_{k-1} - Q_k\|_{d^{\pi_D}} \tag{55}$$

$$\leq \sqrt{\tilde{C}}\|\mathcal{T}Q_{k-1} - Q_k\|_{d^{\pi_D}} + \gamma\|Q_{k-1} - Q^{\pi^*_H}\|_{P(\mu\times\pi)\times\pi_{mix}} \tag{56}$$

The second term $\|Q_{k-1} - Q^{\pi^*_H}\|_{P(\mu\times\pi)\times\pi_{mix}}$ can be expanded via recursion for k times, since the same analysis holds. We now bound $\|\mathcal{T}Q_{k-1} - Q_k\|_{d^{\pi_D}}$.

$$\|\mathcal{T}Q_{k-1} - Q_k\|^2_{d^{\pi_D}} = L_{d^{\pi_D}}(Q_k, Q_{k-1}) - L_{d^{\pi_D}}(\mathcal{T}Q_{k-1}, Q_{k-1}) \tag{57}$$

$$\leq L_D(Q_k, Q_{k-1}) - L_D(\mathcal{T}Q_{k-1}, Q_{k-1}) + 2\tilde{\epsilon}_m \ \text{ w.p} \geq 1-\delta \tag{58}$$

$$\leq 2\tilde{\epsilon}_m \tag{59}$$

As fitted Q iteration converges $k \to \infty$ for $\gamma < 1$, we have:

$$\|Q^{\pi^*_H} - Q_k\|_{\mu\times\pi} \leq \frac{1-\gamma^k}{1-\gamma}\sqrt{2\tilde{\epsilon}_m\tilde{C}} + \gamma^k\frac{V_{max}}{1-\gamma} \tag{60}$$

In this analysis we obtain $\hat{\pi}_H$ by performing fitted Q iteration ($k \to \infty$) under the dataset D. Therefore our bound for Eqn. 26 from the previous analysis under the current assumptions reduces to:

$$\mathbb{E}_{\tau\sim p_\tau}\left[\sum \gamma^t r(s_t, a_t) + \gamma^H\hat{V}(s_H)\right] - \mathbb{E}_{\hat{\tau}\sim p_{\hat{\tau}}}\left[\sum \gamma^t r(s_t, a_t) + \gamma^H\hat{V}(s_H)\right] \tag{61}$$

$$= J^{\pi^*_H}_{\mathcal{H}} - J^{\hat{\pi}_H}_{\mathcal{H}} \tag{62}$$

$$\leq \frac{2(1-\gamma^H)}{1-\gamma}\left(\frac{1}{1-\gamma}\sqrt{2\tilde{\epsilon}_m\tilde{C}}\right) \tag{63}$$

$$\leq C(\tilde{\epsilon}_m, \tilde{C}, H, \gamma) \tag{64}$$

Plugging this back in our previous analysis we have the following performance bound for H-step lookahead policy:

$$J^{\pi^*} - J^{\pi_{H,\hat{V}}} \leq \frac{2}{1-\gamma^H}[C(\tilde{\epsilon}_m, \tilde{C}, H, \gamma) + \gamma^H\epsilon_v] \tag{65}$$

where $C(\tilde{\epsilon}_m, \tilde{C}, H, \gamma) = \frac{2(1-\gamma^H)}{1-\gamma}\left(\frac{1}{1-\gamma}\sqrt{2\tilde{\epsilon}_m\tilde{C}}\right)$. □

A.4 ARC constrains trajectories close to the parameterized actor

In section 5.1, we use ARC, an iterative importance sampling procedure to solve the constrained optimization in Eqn. 6. The following lemma shows that the final trajectory distribution output as a result of finite importance sampling iteration is bounded in total variation to the trajectory distribution given by the parameterized actor.

Lemma 2. *Let p^τ_{prior} be a distribution over action sequences. Applying M KL-based trust region steps of size ϵ to p^τ_{prior} results in a distribution p^τ_M that satisfies:*

$$D_{TV}(p^\tau_{prior}||p^\tau) \leq T\sqrt{\frac{\epsilon}{2}} \tag{66}$$

Proof. This lemma is adapted from [42] and provided for completeness. Let p^τ_k be the distribution at the k trust region step. $p^\tau_0 = p^\tau_{prior}$ Using Pinsker's inequality we have:

$$D_{KL}(p^\tau_k||p^\tau_{k+1}) \leq \epsilon \tag{67}$$

$$D_{TV}(p^\tau_k||p^\tau_{k+1}) \leq \sqrt{\frac{\epsilon}{2}} \tag{68}$$

Using triangle inequality we have:

$$D_{TV}(p^\tau_{prior}||p^\tau_M) \leq M\sqrt{\frac{\epsilon}{2}} \tag{69}$$

□

B Algorithm Details

B.1 LOOP for online RL

Algorithm 1 LOOP-SAC (for Online RL and Safe RL)

