MODEL-BASED REINFORCEMENT LEARNING WITH ENSEMBLED MODEL-VALUE EXPANSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-based reinforcement learning (MBRL) methods are often more data-efficient and quicker to converge than their model-free counterparts, but typically rely crucially on accurate modeling of the environment dynamics and associated uncertainty in order to perform well. Recent approaches have used ensembles of dynamics models within MBRL to separately capture aleatoric and epistemic uncertainty of the learned dynamics, but many MBRL algorithms are still limited because they treat these dynamics models as a "black box" without fully exploiting the uncertainty modeling. In this paper, we propose a simple but effective approach to improving the performance of MBRL by directly incorporating the ensemble prediction *into* the RL method itself: we propose constructing multiple value roll-outs using different members of the dynamics ensemble, and aggregating the separate estimates to form a joint estimate of the state value. Despite its simplicity, we show that this method substantially improves the performance of MBRL methods: we comprehensively evaluate this technique on common locomotion benchmarks, with ablative experiments to show the added value of our proposed components.

1 INTRODUCTION

Model-based reinforcement learning (MBRL) algorithms are generally both sample-efficient and quicker to converge than model-free alternatives. While there are a variety of model-based approaches, recent work has focused very much on "Dyna-like" MBRL algorithms. These generally work by using the data to learn a dynamics model and then use that model to generate synthetic data for an underlying model-free approach. This added model tends to greatly accelerate early training at the cost of asymptotic performance.

A recent advance in MBRL has been in the design of this dynamics model. The use of a simple dynamics model has been supplanted by an ensemble of stochastic dynamics models, which allows for separate estimation of *aleatoric* and *epistemic* uncertainty. Aleatoric uncertainty reflects the randomness inherent to the underlying environment, and is modeled by each stochastic model. Epistemic uncertainty reflects the uncertainty in modeling the system that could be eliminated with additional data and training. This is reflected in the disagreement between different ensemble members. This idea was popularized in MBRL by Chua et al. (2018) and Janner et al. (2019), among others.

Another important development is the use of the dynamics model to roll out the state-value computation. In this, the state-action-value function is augmented with short-term predictions about the trajectory and rewards received to improve the quality of the estimated value. This technique is more commonly called model-value expansion (MVE), and was first formalized by Feinberg et al. (2018). When MVE is performed using ensembled dynamics models as a black box, the resultant model does not improve results over using a simple, single dynamics model. This suggests that integrating MVE with ensembled dynamics models is only effective if we leverage the ensemble structure instead of treating it as a black box.

In this paper, we present a simple, but very effective, technique to perform MVE while using the ensembled dynamics structure. Our technique unrolls the value function separately along each ensemble element, sampling a joint action at each time. We only unroll a few steps along each separate trajectory estimate, and use that with the value function to produce a set of estimates of the value.

The final value of the state is then the mean of these values (or the minimum), and this value is then used to update the policy. In this way, we can incorporate the epistemic uncertainty of the dynamics model into our MVE estimate. Our technique performs extremely well on a standard set of five locomotion benchmarks. We also present a variety of additional experiments to account for key design decisions and to show that our results are statistically significant.

2 Background and related work

This work focuses on "Dyna-like" MBRL, with separate policy and value functions. This is in contrast to planning-based approaches like PE-TS (Chua et al., 2018), constrained gradient-estimators like ME-TRPO (Kurutach et al., 2018), or others. Wang et al. (2019) provide a taxonomy of modern RL algorithms with benchmarks.

MBRL algorithms generally work by using dynamics models to generate synthetic data, and then using that data to train policy and value models. Environment samples are used to train the dynamics model, in the hope that the dynamics model generalizes well enough to produce useful samples. Recent work by Janner et al. (2019) suggests that this generalization performance is key to the asymptotic performance in MBRL. They derive theoretical bounds on model error as a function of roll-out length, and experimentally show how an ensemble of stochastic environment models can greatly improve training performance. This insight leads to the *Model-based Policy Optimization* (MBPO) algorithm, in which the size of an ensembled dynamics model balances roll-out length to ensure that the the generated data remains plausible. This is further supported by Shen et al. (2020), who use unsupervised domain adaptation techniques to ensure that generated data is plausible and show that this improves policy performance.

Separately, Feinberg et al. (2018) observe that this dynamics model could also be applied to improve value estimates. They formalized the idea of *model-based value expansion* (MVE), in which the value function is trained to be consistent with the learned dynamics and reward. Amos et al. (2020) further develop this idea by using MVE unrolling in the policy improvement step instead of the value update step. This improved construction allows them to train a Stochastic Value Gradient (SVG) controller (Heess et al., 2015) with a single deterministic dynamics model to achieve impressive performance on a range of locomotion benchmarks.

Amos et al. (2020) compare the performance of SAC-SVG with both simple and ensembled dynamics models. They show that, even though a simple dynamics model does not generalize as well as an ensemble model, the policies learned by SAC-SVG perform similarly. In other words, even though ensembled dynamics models are more accurate than simple models, SAC-SVG does not use that additional accuracy to learn better policies. This is in contrast to our approach, which specifically leverages the improved accuracy and epistemic uncertainty modeling offered by ensembled dynamics models to improve training performance.

