DIMINISHING RETURN OF VALUE EXPANSION METHODS IN MODEL-BASED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-based reinforcement learning is an approach to increase sample efficiency. However, the accuracy of the dynamics models and the resulting compounding error over trajectories are commonly regarded as a limitation of model-based approaches. A natural question to ask is: How much more sample efficiency can be gained by improving the learned dynamics models? Specifically, this paper addresses the value expansion class of model-based approaches. Our empirical study shows that expanding the value function for the critic or actor update increases sample efficiency, but the gain in improvement decreases with each added expansion step. Therefore, longer horizons yield diminishing returns in terms of sample efficiency. In an extensive experimental comparison that uses the oracle dynamics model to avoid compounding model error, we show that short horizons are sufficient to obtain the lowest sample complexity for the given tasks. For long horizons, the improvements are marginal or can even decrease learning performance despite using the oracle dynamics model. Model-free counterparts, which use off-policy trajectories from a replay buffer and introduce no computational overhead, often show on-par performance and pose as a strong baseline. Finally, as we observe the same issues with both oracle and learned models, we conclude that the limitation of model-based value expansion methods is not so much the model accuracy of the learned models.

1 INTRODUCTION

Sample efficiency is a central issue that prevents reinforcement learning (RL) agents from learning in physical environments. Especially in applications like robotics, samples one the real systems are particularly scarce and expensive to acquire due to the high cost of operating robots. A technique that has proven to substantially enhance sample efficiency is model-based reinforcement learning (Deisenroth et al., 2013) In model-based RL, a model of the system dynamics is usually learned from data, which is then used for planning (Chua et al., 2018; Hafner et al., 2019) or for policy learning (Sutton, 1990; Janner et al., 2019).

Over the years, the model-based RL community has identified several ways of applying (learned) dynamics models in the RL framework. Sutton's (1990) DynaQ uses the model for data augmentation, where model rollouts are started from real environment states coming from a replay buffer. The collected data is afterwards given to a model-free RL agent. Recently, various improvements have been proposed to the original DynaQ algorithm, such as using ensemble neural network models and short rollout horizons (Janner et al., 2019; Lai et al., 2020), and improving the synthetic data generation with model predictive control (Morgan et al., 2021). Feinberg et al. (2018) proposed an alternative way of incorporating a dynamics model into the RL framework. Their model-based value expansion (MVE) algorithm unrolls the dynamics model and discounts along the modelled trajectory to approximate better targets for value function learning. Subsequent works were primarily concerned with adaptively setting the rollout horizon based on some (indirect) measure of the modelling error, e.g., model uncertainty (Buckman et al., 2018; Abbas et al., 2020), a reconstruction loss (Wang et al., 2020) or approximating the local model error through temporal difference learning (Xiao et al., 2019). Lastly, using backpropagation through time (Werbos, 1990), dynamics models have been applied to the policy improvement step to assist in computing better policy gradients. Deisenroth & Rasmussen (2011) uses Gaussian processes (Rasmussen & Williams, 2006) and moment matching to find a closed form solution for the gradient of the trajectory loss function.

Others leverage the model's differentiability to algorithmically compute policy gradients. Stochastic value gradients (SVG) (Heess et al., 2015), uses a model in combination with the reparametrization trick Kingma & Welling (2014) to propagate gradients along real environment trajectories. Others differentiate through model trajectories (Byravan et al., 2020; Amos et al., 2021). On a more abstract level, MVE-type and SVG-type algorithms are very similar. They both learn a dynamics model and use it for H-step trajectories (where H is the number of modelled time steps) to better approximate the quantity of interest – the next state Q-function in the case of SVG and the Q-function for MVE. These value expansion methods all assume that longer rollout horizons will improve learning if the model is sufficiently accurate.

The common opinion is that learning models with less prediction error may further improve the sample efficiency of RL. This argument is based on the fact that the single-step approximation error can become a substantial problem when using learned dynamics models. Minor modelling errors can accumulate quickly along the trajectory when multi-step trajectories are built by bootstrapping successive model predictions. This problem is known as the *compounding model error*. Furthermore, most bounds in model-based RL are usually dependent on the model error (Feinberg et al., 2018), and improvement guarantees assume model errors converging to zero. In practice, rollout horizons are often kept short to avoid significant compounding model error build-up (Janner et al., 2019). Intuitively, using longer model horizons has the potential of exploiting the benefits of the model even more. For this reason, immense research efforts have been put into building and learning better dynamics models. We can differentiate between purely engineered (white-box) models and learned (black-box) models (Nguyen-Tuong & Peters, 2011). White-box models offer many advantages over black-box models, as they are more interpretable and, thus, more predictable. However, most real-world robotics problems are too complex to model analytically, and one has to retreat to learning a (black-box) dynamics model (Chua et al., 2018; Janner et al., 2020). Recently, some authors have designed model architectures with neural networks that use physics-based inductive biases, also known as grey-box models (Lutter et al., 2019a;b; Greydanus et al., 2019).

While research has focused on improving model quality, a question that has received little attention yet is: ***Is model-based reinforcement learning even limited by the model quality?*** For data augmentation algorithms, e.g. DynaQ and model-based policy optimization (MBPO) (Janner et al., 2019), the answer is yes, as these methods treat model- and real-environment samples the same. In this paper, we empirically address this question for a specific class of model-based reinforcement learning — *value expansion methods*. Using the true dynamics model, we empirically show that the sample efficiency does not increase significantly when a perfect model is used. Furthermore, we find that increasing the rollout horizon, even with oracle dynamics, yields *diminishing returns* in improving sample efficiency. With the phrase *diminishing returns*, we refer to its definition in economics (Case & Fair, 1999). In the context of rollout horizons in model-based RL, we mean that the *increase in performance gain decreases with each additional model step* along a modelled trajectory.