Initialize the parametrized actor π_ϕ, Q-function Q_θ, predictive model $\hat{M}_\psi$, empty replay buffer D. Given planning horizon H.

```
// Training
for $t = 1..(train\_steps)$ do
    Select action given by $a = ARC(s, \pi_\phi)$.        ▷ Use safeARC for safeLOOP
    Execute $a$ in the environment and observe reward $r$ and new state $s'$.
    Store the transition $(s, a, r, s')$ in replay buffer $D$.
    Optimize $\pi_\phi$ and $Q_\theta$ using SAC over replay buffer $D$.
    Train model $\hat{M}_\psi$ on the replay buffer $D$ until convergence every $K_m$ training steps.
end for
// Evaluation
for $t = 1..(eval\_steps)$ do
    Select action given by $a = ARC(s, \pi_\phi)$.
    Execute $a$ in the environment and observe reward $r$ and new state $s'$.
end for
```

B.2 LOOP for offline RL

Algorithm 2 LOOP-offline

Initialize the parametrized actor π_ϕ, Q-function Q_θ, predictive model $\hat{M}_\psi$, empty replay buffer D. Given planning horizon H.

```
// Training
Train model $\hat{M}_\psi$ on the replay buffer $D$ till convergence.
Run an Offline RL algorithm till convergence on $D$ to learn $Q_\theta$ and $\pi_\phi$.
// Evaluation
for $t = 1..(eval\_steps)$ do
    Select action given by $a = ARC(s_t, \pi_\phi)$.
    Execute $a$ in the environment and observe reward $r$ and new state $s'$.
end for
```

B.3 Actor Regularized Control (ARC)

Eqn. 6 gives a general constrained optimization for policy update. In Eqn. 6, with terminal state-action value functions,when $[p^{\tau}_{prior} = \text{Uniform}\ , H = 0]$, we recover the SAC [5] deployment policy, when $[p^{\tau}_{prior} = \pi^{\beta}\ , H = 0]$, we recover the AWAC [45] deployment policy and when $\pi_{prior} = \mathcal{N}(0, \sigma)$, we recover the MPPI [35] deployment policy.

In the LOOP framework we use ARC as our trajectory optimization routine to solve Eqn. 6. Algorithm 3 shows the pseudocode for ARC routine used for Online and Offline RL experiments.

Algorithm 3 Actor Regularized Control (ARC)

```
Input: $s_T,\ \pi_\phi$
Given the parameterized actor $\pi_\phi$, Q-function $Q_\theta$, predictive model $\hat{M}_\psi$, reward model $\hat{r}$, replay buffer $D$, Planning Horizon H, 1-timestep shifted solution from the previous timestep $\mu^{T-1}$, ARC iterations $n_{ARC}$, number of trajectories (population size) $N$.
for $i = 1..n_{ARC}$ do
    $\mathbf{R}_{1:N} = 0$                                    ▷ Rewards of N trajectories
    $\mathbf{A}_{1:N,1:H} = 0$                                ▷ N action sequences with horizon H
    for $j = 1..N$ trajectories do
        // Generate a trajectory with the model
        $s_1 = s_T$
        for $t = 1..H$ horizon do
            // Generate actions from a mixture prior
            $\mathbf{A}_{j,t} = a_t = \beta\pi_\phi(s_t) + (1-\beta)\mathcal{N}(\mu^{T-1}_t, \sigma)$
            $s_{t+1} = \hat{M}_\psi(s_t, a_t)$
        end for
        // Rollout the action sequence $P$ times in each model within the ensemble
        $R = 0$
        for $k = 1..K$models do
            for $p = 1..P$ particles do
                $s_1 = s_T$
                for $t = 1..H$ horizon do
                    $a_t = \mathbf{A}_{j,t}$
                    $s_{t+1} = \hat{M}_\psi(s_t, a_t)$
                    $R = R + \gamma^{t-1}(\mathbb{1}(t = H)Q_\theta(s_t, a_t) + \mathbb{1}(t \neq H)\hat{r}(s_t, a_t))$
                end for
            end for
        end for
        //Uncertainty penalized average reward
        $\mathbf{R}_j = \frac{1}{K}\left(\sum_{k=1}^{K}(R/P) - \beta_{pess}\sum(R/P - \sum_{k=1}^{K}(\frac{R}{KP}))^2\right)$
    end for
    $\mu_{new,1:H}$ = weighted-mean($\mathbf{A}_{1:N}$, weights = exp($\mathbf{R}_{1:N}/\eta$))
    $\Sigma_{new,1:H}$ = weighted-mean(($\mathbf{A}_{1:N} - \mu_{new})^2$, weights = exp($\mathbf{R}_{1:N}/\eta$))
    $\mu^T_{i+1} = \alpha * \mu_{new} + (1-\alpha)\mu^T_i$               ▷ Update mean
    $\Sigma^T_{i+1} = \alpha * \Sigma_{new} + (1-\alpha)\Sigma^T_i$       ▷ Update variance
end for
Output: $\mu^T = \mu^T_{n_{ARC}+1}$
```

β_{pess} is set to zero for Online RL experiments and safe RL experiments where trajectories are scored by unpenalized average. It is tuned for Offline RL experiments as detailed in Appendix C.

B.3.1 ARC for safe-RL

We optimize for the following objective in LOOP for safe RL:

$$\text{argmax}_{a_t} \mathbb{E}_{\hat{M}} \left[R_{H,\hat{V}}(s_t) \right] \text{s.t. } \max_{[K]} \sum_{t=0}^{H} \gamma^t c(s_t, a_T) \leq d_0 \tag{70}$$

where $[K]$ are the model ensembles, c is the constraint cost function and $R_{H,\hat{V}}$ is the H-horizon lookahead objective defined in Eqn. 3. We incorporate safety in the trajectory optimization procedure following previous work [62, 63]. The pseudocode for safeARC used in safeLOOP is shown in Algorithm 4.