Our method generates a set of state-action conditional values which we need to reduce to a single value. The use of the minimum function in this role is well-supported in the literature: The most common existing use of Q-ensembles is in the *double Q-function*. Originally proposed by Hasselt (2010), the double Q-function works by taking the minimum of two separately trained Q-functions and effectively mitigates the inherent positive bias in Q-functions. We can extend this to select the minimum over the ensemble. More sophisticated reducing functions have been used in discrete action-spaces. Both Chen et al. (2018); Lee et al. (2021) estimate the upper confidence bound (UCB) of the true Q-value based on the empirical distribution of the Q-functions. This approach provides a natural way to balance exploration and exploitation, but is not directly applicable in the continuous action-spaces that our algorithm targets.

3 Preliminaries and RL Background

We consider the classic discrete-time stochastic RL setting: a Markov decision process (MDP) defined as $(\mathcal{X}, \mathcal{U}, p, r, \rho_0)$. $\mathcal{X}$ and $\mathcal{U}$ are the continuous state and action spaces. $p := \Pr(x'|x, u)$ for some $x, x' \in \mathcal{X}; u \in \mathcal{U}$ is the transition distribution. For notational convenience, we define the transition function $f(x, u)$ to refer to the distribution $p(\cdot|x, u)$. The reward function is $r : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$.

Algorithm 1: MBPO and MBPO-MVE algorithm structure.

Initialize dynamics ensemble $\tilde{f}$, reward model $\tilde{r}$, policy π, and empty buffers for environment transition $\mathcal{D}_{\text{env}}$, and model-generated transitions $\mathcal{D}_{\text{model}}$, and environment $\mathcal{E}$.
Burn-in $\mathcal{D}_{\text{env}}$ using either π or a random policy.
for N epochs **do**
    Train $\tilde{f}$, $\tilde{r}$ to maximize likelihood of $\mathcal{D}_{\text{env}}$
    for E steps **do**
        Advance $\mathcal{E}$ by one step using action from π, add the transition to $\mathcal{D}_{\text{env}}$
        for M model rollouts **do**
            Sample state s uniformly from $\mathcal{D}_{\text{env}}$
            Perform k-step model rollout from s using policy π; add it to $\mathcal{D}_{\text{model}}$
        for G gradient updates **do**
            Update Q function using π and samples from $\mathcal{D}_{\text{model}}$
            if every k^{th} step **then** Update π using Q, and $\mathcal{D}_{\text{model}}$ `// for MBPO`
            if every k^{th} step **then** Update π using $\tilde{f}$, $\tilde{r}$, Q, and $\mathcal{D}_{\text{model}}$ `// for MBPO-MVE`

Our goal is to learn some policy $\pi^* : \mathcal{S} \to \mathcal{U}$ that maximizes the total reward encountered by some agent starting from some position in the state space $x_0 \in \rho_0$.

In this work we build on MBPO which is a model-based extension to the Soft Actor-Critic (SAC) algorithm. Following the structure of SAC, we define a stochastic policy model π (where $\pi(x) := \Pr(u|x)$) and model the expected discounted reward of following policy π from state x as the value function $V : \mathcal{X} \to \mathbb{R}$, and the action-conditional value function $Q : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. These are defined similarly to Watkins (1989):

$$V^\pi(x) := \mathbb{E}[Q(x,u) - \alpha \log \pi(u|x)] \qquad \text{(where } u \text{ is sampled from } \pi(x)) \tag{1}$$

$$Q^\pi(x,u) := r(x,u) + \gamma \mathbb{E}[V(f(x,u))] \tag{2}$$

Where temperature parameter $\alpha > 0$ regularizes the action distribution by penalizing the entropy (Ziebart, 2010), and $\gamma \in [0,1]$ is a discount factor arbitrarily set to $\gamma = 0.99$. We parameterize both Q and π with neural networks.

As detailed in Algorithm 1, Q and π are alternatingly trained on model-generated data. Given a set of transition samples $\{(s,a,r,s'),...\}$ from the model-generated data buffer $\mathcal{D}_{\text{model}}$, we first train Q is to minimize the Bellman residual under the (temporarily fixed) policy: $Q(s,a) - r + \gamma \mathbb{E}[V(s')]$. The policy is then trained to maximize the expected regularized reward under the Q-function: $\mathbb{E}_{\hat{a}\ \pi(\cdot|s)}[Q(s,\hat{a}) - \alpha \log \pi(\hat{a}|s)]$. To close the loop, the policy is then used by MBPO to generate model data for $\mathcal{D}_{\text{model}}$ and to interact with the environment.

To generate data for $\mathcal{D}_{\text{model}}$, MBPO begins with a buffer of data sampled from the environment (called $\mathcal{D}_{\text{env}}$). This data is used to separately train a set of stochastic dynamics functions $\tilde{f}_{(0)}, \tilde{f}_{(1)}, \ldots$ that each predict a distribution of subsequent states reached by taking action a from initial state s. That is, $\tilde{f}_{(j)}(s,a) = \Pr(s'|s,a)$. Starting from states in $\mathcal{D}_{\text{env}}$, MBPO uses the dynamics ensemble and the most recent version of the policy to simulate the evolution of the state over a few steps. These model-generated transitions are then added to $\mathcal{D}_{\text{model}}$, where they will be used to perform the actor- and critic-updates.