Running our analysis would be highly inefficient with traditional CPU-based physics simulators (Todorov et al., 2012; Coumans & Bai, 2016–2021). For this reason, we use Brax (Freeman et al., 2021), a new JAX (Bradbury et al., 2018) based simulator which runs entirely on the GPU. This allows us to seamlessly integrate oracle dynamics rollouts into the entire training procedure. Furthermore, Brax's differentiability is necessary for implementing value gradient methods with oracle dynamics.[1]

Contributions In this paper, we empirically address the question of whether the sample efficiency of value expansion methods is limited by the dynamics model quality. We summarize our contributions as follows. (1) Using the oracle dynamics model, we empirically show that the sample efficiency of value expansion methods does not increase significantly over a learned model. (2) We find that increasing the rollout horizon yields diminishing returns in improving sample efficiency, even with oracle dynamics. (3) We show that off-policy RETRACE (Munos et al., 2016) is a very strong baseline, which adds virtually no computational overhead. (4) Our experiments show that for the critic expansion, the benefits of using a learned model does not appear to justify the added computational complexity.

[1] We will open-source our code upon publication.

2 H-STEP POLICY EVALUATION AND IMPROVEMENT

In this section, we introduce the components of value expansion.

Reinforcement Learning Consider a discrete-time Markov Decision Process (MDP) (Puterman, 2014), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho, \gamma \rangle$ with state space $\mathcal{S} \subseteq \mathbb{R}^n$ and action space $\mathcal{A} \subseteq \mathbb{R}^m$. At each time step t, the agent observes a state $s_t \in \mathcal{S}$ and samples an action $a_t \in \mathcal{A}$ according to a (stochastic) policy $a_t \sim \pi(\cdot|s_t)$. The environment returns a next state $s_{t+1} \in \mathcal{S}$ according to the transition probability density function $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$ and a scalar reward $r_t = \mathcal{R}(s_t, a_t)$. The initial state of a trajectory is sampled from the initial state distribution $s_0 \sim \rho$. $\gamma \in [0, 1)$ is a discount factor. Wherever it is clear, we will make use of the tick notation s' and a' to refer to the next state and action at time $t + 1$. A trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ is a sequence of states and actions. The objective of an RL agent is to find an optimal policy that maximizes the expected sum of discounted rewards $\pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim\rho,\pi,\mathcal{P}} \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. Often π is parametrized and, in the model-free case, it is common to optimize its parameters with on-policy (Sutton et al., 1999; Peters & Schaal, 2008; Schulman et al., 2015; 2017) or off-policy gradient methods (Degris et al., 2012; Silver et al., 2014; Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018).

Model-Based Reinforcement Learning In model-based RL we learn an approximate model $\hat{\mathcal{P}}$ of the true dynamics, and optionally also an approximate reward function $\hat{\mathcal{R}}$. There are several ways to use these models for policy optimization: generate simulated data to populate a replay buffer, e.g. DynaQ (Sutton, 1990) and model-based policy optimization (MBPO) (Janner et al., 2019); planning with model predictive control (MPC) and stochastic optimization, e.g. probabilistic ensembles with trajectory sampling (PETS) (Chua et al., 2018) and deep planning network (PlaNet) (Hafner et al., 2019); and gradient-based optimization with backpropagation through time, e.g. probabilistic inference for learning control (PILCO) (Deisenroth & Rasmussen, 2011), stochastic value gradient (SVG) (Heess et al., 2015), model-augmented actor-critic (MAAC) (Clavera et al., 2020) and SAC-SVG(H) (Amos et al., 2021).

Maximum-Entropy Reinforcement Learning (MaxEnt RL) In MaxEnt RL, the MDP reward r_t is augmented with a policy entropy term, which avoids collapsing to a deterministic policy, thus implicitly enforcing continuing exploration (Ziebart et al., 2008; Fox et al., 2016; Haarnoja et al., 2017). The (soft) action-value function $Q^\pi(s_0, a_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t (r_t - \alpha \log \pi(a_t|s_t))\right]$ computes the expected sum of discounted soft-rewards following a policy π. The goal is to maximize $\mathcal{J}_\pi = \mathbb{E}_{s_0\sim\rho}\left[\sum_{t=0}^{\infty} \gamma^t (r_t - \alpha \log \pi(\,\cdot \mid s_t))\right]$ w.r.t. π. Haarnoja et al. (2018) have proposed the Soft Actor-Critic (SAC) algorithm to solve this objective. The SAC actor and critic losses are defined as

$$\mathcal{J}_\pi(\mathcal{D}) = \mathbb{E}_{s\sim\mathcal{D}}\left[-V^\pi(s)\right] = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi(\cdot|s)}\left[\alpha \log \pi(a|s) - Q^\pi(s, a)\right], \tag{1}$$

$$\mathcal{J}_Q(\mathcal{D}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma V^\pi(s') - Q^\pi(s, a)\right)^2\right], \tag{2}$$

where $\mathcal{D}$ is a dataset of previously collected transitions (replay buffer), and the next state value target is $V^\pi(s') = \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q^\pi(s', a') - \alpha \log \pi(a'|s')\right]$. In practice, the implementation of SAC uses a double Q-function (Fujimoto et al., 2018), but to keep the notation simple we do not include it here.

Value Expansion Through the recursive definition of the value- and action-value functions (Bellman, 1957), they can be approximated by rolling out the model for H time steps. We refer to this (single sample) estimator as the H-step value expansion, defined as

$$Q^H(s, a) = \mathcal{R}(s, a) + \sum_{t=1}^{H-1} \gamma^t \left(r_t - \alpha \log \pi(a_t|s_t)\right) + \gamma^H V^\pi(s_H) \tag{3}$$
$$\text{with} \quad Q^0(s, a) \coloneqq Q^\pi(s, a) \quad \text{and} \quad V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^\pi(s, a) - \alpha \log \pi(a|s)\right]$$

To avoid notation clutter we write $Q^{\pi,H}(s, a) = Q^H(s, a)$. For $H = 0$ this estimator reduces to the function approximator used in SAC. It is important to note that these trajectories are on-policy.