B.4 Discussion on the choice of terminal value function

LOOP-SAC, LOOP-SARSA and POLO use different ways to learn a terminal value funcion. LOOP-SARSA is evaluating the "replay buffer policy" instead of the H-step lookahead policy because we are using off-policy data (where the original SARSA is an on-policy algorithm). We believe this is the main reason behind its poor performance. Unfortunately, on-policy LOOP-SARSA would be too slow, due to the need for collecting on-policy data. POLO is formulated to evaluate V^π with the model. However, POLO requires running trajectory optimization during the value function update, which is computationally expensive. In contrast to these methods, LOOP uses an off-policy algorithm to learn V^*. We found that this approach has good performance and it is significantly more computationally efficient than POLO. An interesting direction of future work could be to try to combine LOOP with an efficient off-policy evaluation algorithm to estimate V^π.

Algorithm 4 safeARC

```
Input: $s_T,\ \pi_\phi$
Given the parameterized actor $\pi_\phi$, Q-function $Q_\theta$, predictive model $\hat{M}_\psi$, reward model $\hat{r}$, replay buffer $D$, planning horizon H, 1 timestep shifted solution from the previous timestep $\mu^{T-1}$, safety threshold $d_0$, minimal safe trajectories $m$, ARC iterations $n_{ARC}$, number of trajectories (population size) $N$.
for $i = 1..n_{ARC}$ do
    $\mathbf{R}_{1:N} = 0$                                        ▷ Rewards of N trajectories
    $\mathbf{C}_{1:N} = 0$                                        ▷ Cost of N trajectories
    $\mathbf{A}_{1:N,1:H} = 0$                                    ▷ N action sequences with horizon H
    for $j = 1..N$ trajectories do
        // Generate a trajectory with the model
        $s_1 = s_T$
        for $t = 1..H$ horizon do
            // Generate actions from a mixture prior
            $\mathbf{A}_{j,t} = a_t = \beta\pi_\phi(s_t) + (1-\beta)\mathcal{N}(\mu_t^{T-1}, \sigma)$
            $s_{t+1} = \hat{M}_\psi(s_t, a_t)$
        end for
        // Rollout the action sequence $P$ times in each model within the ensemble
        $R = 0$
        for $k = 1..K$ models do
            for $p = 1..P$ particles do
                $s_1 = s_T$
                for $t = 1..H$ horizon do
                    $a_t = \mathbf{A}_{j,t}$
                    $s_{t+1} = \hat{M}_\psi(s_t, a_t)$
                    $R = R + \gamma^{t-1}(\mathbb{1}(t = H)Q_\theta(s_t, a_t) + \mathbb{1}(t \neq H)\hat{r}(s_t, a_t))$
                    $C = C + \gamma^{t-1}(\hat{c}(s_t, a_t))$
                end for
            end for
        end for
        $\mathbf{R}_j = \frac{1}{K}\sum_{k=1}^{K}(R/P)$               ▷ Average Reward across the ensemble
        $\mathbf{C}_j = \max_{[K]}\max_{[P]}(C)$             ▷ Maximum Cost across the ensemble and particles
    end for
    if count($\mathbf{C}_{1:N} < d_0$) $< m$ then
        $\mu_{new}$ = weighted-mean($\mathbf{A}_{1:N}$, weights = exp($-\mathbf{C}_N/\eta$))
        $\Sigma_{new}$ = weighted-mean(($\mathbf{A}_{1:N} - \mu_{new})^2$, weights = exp($-\mathbf{C}_N/\eta$))
                                                              ▷ Weighted mean w.r.t neg-cost
    else
        safe-idx $= \{i$ for $\mathbf{C}_i < d_0\}$
        $\mu_{new}$= weighted-mean($\mathbf{A}_{\text{safe-idx}}$, weights = exp($\mathbf{R}_{\text{safe-idx}/\eta}$))
        $\Sigma_{new}$ = weighted-mean($\mathbf{A}_{\text{safe-idx}} - \mu_{new})^2$, weights = exp($\mathbf{R}_{\text{safe-idx}}/\eta$))
                                                              ▷ Weighted mean w.r.t safe actions
    end if
    $\mu_{i+1}^T = \alpha * \mu_{new} + (1-\alpha)\mu_i^T$                       ▷ Update mean
    $\Sigma_{i+1}^T = \alpha * \Sigma_{new} + (1-\alpha)\Sigma_i^T$                 ▷ Update variance
end for
Output: $\mu^T = \mu^T_{n_{ARC}+1}$
```

C Experiment Details

We use the same hyperparameters for the underlying off-policy method (SAC) and the ensemble dynamics models following previous work for LOOP and all the baselines [21, 17, 5]. All the results presented are averaged over 5 random seeds.