3.1 Model-based value expansion

MVE refers to the general class of techniques to improve the value estimate of the critic function by incorporating a dynamics and reward model. From starting state x_0, given horizon $H \in \mathcal{Z}_{\geq}$, we sample a trajectory:

$$((x_0,u_0),(x_1,u_1),...,(x_H,u_H)) \qquad \text{where } u_t = \pi(x_t), \text{ and } x_{t+1} \sim \tilde{f}(x_t,u_t) \tag{3}$$

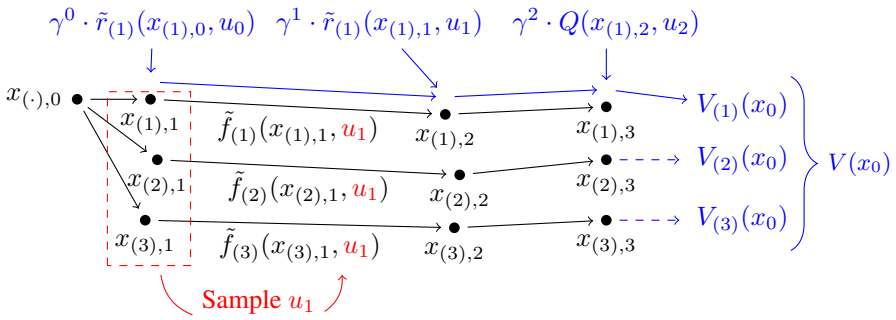

Figure 1: Policy value calculation when $H = 2$. Each ensemble member separately unrolls the policy for H steps for MVE and one final step for the terminal value. The action at step i sampled from the joint distribution $\cup_j \pi(x_{(j)}, i)$. The value estimate by ensemble member j is $\sum_{a=0}^{H-1} \gamma^a \cdot \tilde{r}_{(j)}(x_{(j),a}, u_a) + \gamma^H Q(\pi(x_{(j)}, H))$. The mean (or minimum) of the individual estimates is used as the final value estimate. The exploration temperature is omitted for brevity.

Algorithm 2: MBPO-MVE policy value calculation

Function `ComputeValue` ($\tilde{f}$, π, Q, x, H, γ, α) **:**

`/* separately store running state and value: */`

$x_{(j)} \leftarrow x$ (for each $\tilde{f}_{(j)}$ in $\tilde{f}$)

$V_{(j)} \leftarrow 0$ (for each $\tilde{f}_{(j)}$ in $\tilde{f}$)

for $i \leftarrow 0$ **to** H **do**

  $U_{(j)} \leftarrow \pi(x_{(j)})$ (for each $\tilde{f}_{(j)}$ in $\tilde{f}$)

  `/* pick an ensemble element k and sample an action: */`

  $u \leftarrow$ **sample** $\left(U_{(k)}\right)$

  `/* dynamics model predicts state dist. and reward: */`

  foreach $\tilde{f}_{(j)}$ **in** $\tilde{f}$ **do**

    $X'_{(j)}, r_{(j)} \leftarrow \tilde{f}_{(j)}(x_{(j)}, u)$

    $V_{(j)} \leftarrow V_{(j)} + \begin{cases} \gamma^i \left(r_{(j)} - \alpha \log \pi(u|x_{(k)})\right) & \text{if } i < H \\ \gamma^H \left(Q(x_{(j)}, u) - \alpha \log \pi(u|x_{(k)})\right) & \text{otherwise} \end{cases}$

    $x_{(j)} \leftarrow$ **sample**$(X'_{(j)})$

return $\max(V_{(0)}, V_{(1)}, ...)$

From this, we can compute an estimate for reward that uses the predicted reward for the first H steps and the value estimate of the final step as the tail. Presented without the action entropy penalty:

$$V^{\pi}_{(H)} = \gamma^H Q(x_H, u_H) + \sum_{i=0}^{H-1} \gamma^i r(x_i, u_i) \tag{4}$$

This is equivalent to factoring the value estimate into the short-term value (which is estimated using the reward model) and the residual value (which is estimated using the Q function.) When $H = 0$, this is equivalent to using the Q value without any MVE.

4 MBPO-MVE

Our contribution is MBPO-MVE, an MVE-like extension to MBPO that effectively leverages ensembled dynamics models to accelerate training and improve performance. We retain the design and structure of MBPO, only changing the policy update step. The key insight in our work is that we can make better use of ensembles in value estimation by unrolling the trajectory using each member separately, computing separate Q-values with each, and taking the mean of these values.

As in MBPO, consider a setting with observation space $\mathcal{X} \subseteq \mathbb{R}^n$ and action $\mathcal{U} \subseteq \mathbb{R}^m$. We learn k neural networks that predict the mean and (diagonal) covariance for a Gaussian distribution, forming the dynamics ensemble. The j^{th} model is denoted as: $\tilde{f}_{(j)} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n \times \mathbb{R}^n$. We similarly learn an ensemble of deterministic reward functions: $\tilde{r}_{(j)} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. Dynamics and reward ensemble members are trained to maximize the likelihood of the transitions in $\mathcal{D}_{\text{env}}$. The observations in $\mathcal{D}_{\text{env}}$ are generated by a burn-in policy at the start of training, and subsequently by environment interaction using the learned policy. These models are used to perform many short rollouts to produce imaginary transitions for $\mathcal{D}_{\text{model}}$, as in MBPO. We also have a learned policy $\tilde{\pi} : \mathcal{X} \to \mathcal{U}$ and Q function $\tilde{Q} : \mathcal{X} \times U \to \mathbb{R}$, both parameterized by neural networks.