Actor Expansion The H-step value expansion can now be used in the actor update, by incorporating Equation 3 in the actor loss (Equation 1)

$$\mathcal{J}_\pi^H(\mathcal{D}) = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi(\cdot|s)}\left[\alpha \log \pi(a|s) - Q^H(s, a)\right], \tag{4}$$

Table 1: Overview of key related literature. We categorize model-based RL algorithms using value expansion in the actor and/or critic update, and further note which ones use on-policy trajectories resulting from a dynamics model and which use off-policy trajectories from a replay buffer.

Algorithm	Expansion		Policy	
	Actor	Critic	On	Off
SVG (Heess et al., 2015)	×			×
IVG (Byravan et al., 2020)	×		×	
SAC-SVG (Amos et al., 2021)	×		×	
MAAC (Clavera et al., 2020)	×		×	
MVE (Feinberg et al., 2018)		×	×	
STEVE (Buckman et al., 2018)		×	×	
RETRACE (Munos et al., 2016)		×		×
Dreamer (Hafner et al., 2020)	×	×	×	

which resembles the class of SVG algorithms (Heess et al., 2015; Byravan et al., 2020; Amos et al., 2021; Clavera et al., 2020). The difference from the original SVG is the absence of the entropy term. Note that, for $H = 0$ this definition reduces to the regular SAC actor loss in Equation 1. We will refer to the H-step value expansion for the actor update as *actor expansion* (AE).

Critic Expansion Incorporating value expansion in the policy evaluation step, results in TD-λ style (Sutton & Barto, 2018; Schulman et al., 2016) or Model-Based Value Expansion (MVE) methods (Wang et al., 2020; Buckman et al., 2018). The corresponding critic loss is defined as

$$\mathcal{J}_Q^H(\mathcal{D}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D},\ a'\sim\pi(\cdot|s')}\left[\left(r + \gamma(Q^H(s',a') - \alpha\log\pi(a'|s')) - Q^\pi(s,a)\right)^2\right]. \tag{5}$$

Similar to actor expansion, for $H = 0$ the critic update reduces to the SAC critic update Equation 2. We will refer to the H-step value expansion for the critic update as *critic expansion* (CE).

Extension to Off-Policy Trajectories We can naturally extend the action-value expansion in Equation 3 with the RETRACE formulation (Munos et al., 2016), which allows to use off-policy trajectories collected with a policy $\mu \neq \pi$ (e.g. stored in a replay buffer). The H-step RETRACE state-action value expansion (single sample) estimator is defined as

$$Q^H_{\text{retrace}}(s,a) = \mathcal{R}(s,a) + \left[\sum_{t=1}^{H-1}\gamma^t\left(c_t r_t + c_{t-1}V_t - c_t Q_t\right)\right] + \gamma^H c_{H-1}V_H, \tag{6}$$

$$\text{with}\quad c_t = \prod_{j=1}^{t}\lambda\min\left(1, \frac{\pi(a_j|s_j)}{\mu(a_j|s_j)}\right),\ c_0 := 1,\ Q^0_{\text{retrace}}(s,a) := Q^\pi(s,a)\quad\text{and}\quad \lambda\in[0,1].$$

This extension has the desirable property that $c_t = 1$ for on-policy trajectories ($\mu = \pi$), and Q^H_{retrace} reduces to the H-step action-value target from Q^H (Equation 3). Furthermore, for $H = 0$, it reduces to the SAC target. The second desirable property is that trajectories from real the environment or a dynamics model can be used. Hence, we use the action-value expansion RETRACE formulation to evaluate all combinations of CE and AE with on- and off-policy data and real and model-generated data. Since RETRACE encompasses all the variants we introduced, for readability, we will from now on overload notation and define $Q^H(s,a) := Q^H_{\text{retrace}}(s,a)$.

Table 1 provides an overview of related value expansion literature and displays further information about their usage of actor and critic expansion and on- and off-policy data.

A natural combination is to merge actor and critic expansion into one algorithm. However, we decided to do our analysis on each method separately, to better understand their issues.

3 Empirical Study

In this experimental section, we investigate the initial research question, **how much will more accurate dynamics models increase sample efficiency?** To answer it, we construct an experimental setup that replaces the learned dynamics model with an oracle model – the environment's simulator

– which is the most accurate model that could be learned. In this scenario, no model error exists, which lets us examine the impact of the compounding model error — or rather the lack of it. In these experiments, we specifically exclude data augmentation methods; as for these methods, it is trivial to realize that they can arbitrarily benefit from the oracle dynamics, i.e. more accurate dynamics models. In the following, we conduct a series of empirical studies across the introduced value expansion algorithms on five standard continuous control benchmark tasks: InvertedPendulum, Cartpole SwingUp, Hopper, Walker2d and HalfCheetah. These tasks range from simple to more complex, with the latter three including contact forces, often resulting in discontinuous dynamics. We implemented all experiments using the Brax simulator (Freeman et al., 2021) and the JAX framework (Bradbury et al., 2018) for seamless integration and efficient GPU-only training.
In the following subsections, we first present the premise of this paper — the diminishing returns of value expansion methods even with a perfect dynamics model (section 3.1). Given the oracle model, we *rule out the existence of a compounding model error* and, therefore, possible impacts on the learning performance. Next, we turn to the generated H-step target values and analyse the impact of increasing horizons (section 3.2), and finally, in section 3.3 we look into the resulting critic and actor gradients. Throughout this section, we also use a learned dynamics model, implemented as a stochastic ensemble of neural networks whose output is the mean and variance of the predicted next state, similar to Chua et al. (2018) and Janner et al. (2019).