C.1 Implementation Details for the Dynamics Model Ensemble

Following [21, 17], we use probabilistic ensembles of dynamics models that capture the epistemic uncertainty as well as the aleatoric uncertainty in forward predictions [64]. The dynamics model $\hat{M}$ is comprised of K neural networks. Each individual network is randomly initialized and trained with the same dataset. Using the transition dataset, we train the dynamics model to predict the next state as well as the reward. In practice, instead of directly regressing to the next state, we instead predict Δ_{t+1}, where $\Delta_{t+1} = s_{t+1} - s_t$ parametrized as a Gaussian distribution with a diagonal covariance matrix. We regress directly to the scalar reward.

C.2 Online RL

Additional details on PenGoal-v1 and Claw-v1: We modify the original Pen-v1 environment [2] to have a narrower range of goals given by: $[0.7, 0.7] + \mathcal{N}(0, 0.1)$ and name this environment as PenGoal-v1. We use the Claw-v1 environment from Nagabandi et al. [21] using the original implementation[3] but we find the scale of rewards to be different from the paper.

Baselines: We use the original implementation for MBPO[4]. For SAC, we use a public implementation [5]. We use a planning horizon of 3 for PETS-restricted which is the same as LOOP. LOOP-SARSA is based on the same H-step lookahead idea, but with a terminal value function that is a evaluation of the replay buffer. The value function is updated using the following SARSA update from the replay buffer transitions:

$$\mathcal{T}^{\pi_D} Q(s_t, a_t) = r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) \text{ , where } (s_t, a_t, r_t, s_{t+1}, a_{t+1} \sim D) \qquad (71)$$

This baseline is similar to MBOP [33]. The main difference is that in this case the Q-function is learned via TD-backups for evaluation whereas MBOP uses Monte Carlo Evaluation. For SAC-VE, we implement H-step value expansion from [15] on top of SAC for a fair comparison. This is following the value expansions baseline implemented in MBPO [13].

Training Details: For LOOP-SAC we use SAC [5] as the underlying off-policy RL algorithm. Both the policy network (the parameterized actor) and the Q-function are parameterized by (256, 256) MLP with ReLU activations. The output of the policy network is a tanh squashed Gaussian. We use Adam to optimize both the policy and the Q-network with a learning rate of 3e-4. The temperature for SAC is learned to match a predefined target entropy. The replay buffer has a size of 1e6 and we use a batch size of 256. The target networks are updated with polyak averaging. Dynamics model related hyperparameters are listed in Table 2 and ARC related hyperparameters are in Table 3.

Hyperparameter	Value
Model Update frequency (K_m)	250
Ensemble Size (K)	5
Network Architecture	(200,200,200,200)
Model Learning rate	0.001

Table 2: Dynamics Model Hyperparameters

C.3 Offline RL

Baselines: We reimplement the CRR baseline in Pytorch. For PLAS, we use the original implementation [6]. Note that LOOP requires terminal Q-functions which estimate the cumulative value of future rewards. Some offline RL methods such as CQL will not be suitable to be combined with LOOP because CQL estimates a conservative lower-bound of the Q-function [31]. For MBOP [33], we report the results from their paper.

Training Details: For both LOOP-CRR and LOOP-PLAS we use the provided hyperparameters in the original papers. To optimize for the H-step lookahead objective given in Eqn. 3, we use ARC with

---

[2] https://github.com/vikashplus/mj_envs

[3] https://github.com/google-research/pddm/tree/master/pddm

[4] https://github.com/JannerM/mbpo

[5] https://github.com/openai/spinningup

[6] https://github.com/Wenxuan-Zhou/PLAS

Hyperparamater	**Value**
Planning Horizon (H)	3
Population Size (N)	100
Number of Particles (P)	4
Alpha (α)	0.1
Iterations (n_{ARC})	5
Mixture ratio (β)	0.05
Eta (η)	1

Table 3: Online RL: ARC Hyperparameters

1 iteration of Iterative importance sampling and $\beta = 1$ in the mixture prior. This is done to ensure that ARC trajectories are close to the actor trajectory distribution since the estimated Q-functions are only accurate within the data distribution. For each dataset, We perform an hyperparameter search over horizons h - [2,4,10], pessimism parameter β_{pess} - [0,0.5,1,5], exponential weighting temperature $1/\eta$ - [0.01,0.03,0.1,1,3,10] and noise standard deviation σ- [0.01,0.05,0.1]. We list the hyperparameters from the best experiments in Table 1.

Dataset-type	Environments	LOOP-CRR				LOOP-PLAS			
		h	$1/\eta$	β_{pess}	σ	h	$1/\eta$	β_{pess}	σ
random	hopper	2	10.0	0.5	0.4	4	3	0.5	0.4
	halfcheetah	2	0.01	5.0	0.4	2	1	0	0.01
	walker2d	4	3.0	0.0	0.05	10	10	0	0.05
medium	hopper	2	3.0	1.0	0.01	2	10	0.5	0.01
	halfcheetah	2	0.01	0.0	0.01	2	0.01	1.0	0.01
	walker2d	2	0.1	0.5	0.05	4	3.0	0	0.01
med-replay	hopper	4	0.03	0.5	0.01	2	3.0	1.0	0.01
	halfcheetah	2	1.0	0.5	0.01	2	0.03	5.0	0.01
	walker2d	4	1.0	0.5	0.1	4	0.01	0.5	0.01
med-expert	hopper	4	0.1	1.0	0.05	4	0.1	5.0	0.01
	halfcheetah	2	0.01	5.0	0.01	2	0.01	0.5	0.05
	walker2d	4	0.01	1.0	0.01	2	10.0	1.0	0.01

Table 4: Hyperparameters used in LOOP behavior policy during evaluation for Offline RL methods

C.4 Safe RL

Details on the Environments: For benchmarking safety environments we use the OpenAI safety gym environments [55]. We use a modified observation space for the agents where each agent observes its readings from velocimeter, magnetometer, and gyro sensors, LiDAR observations for the obstacles, and the goal location to a total of a 26-dimensional observation space. We also use an RC-car environment [56] in safe RL experiments shown in Figure 7. RC-car environment has a 6-dimensional observation space consisting of car's position and rate of change of its position. It's action space comprises of throttle and steer command.