Departing from MBPO, our method also uses the dynamics model to improve the value computation in the policy update step. In the policy update step, given a state s, we wish to update the policy π to improve $Q(s, \pi(s))$, the expected value of following the policy from that state. We use the ensemble of k dynamics models to compute k different possible H-step unrollings. To obtain the j^{th} trajectory, we begin from the starting state $x_{(j),0} := s$. To obtain the successor state $x_{(j),i+1}$ from state $x_{(j),i}$, we begin by sampling a single action from the joint distribution of actions proposed by $\tilde{\pi}$ across all state estimates: $u_i \leftarrow \text{SAMPLE}\left(\cup_j \pi(x_{(j),i})\right)$. We then use the j^{th} ensemble member to estimate the successor state: $x_{(j),i+1} := \tilde{f}_{(j)}(x_{(j),i}, u_i)$. Performing this step H times gives us the sequence $x_{(j),0}, x_{(j),1}, \ldots, x_{(j),H-1}, x_{(j),H}$. From this sequence, we can obtain $V^{\pi}_{(j)}(s)$, the j^{th} estimate of the value of following π from state s:

$$V^{\pi}_{(j)}(s) = \underbrace{\gamma^0 \tilde{r}(x_{(j),0}, u_0) + \ldots + \gamma^{H-1} \tilde{r}(x_{(j),H-1}, u_{H-1})}_{H\text{-step unrolling}} + \underbrace{\gamma^H \tilde{Q}(x_H, u_H)}_{\text{terminal value estimate}} \tag{5}$$

$$= \sum_{i=0}^{H-1} \gamma^i \tilde{r}(x_{(j),i}, u_i) + \gamma^H \tilde{Q}(x_{(j),H}, u_H) \tag{6}$$

Now that we have obtained k separate estimates of the value of following π from s, we can combine these into a single value estimate by taking the mean (or minimum) value. The final value function used by MBPO-MVE in the policy update step is therefore:

$$V^{\pi}(s) = \frac{1}{k} \sum_{j=0}^{k-1} \left[\gamma^H \tilde{Q}(x_{(j),H}, u_H) + \sum_{i=0}^{H-1} \gamma^i r(x_{(j),i}, u_i) \right] \tag{7}$$

Where $x_{(j),i}$ is the i^{th} unrolling step of the j^{th} ensemble. The policy is altered to maximize this value.

4.1 MVE Horizon

Our method introduces one additional hyperparameter: the number of time-steps over which to perform the MVE expansion. The best value of this parameter appears to be dependent on the environment without any clear heuristic or rule to suggest how to select it. In the paper introducing MVE, Feinberg et al. (2018) conjecture that there are two conflicting pressures that affect the optimal value. Longer-horizon rollouts may, under ideal conditions, improve the potential target error by $\mathcal{O}(\gamma^H)$. The potential gains, however, are noted by Feinberg et al. (2018) to be very sensitive to both error in dynamics and non-uniform bias in Q-functions. These problems are mitigated by shorter-horizon rollouts, which reduce compounding inaccuracy from imperfect dynamics models.

Different environments have different optimum rollout lengths, as empirically determined by Amos et al. (2020). They evaluate the performance of SAC-SVG for $H \in \{1, 2, 3, 4, 5, 10\}$, and from their data we observe that different environments have different optimal rollout lengths, though these values tend to be small $(2 - 4)$. For our primary experiments, we simply select the rollout length that produces the best mean results in SAC-SVG, on the assumption that the factors affecting the best value are similar. We also conduct a small experiment to validate this.

	Ant-v2*	Hopper-v2	Swimmer-v2	HalfCheetah-v2	Walker2d-v2
This paper					
MBPO-MVE (mean)	2542±2138	**3450±338**	124.0± 23.3	**11265±1007**	**4254±2027**
MBPO-MVE (min)	**5241±1029**	3230±833	102.5± 66.8	**11626±1102**	**3809±2416**
MBPO† (Janner et al., 2019)	3820±2221	2000±903	96.7± 32.9	8469±1304	**4174± 626**
Dyna-like					
SUNRISE‡ (Lee et al., 2021)	1502±484	2643±472		4502± 444	1237±1124
SAC-SVG (best) § (Amos et al., 2020)	4473±893	2852±362	**350.2± 3.6**	9220±1432	1852± 968
ME-TRPO¶ (Kurutach et al., 2018)	282± 18.0	1273±501	225.5±105	2284± 900	-1609± 658
Shooting/Planning Methods					
PETS-RS¶ (Chua et al., 2018)	1166±227	115±621	326.2± 12.6	2288±1019	283± 502
POPLIN-P¶ (Wang & Ba, 2020)	2330±321	2055±614	334.4± 34.2	4235±1133	597± 479
Model-Free RL					
SAC§ (Haarnoja et al., 2018)	511± 76.4	2180±977	**351.2± 5.3**	6515±1101	1265±1317
TD3¶ (Fujimoto et al., 2018)	871±284	1817±995	72.1±131	3016± 970	-516± 812

Table 1: We report the mean and standard deviation of rewards across a range of OpenAI Gym locomotion tasks. Values are rounded, and empty cells indicate no reported data. All results that fall within the 5^{th} percentile lower confidence bound of the highest mean result are marked in bold.