3.1 Diminishing returns on continuous control tasks

Figure 1a presents an extensive evaluation of the training performance for the referred value expansion methods based on SAC. Figure 1b shows results for DDPG, which serves as a baseline to ensure that the diminishing returns do not simply stem from training a stochastic policy. All experiments are performed with the oracle dynamics model and a learned model for reference. We further provide results for RETRACE as a model-free H-step baseline that uses off-policy trajectories from a replay buffer. We used rollout horizons ranging from very short to long $H \in [0, 1, 3, 5, 10, 20, 30]$. $H = 0$ reduces to vanilla SAC.

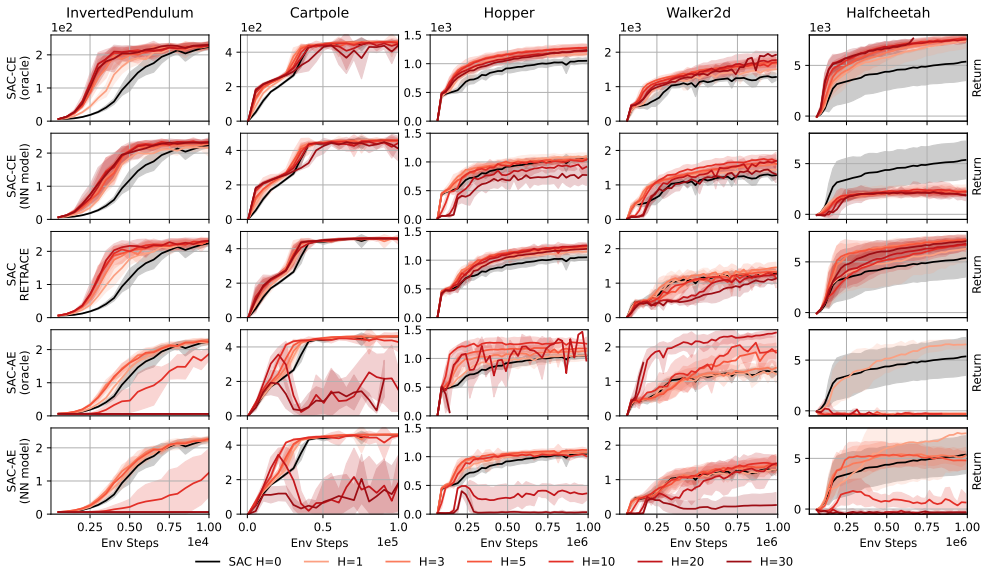
Within short horizons, less than 5 steps, we consistently observed that in most cases a single increase in the rollout horizon leads to increased performance. However, the rate of improvement decreases quickly with each additional expansion step (see figure 1a), and close to all variants of SAC reach the peak performance for $H = 5$. We refer to this phenomenon as the *diminishing return* of value expansion methods. SAC with Actor Expansion (SAC-AE), either with an oracle (4th row) or a learned dynamics model (5th row), typically shows comparable or lower returns than SAC with Critic Expansion (SAC-CE) (1st and 2th rows), higher variance and learning instability, especially with longer horizons. Agents with a neural network dynamics model show the same pattern but generally slightly reduced overall learning performance compared to the oracle model, which is expected. Nevertheless, with the oracle model experiments, we can say that the diminishing returns are not primarily caused by the compounding model errors, which might have been hypothesised by only observing the experiments with the learned neural network model. Moreover, since both SAC and DDPG show the same qualitative results, we conclude that the diminishing returns do not directly just originate from our choice of policy gradient algorithm.

3.2 Empirical analysis of $Q^H(s,a)$ target distributions

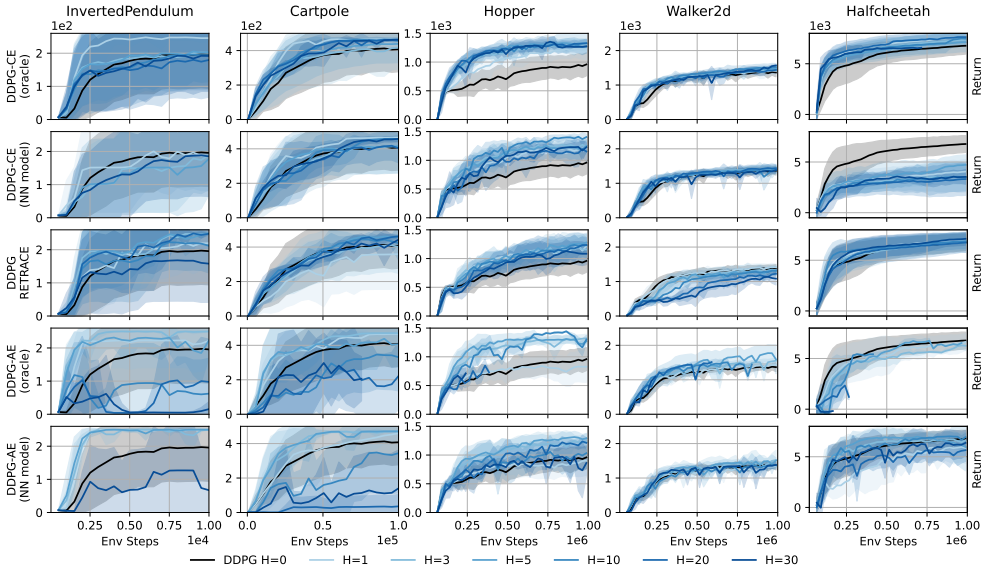
Since $Q^H(s,a)$ is estimated by rolling out a stochastic model, which can be seen as traversing a Markov reward process for H steps, it is well known that the target estimator's variance increases linearly with H, similar to the policy gradient REINFORCE algorithm (Williams, 1992), as pointed out in (Pflug, 1996; Bhatt et al., 2019; Carvalho et al., 2021). On the other hand, the targets will be more accurate (less bias), due to discounting the learned Q-function (Feinberg et al., 2018). In our experiments, the oracle dynamics guarantee that there is no bias coming from a learned model. In this section, we analyse the hypothesis that the increased target variance is the main reason for the diminishing returns. For that, we compare the true return distribution, approximated by particles of infinite horizon Monte Carlo returns, to the targets produced by different rollout horizons, using

$$D_{\mathrm{W}}^{H} = \mathbb{E}_{s,a\sim\mathcal{D}}\big[D_{\mathrm{W}}\big(\{Q_p^{\infty}(s,a)\}_{p=1\dots P} \big| \{Q_p^{H}(s,a)\}_{p=1\dots P}\big)\big],$$

where D_W is the Wasserstein distance (Peyré & Cuturi, 2019), $P = 100$ particles per target (to model the distribution over targets) and $Q^{\infty}(s,a)$ is the true Monte Carlo return to compare