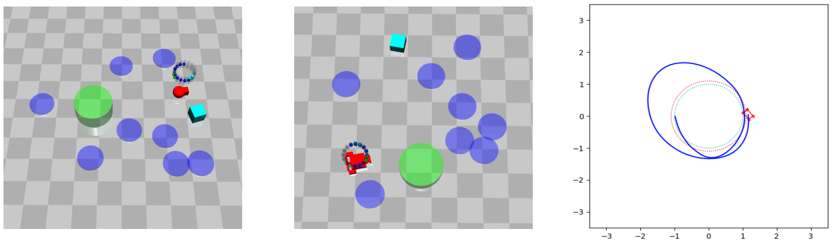

Figure 7: Safety environments. Left to Right: PointGoal1, CarGoal1, Drift-v0

Baselines: We compare against CPO [57], LBPO [58], and PPO-lagrangian [59]. We use the original implementation for LBPO[7] and the safety benchmark [55] for CPO and PPO-lagrangian. All of the three baselines require a threshold to be set in order to optimize for safety. SafeLOOP optimizes for in-horizon safety whereas the baselines optimize for the infinite-step cumulative discounted return, so it becomes difficult to compare the methods directly. We design safeLOOP to optimize for 0 cost within the planning horizon and use the asymptotic safety cost reached by safeLOOP as the threshold for the baselines. We see that safeLOOP can reach average infinite horizon cost less than 10 which is lower than the threshold of 25 used in the official benchmark.

Training details: We use the safeARC algorithm presented in Algorithm 4 to solve the constrained optimization objective in Eqn. 10. The ARC parameters are the same as given in Table 3 with the Iterations(N) changed to 8 and Planning horizon(H) changed to 8. For OpenAI safety environments we use an action repeat of 5 across our method and the baselines.

---

[7] https://github.com/hari-sikchi/LBPO

D Additional Experiments

D.1 Online RL experiments for additional environments

Figure 8 shows the comparison of LOOP-SAC with baselines on additional tasks InvertedPendulum-v2, Swimmer, Hopper-v2 and TruncatedHumanoid-v2. LOOP-SAC is significantly more sample efficient than SAC as we observed in Figure 3. PETS-restricted has poor performance due to planning over a limited horizon. LOOP-SAC outperforms SAC-VE and is competitive to MBPO, except in Humanoid-v2 where MBPO outperforms. LOOP-SARSA has a poor performance across environments.

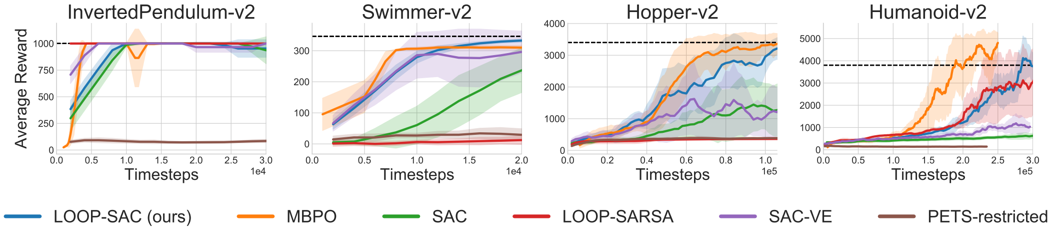

Figure 8: Comparisons of LOOP and the baselines for online RL for InvertedPendulum-v2, Swimmer, Hopper-v2 and TruncatedHumanoid-v2. LOOP-SAC is significantly more sample efficient than SAC. The dashed line indicates the performance of SAC at 1 million timesteps.

D.2 Comparison to modified POLO

In this section, we compare against POLO for Claw-v0 and HalfCheetah-v0 in Figure 9. The author's implementation of POLO is unavailable so we tried our best to implement it. To have a fair comparison with LOOP, we keep the hyperparameters as close to LOOP as possible and use a learned model; since the code for POLO is not available, we are unsure what hyperparameters were used in the original experiments. The performance of POLO is pretty low compared to LOOP, potentially due to the limited computation of CEM used for the value function update. Potentially the performance of POLO would be better with much larger computational resources than what we have available.

We would also like to highlight the difference in computational efficiency in LOOP and POLO. POLO requires an additional trajectory optimization procedure for value function computation, which is very computationally expensive. In contrast, LOOP learns a parameterized policy and value function to make the value function computation significantly faster. In addition, normally for online planning, we can warm start from the results from the previous time step ("amortization"); LOOP and PETS [17] take advantage of this optimization. In contrast, POLO cannot take advantage of this amortization during optimization for the value computation because we sample states IID from the replay buffer. Our implementation of POLO (after reasonable optimizations) takes ≈84 hours for 100k steps of HalfCheetah on a single NVIDIA 1080 GPU whereas LOOP takes ≈7 hours (12x less computation) while taking ≈1/5 the memory consumption of POLO.