5 Results and Discussion

The algorithm is evaluated on five common locomotion tasks in the OpenAI Gym environment (Brockman et al., 2016), simulated by MuJoCo (Todorov et al., 2012). Results are presented in Table 1, categorized by approach. MBPO-MVE consistently outperforms popular MBRL and MFRL algorithms benchmarked by Wang et al. (2019) as well as more recent work. In particular, by bringing both MVE and ensemble-based models together, we can show improvement over the best-performing contemporary MVE work SAC-SVG (Amos et al., 2020), and over ensemble-based MBRL work (like SUNRISE (Lee et al., 2021), and vanilla MBPO (Janner et al., 2019)).

We present our method using the mean of ensemble estimates (mean), and one that uses the minimum (min). Either method consistently performs well on most tasks, but the largest difference in performance is on the `Ant-v2` task, where the much more conservative estimates of MBPO-MVE (min) yield a much higher score. Our method improves on the underlying MBPO performance on `Swimmer-v2`, but despite that is unable to match the performance of SAC or SAC-SVG on that task. It is likely that additional hyperparameter tuning would lead to better results on that particular task, both for MBPO and for our MBPO-MVE.

To show that our method statistically produces higher rewards than the baseline MBPO implementation, we perform individual one-tailed t-tests and combine them with Fisher's method. With these, we are able to accept the alternative hypothesis that MBPO-MVE (mean) produces higher rewards than vanilla MBPO at the 5% significance level. For full details, see Appendix B. Note that the hyperparameters we using for both MBPO-MVE and MBPO were tuned by the MBRL-lib team (Pineda et al., 2021) for the MBPO baseline, so this result cannot be attributed to additional hyperparameter tuning.

We investigate three key properties of our method: use of a single joint action over all ensembles, the choice of reducing function, and the choice of MVE horizon.

5.1 Joint action does not reduce performance

We compare the performance of our method to a similar one where each ensemble element separately samples an action. We wish to show that using a joint action does not reduce performance. We conduct a statistical test as before, obtaining the results in Table 2. Detailed in Appendix B,

*Modified to exclude external forces in the observation. See Pineda et al. (2021) for details.

†For comparison, we include the performance of our method with $H = 0$. This is equivalent to the MBPO implementation by Pineda et al. (2021).

‡Reported by Lee et al. (2021)

§Reported by Amos et al. (2020).

¶From benchmarks by Wang et al. (2019)

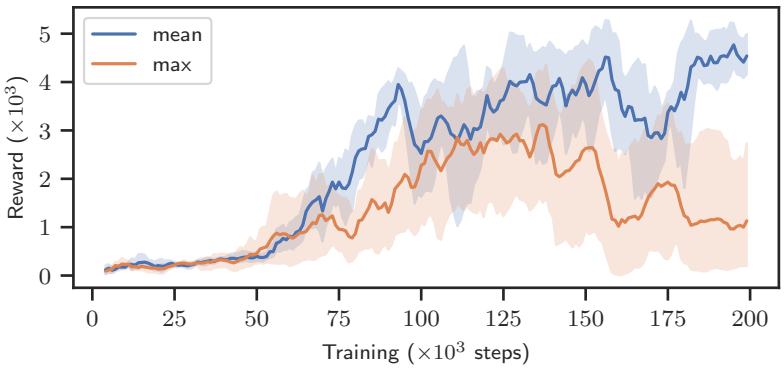

Figure 2: Training collapse with `max`, but not with `mean`. This is demonstrated by plotting the reward obtained from both mean and max when training a model on `Walker2d-v2`. The error bars indicates the range of observed values over five seeds. While both mean and max learn at the start, the training of max falls to a small value.

the test fails to reject the null hypothesis (at the $\alpha = 5\%$ significance) that using a joint action and separate actions has the same mean reward over all environments.

5.2 Choice of Reducing Function

The performance of our method is very sensitive to the selection of the reducing function. We evaluated our method on five environments using the minimum, mean, and maximum of the ensemble Q-values. To mimic the UCB work of Chen et al. (2018); Lee et al. (2021), we also test `topk`, which is the mean of the top five of the seven estimates. The results are presented in Table 3.

Looking deeper into the reward obtained during training, we see that some reducing functions are prone to a form of training collapse in which the returns of a policy model suddenly falls to zero during training. This is most apparent in the evaluation of `Walker2d-v2`, shown in Figure 2. This suggests there are two different effects at play: one effect from the use of ensemble unrolling that improves training performance over most reducing functions, and a second effect that is more likely to cause training collapse on some functions. This second effect could potentially be lessened through hyperparameter optimization, and more study is needed to understand this.