(a) SAC



(b) DDPG

Figure 1: This plot shows the diminishing return of critic and actor expansion methods. For each, we present learning curves for multiple rollout horizons H based on SAC and DDPG. Critic expansion does not benefit noticeably from using horizons greater than five in most cases. There is also no clear trend that critic expansion methods would benefit from better models since the neural network model (row 2) performs mostly on par with the oracle model (row 1). Model-free RETRACE often is a strong contender, showing comparable performance. For actor expansion (rows 4 and 5), larger horizons can be detrimental, even with an oracle model. We plot the mean (solid line) and standard deviation (shaded area) of 9 random seeds of the average episode undiscounted return on the y-axis. The x-axis displays the number of interaction steps with the real environment. Note: some actor expansion runs for longer horizons terminate early due to exploding gradients (See section 3.3).

against (in practice, we use a large rollout horizon based on the discount factor γ). Finally, we evaluate the expectation of this estimator over 10^4 samples (s, a) drawn from the replay buffer. We repeat this analysis for all horizons and 25 checkpoints along the vanilla SAC training.

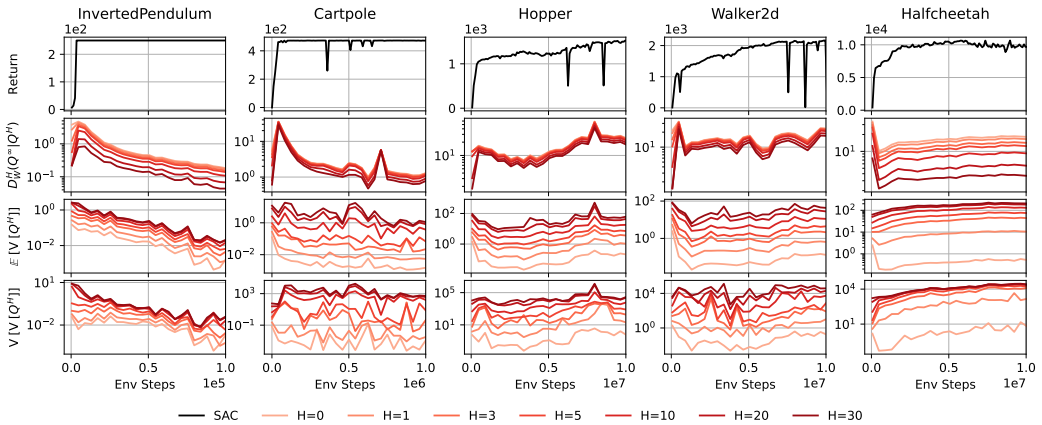


Figure 2: Analysis of H-step targets for multiple checkpoints along the training process. The first row shows the undiscounted episodic return of SAC along training for one seed. The checkpoints along this single training run are used for the analyses in the following rows. The second row shows the Wasserstein distance between the sample-based target distribution predicted by the H-step estimator and the true target distribution represented by Monte Carlo samples. The third row displays the mean of the variance over particles of the sample-based target distributions, and the bottom row displays the variance of the variances over particles of the sample-based target distributions.

The 2nd row of figure 2 shows how the Wasserstein distance decreases with increasing horizons, from which we conclude that longer rollouts better capture the true return distribution. Rows 3 and 4 depict the expectation of the variances over particles, and the variance of the variances over particles of the sample-based target distributions

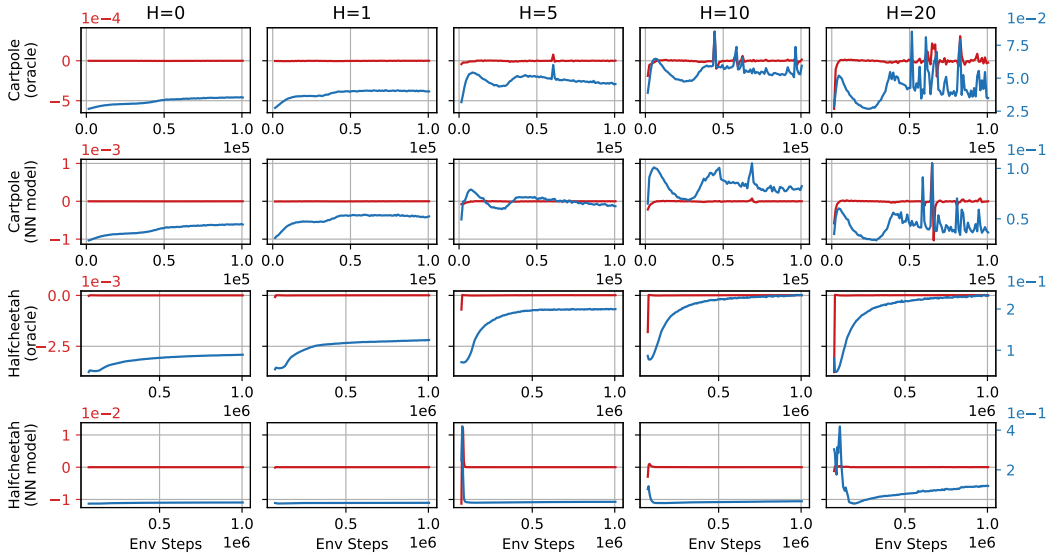
$$\mathbb{E}_{s,a\sim\mathcal{D}}[\text{Var}[\{Q_p^H(s,a)\}_{p=1..P}]], \qquad \text{Var}_{s,a\sim\mathcal{D}}[\text{Var}[\{Q_p^H(s,a)\}_{p=1..P}]].$$

These measures show that the expected value and variance increase with the horizon length. This observation aligns with the theory at the beginning of this section that when traversing a Markov reward process, the estimator's variance increases with the horizon.