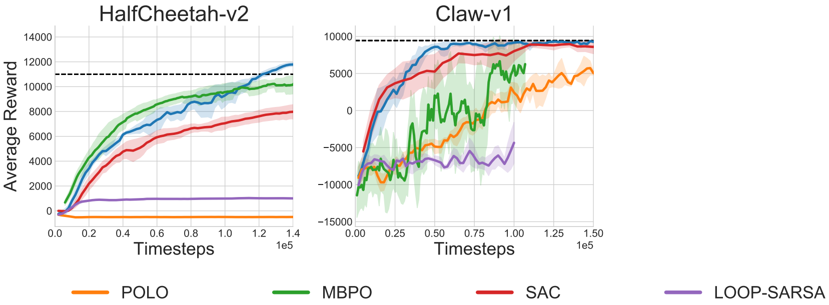

Figure 9: Comparisons of LOOP with modified-POLO for online RL for HalfCheetah-v2 and Claw-v1. POLO demonstrates poor performance which might be attributed to one of the reasons mentioned above.

D.3 Offline RL experiments for D4RL

Table 5 shows the performance of LOOP on four types of D4RL locomotion datasets. The random dataset is generated by a randomly initialized policy. The medium dataset is generated by executing a "medium quality" policy trained up to half of the final performance at convergence. The medium-replay dataset is the replay buffer of the medium quality policy. The medium-expert dataset is generated by a medium quality policy and a fully trained policy.

Dataset	Env	CRR	LOOP CRR	Improve%	PLAS	LOOP PLAS	Improve%	MBOP
random	hopper	10.40	10.68	2.7	10.35	10.71	3.5	**10.8**
	halfcheetah	4.23	7.55	78.5	26.05	**26.14**	0.3	6.3
	walker2d	1.94	2.04	5.2	0.89	2.83	218.0	**8.1**
medium	hopper	65.73	**85.83**	30.6	32.08	56.47	76.0	48.8
	halfcheetah	41.14	41.54	1.0	39.33	39.54	0.5	**44.6**
	walker2d	69.98	**79.18**	13.1	46.20	52.66	14.0	41.0
med-replay	hopper	27.69	29.08	5.0	29.29	**31.29**	6.8	12.4
	halfcheetah	42.29	42.84	1.3	43.96	**44.25**	0.7	42.3
	walker2d	19.84	27.30	37.6	35.59	**41.16**	15.7	9.7
med-expert	hopper	112.02	113.71	1.5	110.95	**114.32**	3.0	55.1
	halfcheetah	21.48	24.19	12.6	93.08	98.16	5.5	**105.9**
	walker2d	103.77	**105.76**	1.9	90.07	99.03	9.9	70.2

Table 5: Normalized scores for LOOP on the D4RL datasets comparing to the underlying offline RL algorithms and a baseline MBOP. LOOP improves the base algorithm across various types of datasets and environments.

D.4 Pessimism ablation for Offline RL

Table 6 shows an ablation of the pessimism term β_{pess} in Eqn. 9 as used in LOOP for Offline RL experiments. We note that the pessimistic term is not itself one of our contributions; this pessimistic term was used in previous works in model-based offline RL like [49, 50] which learn a policy given the data in an uncertainty penalized MDP. We observe that being pessimistic allows us to control incorrect extrapolation and obtain higher returns in most of the environments.

Dataset	Env	LOOP CRR ($\beta = 0$)	LOOP CRR ($\beta = \beta^*$)	β^*	LOOP PLAS ($\beta = 0$)	LOOP PLAS ($\beta = \beta^*$)	β^*
random	hopper	10.31	**10.68**	0.5	10.67	**10.71**	0.5
	halfcheetah	5.12	**7.55**	5.0	**26.14**	26.14	0.0
	walker2d	**2.04**	2.04	0.0	**2.83**	2.83	0.0
medium	hopper	78.56	**85.83**	1.0	54.97	**56.47**	0.5
	halfcheetah	**41.54**	41.54	0.0	38.01	**39.54**	1.0
	walker2d	75.21	**79.18**	0.5	**52.66**	52.66	0.0
med-replay	hopper	28.28	**29.08**	0.5	31.08	**31.29**	1.0
	halfcheetah	42.71	**42.84**	0.5	44.01	**44.25**	5.0
	walker2d	23.17	**27.30**	0.5	32.99	**41.16**	0.5
med-expert	hopper	104.57	**113.71**	1.0	98.87	**114.32**	5.0
	halfcheetah	23.84	**24.19**	5.0	94.19	**98.16**	0.5
	walker2d	104.57	**105.76**	1.0	97.87	**99.03**	1.0

Table 6: Normalized scores for LOOP on the D4RL datasets ablating the pessimism parameter.

D.5 Empirical analysis for ARC

In this section, we aim to verify how the ARC and its specfic hyperparameters affect the performance of LOOP for both the online RL and offline RL settings.