5.3 MVE Horizon

From the experiments of Amos et al. (2020), the MVE horizon is a key element determining the performance of the model. Here we test this hypothesis on `HalfCheetah-v2` by comparing performance over different roll-out lengths, as shown in Table 4. We qualitatively observe that both MBPO-MVE and SAC-SVG achieve the highest mean and smallest variance when $H = 3$, and the score distributions remain high until $H = 5$. At $H = 10$, both methods exhibit poor performance.

There is as yet no theory accounting for the variation between environments. The optimal rollout length is generally assumed to be the result of a trade-off between two effects: compounding dynamics inaccuracy that promotes shorter rollouts, and target error improvement that promotes longer rollouts. In the latter case, Feinberg et al. (2018) show that – under ideal conditions – the potential

	`Ant-v2`	`Hopper-v2`	`Swimmer-v2`	`HalfCheetah-v2`	`Walker2d-v2`
MBPO-MVE (mean)	2542±2138	3450±338	124.0±23.3	11265±1007	4254±2027
Without joint action	3060±3125	2784±1024	113.1±44.7	11870±1376	3363±2696

Table 2: We compare the performance of our method with and without the joint action. Combining individual t-tests with Fisher's method, we cannot conclude that the use of separate actions produces significant improvement at the $\alpha = 5\%$ level. Details in Appendix B.

MBPO-MVE	Ant-v2	Hopper-v2	Swimmer-v2	HalfCheetah-v2	Walker2d-v2
mean	2542±2138	**3450±338**	**124.0± 23.3**	**11265±1007**	**4254±2027**
min	**5241±1029**	3230±833	102.5± 66.8	**11626±1102**	3809±2416
max	837±1754	2728±883	88.1±48.8	8659±5096	3308±1555
top-k	1569±1274	2926±463	97.4±67.1	10394±4445	**4493±2377**

Table 3: Performance over different reducing functions on various environments.

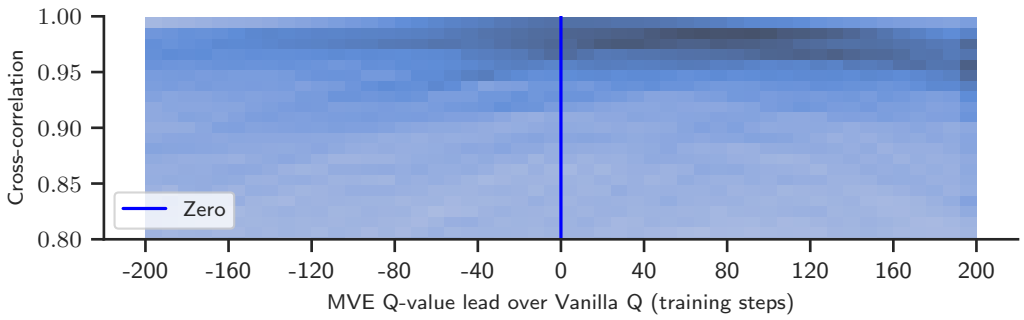

Figure 3: We compute the Pearson correlation coefficient between Ensemble Q-values and vanilla Q-values for a fixed set of states every other training step for a subset of the training steps. The Ensemble Q-values lead the vanilla Q-values, as shown by the rightward skew. This suggests that the MVE construction counteracts some delay, perhaps introduced by the slow target network.

target error may be improved by $\mathcal{O}(\gamma^H)$. Unfortunately, this is very sensitive to both model error and non-uniform bias in Q-functions.

6 Why does MVE work?

MVE naturally follows from the observation that both the Q-value estimates and the dynamics model learn information about the true system dynamics. In fact, in the original paper, Feinberg et al. (2018) use MVE to update the Q-function to be consistent with the system dynamics. (In contrast, our work and SAC-SVG both update the policy from the Q-function using MVE.) The original justification for MVE is that there is a potential $\mathcal{O}(\gamma^H)$ reduction in target error, but this is only realized if the model error is very low and the bias in the Q-function is uniform. The observed benefits of MVE seem to be much larger than this would suggest, and so we propose an alternative hypothesis: the use of MVE mitigates the training lag introduced by the use of the target Q-function.

Introduced by Mnih et al. (2015), many MBRL methods use a slowly-updating second "target" Q-function in addition to the regular Q-function. By splitting the Q-function into slowly- and quickly-learning components, we can prevent runaway reinforcement. Since states visited in sequence tend to be close together in space, it is possible for an increase in the current state-value to inadvertently increase the value of future states in the same trajectory. This process, if uncorrected, introduces a spurious bias in the value function that may never be unlearned. While the use of a separate target function effectively mitigates the runaway reinforcement problem, it also slows down training considerably.

	$H=1$	$H=2$	$\boldsymbol{H=3}$	$H=4$	$H=5$	$H=10$
MBPO-MVE (mean)	10305±1260	10820±2020	11736± 849	10923±2860	7216±6740	1236±3480
SAC-SVG	6890±1860	8752±1786	9220±1431	8175±3226	6129±3520	2477±2596

Table 4: Performance over different unrolling horizons on HalfCheetah-v2. $H=3$ is reported in the main results. $H=0$ (not shown here) would be equivalent to MBPO without MVE.