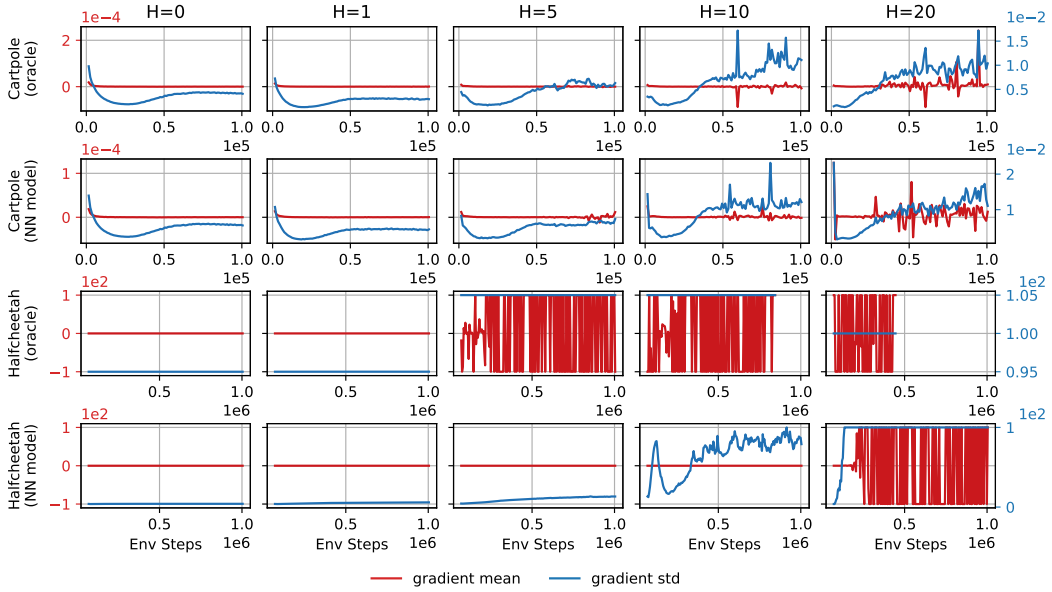
3.3 Impact on the Gradients

We now analyse the impact of the different rollout horizons on the actor and critic gradients. It is important to note that the critic gradient $\nabla\mathcal{J}_Q^H$ of the loss in Equation 5 is just simply differentiating through Q^π in the squared loss, since we assume $Q^H(s',a')$ is a fixed target as is common in RL algorithms (Ernst et al., 2005; Riedmiller, 2005; Mnih et al., 2013). To calculate the actor gradient $\nabla\mathcal{J}_\pi^H$ we perform backpropagation through time through the whole trajectory resulting from expanding $Q^H(s,a)$ in Equation 4. To backpropagate through the stochastic nodes, we use the reparametrization trick (Kingma & Welling, 2014) as done in SAC-SVG (Amos et al., 2021), MAAC (Clavera et al., 2020) or SHAC (Xu et al., 2022).

Critic expansion gradients Figure 3a shows the mean and standard deviation of the critic gradients along the training procedure for the critic expansion algorithms. Due to space constraints, we present only a subset of the environments and horizons in the main paper (see figure A.1 in the appendix for all experiments). For all environments and horizons, the gradient mean and standard deviation appear within the same order of magnitude. Although, as expected, we observe a slight increase in the standard deviation with an increasing horizon for both the oracle and the learned neural network model. From this observation, we show that the critic gradients are probably not the source of the diminishing returns. Looking at the first and second rows of figure 1a, the returns of the oracle dynamics and the learned neural network model are comparable. Also, increasing the horizon for target estimation in critic expansion does not seem to translate into much better returns. Therefore, the overhead of building a better model for critic expansion and using longer horizons does not seem to translate into significantly better returns - hence the diminishing returns. For critic expansion, our experiments show that using off-policy data with RETRACE might be an option to



(a) Critic gradients mean and standard deviation for H-step critic expansion.



(b) Actor gradients mean and standard deviation for H-step actor expansion.

Figure 3: These plots show the critic and actor gradients for selected environments and rollout horizons. In figure 3a we observe that the critic gradients show resonable magnitudes and low variance. On the contrary, in figure 3b, we see that for some environments the actor gradient explodes, which hinders learning. For readability, we clip large gradients to 10^2 (Halfcheetah environment).

consider without losing performance if the computational overhead of learning and rolling out a model does not justify it.

Actor expansion gradients Figure 3b shows the mean and standard deviation of the actor gradients along the training process for the actor expansion algorithms (see figure A.2 in the appendix for all experiments). Contrary to critic expansion, in actor expansion, we backpropagate through the whole rollout trajectory to compute the gradient w.r.t. the policy parameters. Comparing the results from the fourth and fifth rows of figure 1a, we observe that having a better dynamics model does

not directly translate into faster learning or higher rewards since both the oracle and neural network model show the same effect - lower returns for larger horizons. We partially correlate these results with the actor gradients in figure 3b. Note, e.g., how in the Halfcheetah environment, the gradient mean and standard deviation explodes for horizons larger or equal to 5 for the oracle and 20 for the learned models. Therefore, increasing the horizon must be done with care even if a better model does not suffer from compounding model errors. Our observation is that when doing actor expansion, developing a very accurate differentiable model that predicts into the future might not be worth it since, due to gradient instability, it might only be possible to expand for a few steps; otherwise we might need methods to handle gradient variance (Parmas et al., 2018).