D.5.1 Ablation Study on Actor Regularized Control

In this experiment, we compare the performance of LOOP with ARC to a variant of LOOP which optimize Eqn. 3 without any constraint using CEM in the Online RL setting. CEM starts the optimization from the mean action sequence from the previous environment time step. It does not include actions proposed by the parameterized actor in the population. During training, we measure the actor-divergence defined to be the L_2 distance between the proposed action means of the parameterized actor and the CEM output.

The results are shown in Figure 10. The training process sometimes become unstable in the absence of ARC. We also observe that ARC empirically reduces actor-divergence during training.

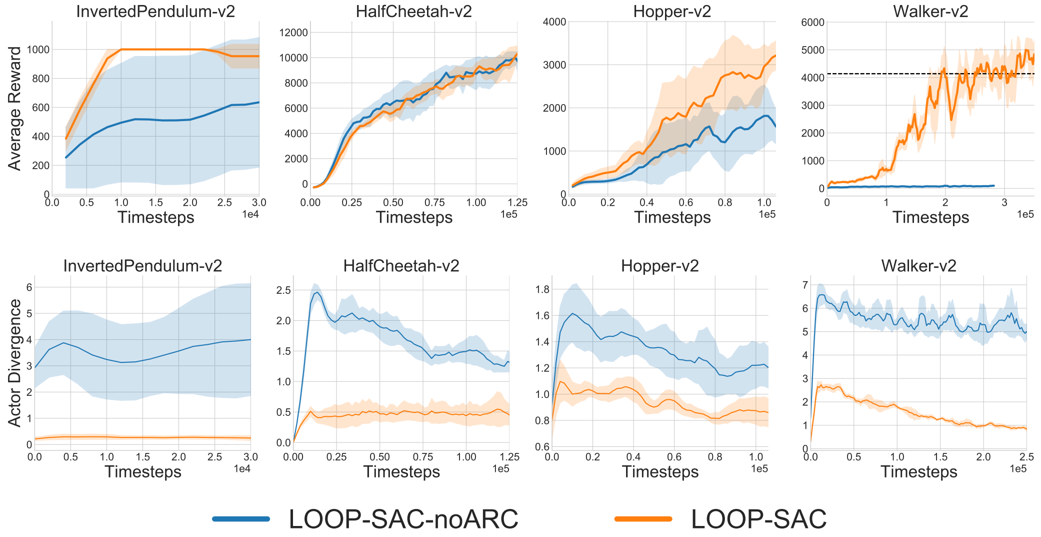

Figure 10: Top: We illustrate the effect of ARC on the performance for online-RL. Without ARC, the performance is worse and the training becomes unstable. Bottom: We illustrate that ARC effectively reduces the actor-divergence between the H-step lookahead policy and the parameterized actor.

D.5.2 ARC runtime

ARC runs at 14.3 Hz for the HalfCheetah-v1 environment with the hyperparameters specified in Table 3 on a machine with Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz and NVIDIA GeForce GTX 1080 Ti with a GPU memory consumption of 1500 MB.

D.5.3 Effect of β in ARC for Online RL

We see in Section 5.1 that ARC uses a mixture distribution of actor and 1-step shifted output from the previous timestep as the prior given by:

$$p^{\tau}_{prior} = \beta\pi_\theta + (1-\beta)\mathcal{N}(\mu_{t-1}, \sigma)$$

where β is the mixture coefficient.

In this experiment, we compare ARC with different parameters of beta for online RL. We observe empirically in Figure 11 that ARC with $\beta < 1$ is more suitable to online RL as it is less restrictive and allows for a greater improvement on the parametric actor.

D.5.4 Effect of β in ARC for in Offline RL

We use ARC with 1 importance sampling iteration for offline RL. In the following experiment, we compare ARC over 1 importance sampling iteration with $\beta = 1$ against ARC over 5 iterations with $\beta = 0.05$. The latter version utilizes the solution obtained by ARC in previous timestep which may potentially select trajectories that lead to out-of-distribution states with overestimated value. We see

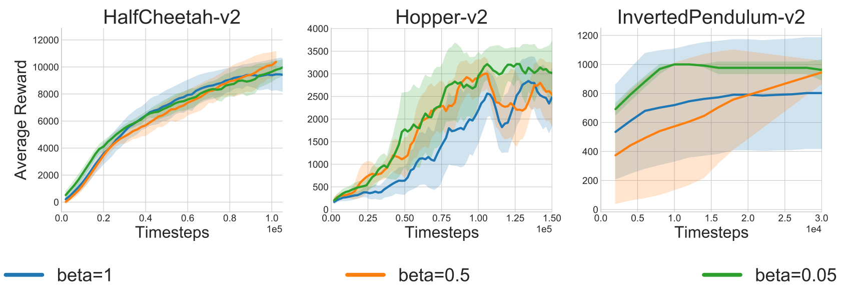

Figure 11: Effect of β in Online RL experiments. A high β constrains output actions to be close to actor and can restrict policy improvement.

in Table 7 that offline RL results for LOOP using ARC with $\beta = 0.05$ has much worse performance than ARC with $\beta = 1$ and also performs worse than the underlying offline RL method.