Our MVE uses a reward and dynamics model in addition to the Q-function network. Since the reward and dynamics models are trained directly from data, and not from previous Q-values, they are not subject to this runaway reinforcement effect. Our hypothesis is that this is the primary mechanism by which MVE accelerates training. To show that this explanation is plausible, we compute the time-lagged Pearson correlation coefficient between ensemble and vanilla Q-values for a set of fixed states known to be in the distribution of states visited by the algorithm. Figure 3 shows that the correlation distribution is skewed right, indicating that MVE Q-values lead vanilla Q-values. Experiment details are in Appendix C. This is by no means a conclusive result, and more investigation is needed to exclude competing possibilities. Such hypotheses include that MVE corrects a bias induced by Q-function structure, or that MVE provides a form of exploration incentive comparable to upper-confidence-bound-style exploration.

Alternative approaches to address the target network's shortcomings have been proposed in the literature. One option is to replace the target network with a smoothed Q-function, like DeepMellow (Kim et al., 2019), which is designed to be non-expansive. Another approach is to use neural networks that behave differently in the short- and long-term, like Two-Timescale Networks (Chung et al., 2019). Two timescale networks split the network into segments that learn at different rates, which appears to mitigate this problem. This is a promising area for future research, especially for designs that can incorporate the dynamics model into the Q-value estimation.

7 Conclusion

We have presented a novel method of combining ensembles of dynamics models with model-value expansion and provided a comprehensive evaluation to justify our design. Our method produces significant improvements with few additional hyperparameters, and can be easily adapted to other MBRL algorithms. There are many opportunities for future work in this area:

The locomotion benchmarks most continuous-state and -action RL work focuses on respond well to short-horizon exploration techniques like dithering, α-greedy policies, or an adaptive action-distribution temperature. In the discrete RL world, more sophisticated techniques are necessary to achieve long-horizon exploration. Our use of a dynamics ensemble allows us to measure the epistemic uncertainty associated with each state, which provides a natural extension to the work of Burda et al. (2019); Kurutach et al. (2018); Chen et al. (2018). In these approaches, the authors use random-network distillation or ensemble-model approaches to estimate how closely a state has been explored so they can manage exploration in more complex tasks with sparse rewards. We could potentially apply our MVE work to these areas.

As yet, questions remain about how MVE works. While we provide an account of how it might work with target Q-networks, we do not yet account for MVE without target networks. Further study is needed to see if MVE has any benefit without a target network, and if so, how MVE works in that case. Better understanding this will allow us to design better value functions that potentially elegantly include dynamics models into the value estimation.

Finally, it is not entirely clear that the presence of a target network is a necessary component to Dyna-like RL. Some prior work has shown that is is possible to remove the target network (Kim et al., 2019; Chung et al., 2019), but these approaches may be restricted to discrete action domains or are otherwise not as powerful as MBPO. It may be possible to provide a similar effect using MVE, which is an area for future research.

Reproducibility statement. We modify the baseline MBPO implementation in MBRL-lib by Pineda et al. (2021), and use their original hyperparameters without further tuning. The unrolling horizon H is selected to match the most effective horizon in SAC-SVG (Amos et al., 2020) for each task. Evaluations of MBPO-MVE (mean) with and without joint action are repeated over ten seeds, and all other evaluations are repeated over five seeds. The mean and standard deviation of the rewards obtained by the last version of the agent at the end of training is reported. Hyperparameters are documented in Appendix A.

Source code is attached as supplementary material, and will be available online at `https://www.github.com/[OMITTED]`.

REFERENCES

Brandon Amos, Samuel Stanton, Denis Yarats, and Andrew Gordon Wilson. On the model-based stochastic value gradient for continuous reinforcement learning. *CoRR*, abs/2008.12775, 2020. URL https://arxiv.org/abs/2008.12775.

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

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Richard Y. Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. UCB Exploration via q-ensembles, 2018. URL https://openreview.net/forum?id=H1cKvl-Rb.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf.

Wesley Chung, Somjit Nath, Ajin Joseph, and Martha White. Two-timescale networks for nonlinear value function approximation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJleN20qK7.

Vladimir Feinberg, Alvin Wan, I. Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *ArXiv*, abs/1803.00101, 2018.

R.A. Fisher. *Statistical methods for research workers*. Edinburgh Oliver & Boyd, 1925.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1582–1591, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *ICML*, 2018.

Hado Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

Nicolas Heess, Greg Wayne, David Silver, Timothy P. Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients. *CoRR*, abs/1510.09142, 2015. URL http://arxiv.org/abs/1510.09142.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

Seungchan Kim, Kavosh Asadi, Michael Littman, and George Konidaris. Deepmellow: Removing the need for a target network in deep q-learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 2733–2739. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/379. URL https://doi.org/10.24963/ijcai.2019/379.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJJinbWRZ.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 6131–6141. PMLR, 2021.

Scott E Maxwell, Harold D Delaney, and Ken Kelley. Designing experiments and analyzing data: a model comparison perspective. 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL https://arxiv.org/abs/2104.10159.

Jian Shen, Han Zhao, Weinan Zhang, and Yong Yu. Model-based policy optimization with unsupervised model adaptation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2823–2834. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1dc3a89d0d440ba31729b0ba74b93a33-Paper.pdf.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pp. 5026–5033. IEEE, 2012. ISBN 978-1-4673-1737-5. URL http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1exf64KwH.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning, 2019.

Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989. URL http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy, 2010.

A HYPERPARAMETERS AND RESULTS

Each experiment was run across five seeds with the mean and standard deviation reported. Each evaluation was performed on machines with commodity CPUs and NVIDIA 1080 Ti or 2080 Ti GPUs, and takes no more than two days to run on our hardware.

Most hyperparameters were used as-is from previous work by others. MBPO-related hyperparameters were directly used from the MBRL-lib implementation from Facebook research (Pineda et al., 2021), only reducing the number of training steps to match benchmarks by Wang et al. (2019). In particular, we inherit the ensemble size (7) and the value of $k = 5$ for top-k evaluation. The rollout horizon H was selected to match the horizon with the highest mean performance in Amos et al. (2020).

The summary of key hyperparameters is:

Environment	Environment Samples	Horizon H
`Ant-v2`	200,000	2
`Hopper-v2`	125,000	2
`Swimmer-v2`	200,000	2
`HalfCheetah-v2`	200,000	3
`Walker2d-v2`	200,000	4

Results are reported as the mean and standard deviation of samples over five seeds at the end of training. We report this for each of three reducing functions.

Reporting Standards We note that the benchmarking work by Wang et al. (2019) run all environments (including `Hopper-v2`) 200,000 timesteps and report the distribution of results over 4 random seeds and 5000 timesteps. Amos et al. (2020) runs `Swimmer-v2` for only 50,000 steps and reports the performance at the end of training over 10 seeds and a range of horizon values.

B STATISTICAL TESTS FOR RESULT SIGNIFICANCE

In the main results (Section 5), we show the difference in performance between our method with and without model-value expansion. The table shows there are differences in performance, but it is not immediately obvious that our method is significantly better across all environments. Here we describe the aggregate t-test we perform to show that using MVE achieves a higher mean reward at the $\alpha = 5\%$ significance level. (The significance level is chosen to reflect the expense of conducting experiments at scale.)

We begin by setting the null hypothesis that both methods produce the same mean score over all environments, and the alternative hypothesis that MBPO-MVE has a higher mean score in all environments. Additionally, we assume that:

1. the reward distributions between environments are uncorrelated once conditioned on the use of MBPO or MBPO-MVE. ("Independence" assumption.)

2. the distribution of reward achieved by each method has the same variance for each environment, but not between environments. ("Identical distribution" assumption.)

To effectively combine these, we perform a one-tailed Student's t-test within each environment, and find the significance of the aggregate results using Fisher's method. These are standard statistical methods, and the textbook by Maxwell et al. (2017) derives these methods and explains their use.

We begin by performing a Student's t-test on each environment separately. We choose the t-test over the z-test as we have a small sample size (5 per method) within each environment. Here are the individual t-statistics and p-values:

Environment	t-statistic	p-value
`Ant-v2`	-1.078	0.150
`Hopper-v2`	4.611	1.000
`Swimmer-v2`	1.889	0.959
`HalfCheetah-v2`	4.611	1.000
`Walker2d-v2`	0.085	0.467

We then use Fisher's method to combine the probabilities of observing the t-statistic while controlling for Type-I errors. We obtain a single χ^2 test statistic using:

$$\chi^2_{2k} \sim -2\sum_{i=1}^{k} \ln p_i, \text{where } k = 5 \text{ is the number of environments}$$

We obtain the test statistic $\chi^2_{10}(44.99)$, and a cumulative p-value of about $2 \times 10^{-5} < \alpha = 5\%$.

Note that we cannot use ANOVA (Fisher, 1925) for two important reasons: (1) that the variance of each comparison pair is different, and (2) because multivariate-ANOVA methods would test if MBPO-MVE performs significantly better in any environment, not over all environments.

B.1 Joint action is not worse

In section 5.1, we perform the same test to distinguish between the results when the action at each step is shared ("joint action"), or when each ensemble samples a separate action ("no joint action"). We begin by setting the null hypothesis that both methods produce the same mean score, and the alternative hypothesis that the no-joint-action method agents have a higher mean score than joint-action methods. We obtain these individual t-statistics from data in Table 2:

Environment	t-statistic	p-value
`Ant-v2`	0.433	0.665
`Hopper-v2`	-1.954	0.003
`Swimmer-v2`	-0.698	0.247
`HalfCheetah-v2`	0.424	0.659
`Walker2d-v2`	-0.835	0.207

Applying Fisher's method we obtain the test statistic $\chi^2_{10}(5.43)$, which yields a cumulative p-value of about $86.0\% > \alpha = 5\%$. We fail to reject the null hypothesis that using a joint action produces agents with similar mean rewards to using separate actions.

C Ensemble Q leads Vanilla Q

To obtain Figure 3, we randomly sample 2048 visited states from one run of our algorithm on `Hopper-v2`, and then compute the predicted ensemble and vanilla Q-values every two training steps during a second run with the same random seed. The cross-correlation chart is built from the Q-values logged between 9,000 and 10,000 training steps, out of a total training horizon of 125,000 steps. The distribution of cross-correlations skews right: the modal lag that maximizes cross-entropy is at about 46 training steps.