4 Conclusion & Future Work

In this work, we answer the question of whether dynamics model errors are the main limitation for improving the sample efficiency of model-based RL. More specifically, we focus on value expansion methods, which unroll a model in the actor- or critic-update step. For these methods, we observe that in most cases, oracle dynamics are barely better than current learned dynamics models. Furthermore, we show that model-free value expansion through RETRACE, which simply uses off-policy rollouts from a replay buffer, is often only negligibly worse. With this finding in mind, we argue that the computational overhead of model learning for value expansion methods outweighs the benefits. Most importantly, we present the diminishing returns of value expansion methods. Our experiments with the oracle dynamics model show that model errors are not the root cause of this phenomenon, contrary to common belief. We cannot pinpoint the root causes – neither the produced targets nor the produced gradients reveal clear signals. Our experiments show that for model-based RL we may rethink if we should put the effort into learning better models, since computing critic expansion targets with RETRACE matched the oracle dynamics, and updating policies with actor expansion methods can lead to gradient problems. Therefore, the search for the root cause remains an open research problem. From this observation, we conclude that the gains in sample efficiency with this class of algorithms, which can be gained by learning more accurate models, are limited.

References

Zaheer Abbas, Samuel Sokota, Erin Talvitie, and Martha White. Selective dyna-style planning under limited model capacity. In *International Conference on Machine Learning*, 2020.

Brandon Amos, Samuel Stanton, Denis Yarats, and Andrew Gordon Wilson. On the model-based stochastic value gradient for continuous reinforcement learning. In *Learning for Dynamics and Control*, 2021.

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1 edition, 1957.

Sujay Bhatt, Alec Koppel, and Vikram Krishnamurthy. Policy gradient using weak derivatives for reinforcement learning. In *Conference on Decision and Control*, 2019.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, 2018.

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

João Carvalho, Davide Tateo, Fabio Muratore, and Jan Peters. An empirical analysis of measure-valued derivatives for policy gradients. In *International Joint Conference on Neural Networks*, 2021.

K.E. Case and R.C. Fair. *Principles of Economics*. Prentice Hall, 1999.

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

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

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

Thomas Degris, Martha White, and Richard S. Sutton. Off-policy actor-critic. In *International Conference on Machine Learning*, 2012.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, 2011.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1–2):1–141, 2013.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. In *International Conference on Machine Learning*, 2018.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the Noise in Reinforcement Learning via Soft Updates. In *Conference on Uncertainty in Artificial Intelligence*, 2016.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL `http://github.com/google/brax`.

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

Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, 2019.

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

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

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

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

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.

Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. In *Advances in Neural Information Processing Systems*, 2020.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

Hang Lai, Jian Shen, Weinan Zhang, and Yong Yu. Bidirectional model-based policy optimization. In *International Conference on Machine Learning*, 2020.

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

Michael Lutter, Kim Listmann, and Jan Peters. Deep lagrangian networks for end-to-end learning of energy-based control for under-actuated systems. In *International Conference on Intelligent Robots and Systems*, 2019a.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019b.

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

Andrew S Morgan, Daljeet Nandha, Georgia Chalvatzaki, Carlo D'Eramo, Aaron M Dollar, and Jan Peters. Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning. In *International Conference on Robotics and Automation*, 2021.

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

Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive Processing*, 12(4):319–340, 2011.

Paavo Parmas, Carl E Rasmussen, Jan Peters, and Kenji Doya. Pipps: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*, 2018.

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 2008.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Trends in Machine Learning*, 11 (5-6):355–607, 2019.

Georg Ch. Pflug. *Optimization of Stochastic Models*. Springer, 1996.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Carl E Rasmussen and Christopher Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

Martin A. Riedmiller. Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, 2005.

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

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

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

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Machine Learning*, 1990.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

Richard S. Sutton, David A. McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1999.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

Junjie Wang, Qichao Zhang, Dongbin Zhao, Mengchen Zhao, and Jianye Hao. Dynamic horizon value estimation for model-based reinforcement learning. *arXiv preprint arXiv:2009.09593*, 2020.

P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. doi: 10.1109/5.58337.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

Chenjun Xiao, Yifan Wu, Chen Ma, Dale Schuurmans, and Martin Müller. Learning to combat compounding-error in model-based reinforcement learning. *arXiv preprint arXiv:1912.11206*, 2019.

Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. 2022.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Conference on Artificial Intelligence*, 2008.