Dataset	Env	CRR	LOOP-CRR (β=0.05)	LOOP-CRR (β=1.0)	PLAS	LOOP-PLAS (β=0.05)	LOOP-PLAS (β=1.0)
random	hopper	10.40	7.50	10.68	10.35	7.66	10.71
	halfcheetah	4.23	2.32	7.55	26.05	5.21	26.14
	walker2d	1.94	2.43	2.04	0.89	1.24	2.83
medium	hopper	65.73	15.02	85.83	32.08	9.64	56.47
	halfcheetah	41.14	3.09	41.54	39.33	2.99	39.54
	walker2d	69.98	6.02	79.18	46.20	4.25	52.66
med-replay	hopper	27.69	8.78	29.08	29.29	6.8	31.29
	halfcheetah	42.29	3.10	42.84	43.96	4.68	44.25
	walker2d	19.84	6.02	27.30	35.59	6.89	41.16
med-expert	hopper	112.02	8.78	113.71	110.95	7.48	114.32
	halfcheetah	21.48	3.09	24.19	93.08	4.10	98.16
	walker2d	103.77	6.01	105.76	90.07	3.01	99.03

Table 7: Effect of β in offline-LOOP for the offline RL experiments. A low β can potentially select trajectories with overestimated returns.

D.6 Benefits of deploying H-step lookahead in Online RL

In LOOP we use a H-step lookahead policy for both exploration and evaluation. In this experiment, we run the Online RL experiments with LOOP only used for evaluation but not for exploration similar to the Offline RL experiments. This baseline is named SAC-evalLOOP. From Figure 12, LOOP-SAC outperforms both SAC-evalLOOP and SAC, which shows the benefits of H-step lookahead in LOOP during training-time deployment.

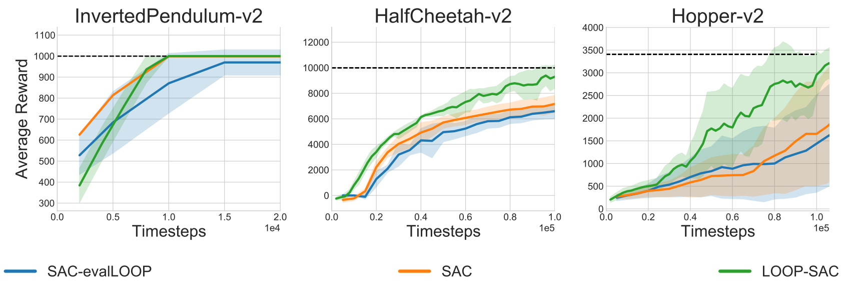

Figure 12: We compare the performance of LOOP-SAC to SAC-Online and SAC-Online-LOOPeval. LOOP-SAC outperforms both baselines and suggests that LOOP benefits from the H-step lookahead policy during training-time deployment.

D.7 Using Offline RL Algorithms with LOOP for Online RL

In Section 5.1, we mentioned that naively combining H-step lookahead policy with an off-policy algorithm will lead to the Actor Divergence issue. One potential solution we have considered besides ARC is to use an Offline RL algorithm as the underlying off-policy algorithm. Offline RL algorithm are designed to train a policy over a static dataset that is not collected by the parameterized actor which in principle should mitigate the instability issues of the value function learning caused by Actor Divergence. Note that in this case we are considering an Online RL *problem setting* with the help of Offline RL *algorithms*.

We investigate a combination of LOOP with Offline RL methods MOPO [49] and CRR [54]. We reimplement MOPO in PyTorch (originally in Tensorflow) for compatibility with other modules of LOOP. We also modify the dynamics model activations from Swish to ReLU. We use the same CRR implementation as the Offline RL experiments discussed above. We adapt MOPO and CRR to the Online RL setting by updating the policy and the value function for 20 gradient updates for each environment timestep. From Figure 13, LOOP-SAC has the most consistent performance across all the environments. LOOP-CRR and LOOP-MOPO work well in some cases but are significantly worse than LOOP-SAC in the others.

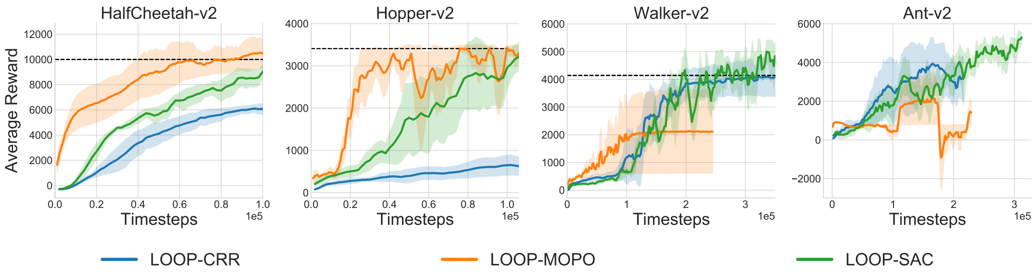

Figure 13: Using offline RL methods like CRR (model-free) or MOPO (model-based) with LOOP does not lead to consistently better performance.

Model Update frequency (K_m)	250
Ensemble Size	5
Network Architecture	MLP with 4 hidden layers of size 200
Model Horizon (H)	3
Model Learning rate	0.001
Policy update per environment step (R)	20
Replay Buffer Size	1e6
Gradient updates per timestep(R)	20
Pessimism parameter(λ)	1
Model rollout length	1

Table 8: LOOP-MOPO Hyperparameters