A APPENDIX

A.1 GRADIENT EXPERIMENTS

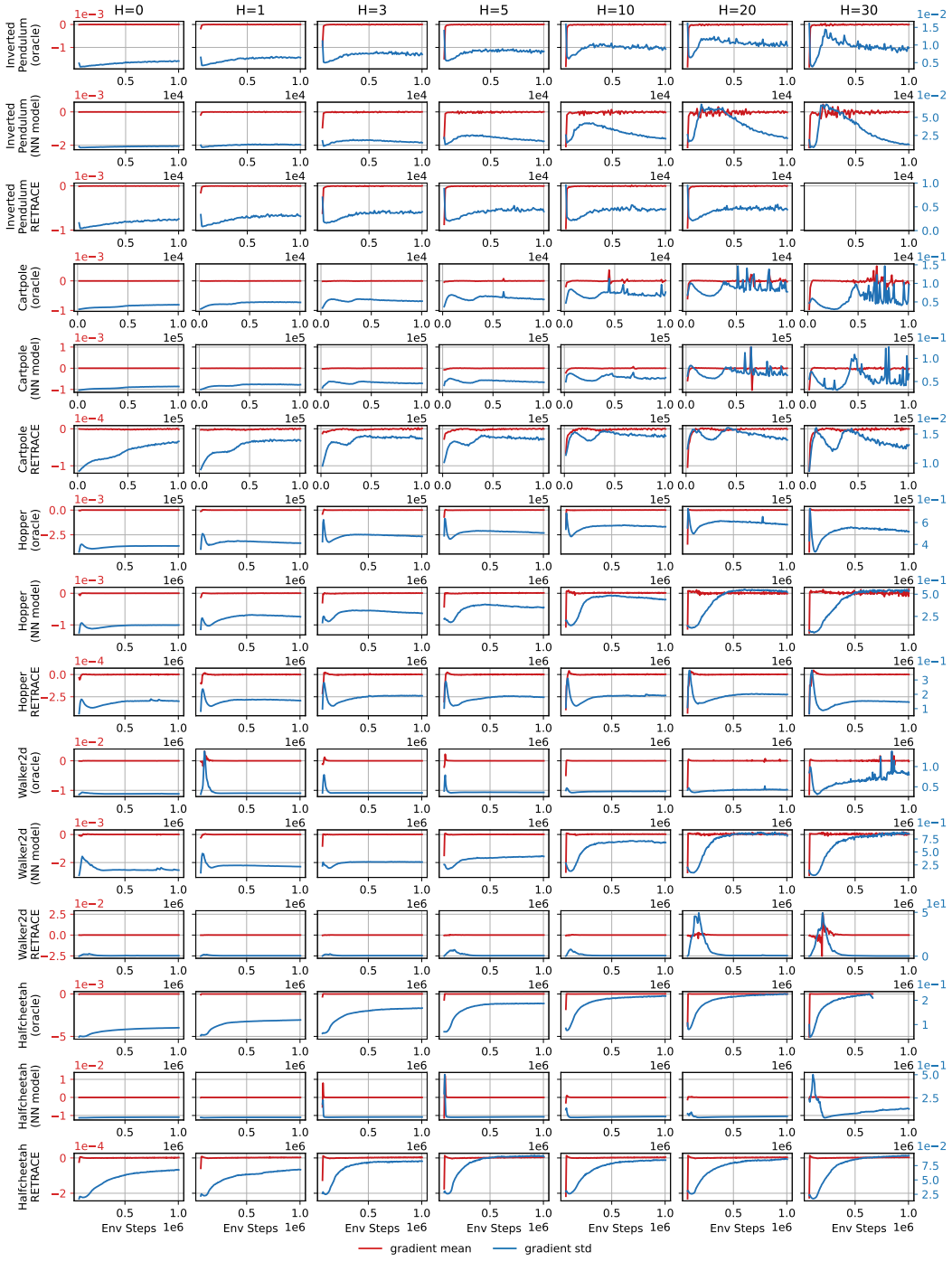


Figure A.1: These plots show the critic gradients mean and standard deviation of the critic expansion for all environments and all rollout horizons.

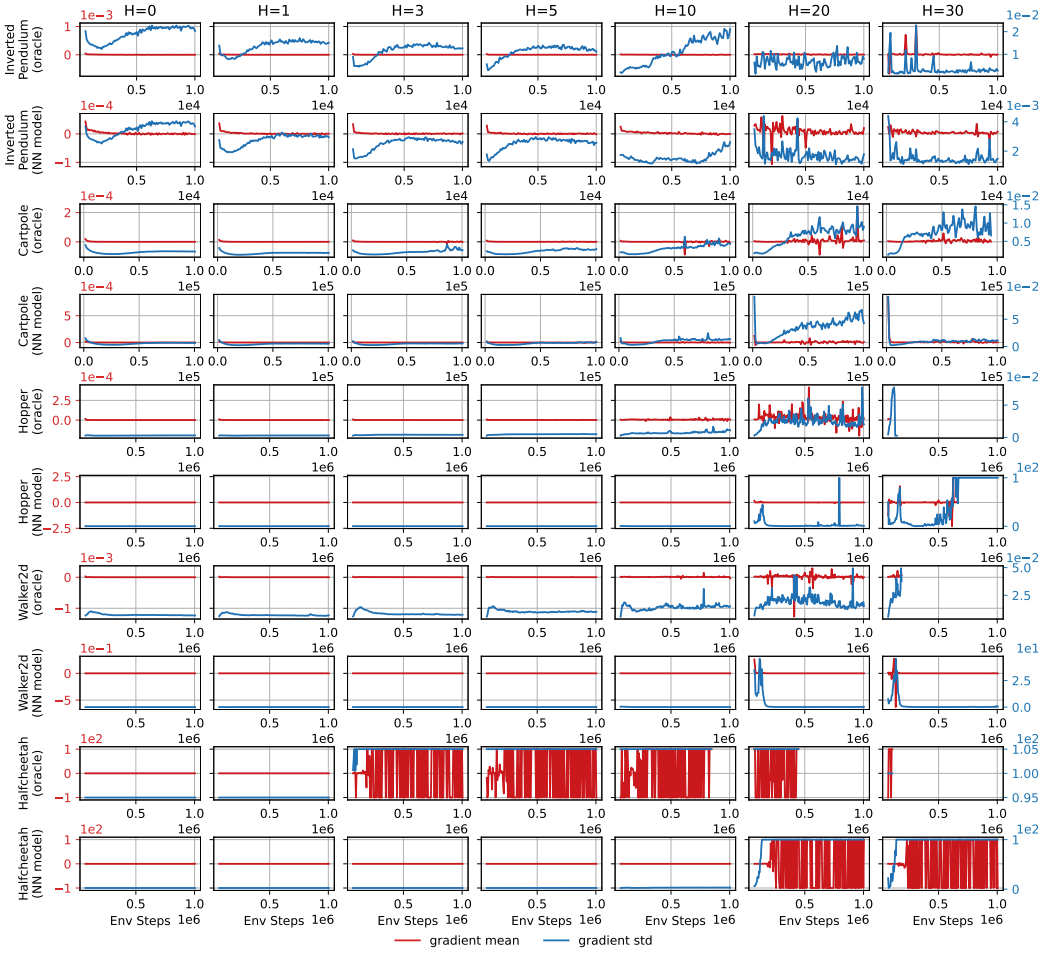


Figure A.2: These plots show the actor gradients mean and standard deviation of the actor expansion for all environments and all rollout. For readability, we clip large gradients to 10^2 (Halfcheetah environment).