Model-Augmented Actor-Critic: Backpropagating through Paths

Ignasi Clavera*
University of California, Berkeley
iclavera@berkeley.edu

Yao Fu*
University of California, Berkeley
violetfuyao@berkeley.edu

Pieter Abbeel
University of California, Berkeley
pabbeel@cs.berkeley.edu

Abstract

Current model-based reinforcement learning approaches use the model simply as a learned black-box simulator to augment the data for policy optimization or value function learning. In this paper, we show how to make more effective use of the model by exploiting its differentiability. We construct a policy optimization algorithm that uses the pathwise derivative of the learned model and policy across future timesteps. Instabilities of learning across many timesteps are prevented by using a terminal value function, learning the policy in an actor-critic fashion. Furthermore, we present a derivation on the monotonic improvement of our objective in terms of the gradient error in the model and value function. We show that our approach (i) is consistently more sample efficient than existing state-of-the-art model-based algorithms, (ii) matches the asymptotic performance of model-free algorithms, and (iii) scales to long horizons, a regime where typically past model-based approaches have struggled.

1 Introduction

Model-based reinforcement learning (RL) offers the potential to be a general-purpose tool for learning complex policies while being sample efficient. When learning in real-world physical systems, data collection can be an arduous process. Contrary to model-free methods, model-based approaches are appealing due to their comparatively fast learning. By first learning the dynamics of the system in a supervised learning way, it can exploit off-policy data. Then, model-based methods use the model to derive controllers from it, either parametric [14, 1, 9] or non-parametric [17, 2].

Current model-based methods learn with an order of magnitude less data than their model-free counterparts while achieving the same asymptotic convergence. Tools like ensembles, probabilistic models, and meta-learning have been used to achieved such performance [11, 2, 3]. However, the model usage in all of these methods is the same: simple data augmentation. They use the learned model as a black-box simulator generating samples from it. In high-dimensional environments or environments that require longer planning, substantial sampling is needed to provide meaningful signal for the policy. *Can we further exploit our learned models?*

In this work, we propose to estimate the policy gradient by backpropagating its gradient through the model using the pathwise derivative estimator. Since the learned model is differentiable, one can link together the model, reward function, and policy to obtain an analytic expression for the gradient

*Equal contribution

NeurIPS 2019 Workshop on Robot Learning: Control and Interaction in the Real World, Vancouver, Canada

of the returns with respect to the policy. By computing the gradient in this manner, we obtain an expressive signal that allows rapid policy learning. We avoid the instabilities that often result from back-propagating through long horizons by using a terminal Q-function. This scheme fully exploits the learned model without harming the learning stability in previous approaches [11, 8]. The horizon at which we apply the terminal Q-function acts as a hyperparameter between model-free (when fully relying on the Q-function) and model-based (when using a longer horizon) of our algorithm.

The main contribution of this work is a model-based method that significantly reduces the sample complexity compared to state-of-the-art model-based algorithms [9, 1]. For instance, we achieve a 10k return in half-cheetah environment in just 50 trajectories. We theoretically justify our optimization objective and derive the monotonic improvement of our learned policy in terms of the Q-function and the model error. The theoretical results are experimentally analyzed. Finally, we pinpoint the importance of our objective by ablating each component of our algorithm. The results are reported in four model-based benchmarking environments [22, 21]. The low sample complexity and high performance of our method carry high promise towards learning directly on real robots.

2 Reinforcement Learning

A discrete-time finite Markov decision process (MDP) $\mathcal{M}$ is defined by the tuple $(\mathcal{S}, \mathcal{A}, f, r, \gamma, p_0, T)$. Here, $\mathcal{S}$ is the set of states, $\mathcal{A}$ the action space, $s_{t+1} \sim f(s_t, a_t)$ the transition distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $p_0 : \mathcal{S} \to \mathbb{R}_+$ represents the initial state distribution, γ the discount factor, and T is the horizon of the process. We define the return as the sum of rewards $r(s_t, a_t)$ along a trajectory $\tau := (s_0, a_0, ..., s_{T-1}, a_{T-1}, s_T)$. The goal of reinforcement learning is to find a policy $\pi_{\boldsymbol{\theta}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ that maximizes the expected return, i.e., $\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \mathbb{E}[\sum_t \gamma^t r(s_t, a_t)]$.

Actor-Critic. In actor-critic methods, the policy (actor), is updated with the gradient of the Q-function (critic) w.r.t the action, and the Q-function is update to minimize the Bellman error:

$$\nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta}) = \mathbb{E}\left[\nabla_a Q^\pi_{\boldsymbol{\psi}}(s,a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(s)\right] \qquad J_Q(\boldsymbol{\psi}) = \mathbb{E}[(Q^\pi_{\boldsymbol{\psi}}(s,a) - (r(s,a) + \gamma Q^\pi_{\boldsymbol{\psi}}(s', a')))^2]$$

The key benefit of this update is that it can be applied in an off-policy fashion, sampling random mini-batches of transitions from an experience replay buffer [13].

Model-Based RL. Model-based methods, contrary to model-free RL, which does not explicitly model state transitions, they learn the transition distribution, also known as dynamics model, from the experience. This can be done with a parametric function approximator $\hat{p}_{\boldsymbol{\phi}}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. In such case, the parameters $\boldsymbol{\phi}$ of the dynamics model are optimized by maximum likelihood.

3 Algorithm

We develop a new algorithm that explicitly optimizes the model-augmented actor-critic (MAAC) objective. The overall algorithm is divided into three main steps: model learning, policy optimization, and Q-function learning. The theoretical guarantees of our method are provided in the Appendix.

Model learning. In order to prevent overfitting and overcome model-bias [4], we use a bootstrap ensemble of dynamics models $\{\hat{f}_{\boldsymbol{\phi}_1}, ..., \hat{f}_{\boldsymbol{\phi}_M}\}$. Each of the dynamics models parameterizes the mean and the covariance of a Gaussian distribution with diagonal covariance. The bootstrap ensemble captures the epistemic uncertainty, uncertainty due to the limited capacity or data, while the probabilistic models are able to capture the aleatoric uncertainty [2], inherent uncertainty of the environment. We denote by $f_{\boldsymbol{\phi}}$ the transitions dynamics resulting from $\boldsymbol{\phi}_U$, where $U \sim \mathcal{U}[M]$ is uniform random variable. The dynamics models are trained via maximum likelihood with early stopping on a validation set.

Policy Optimization. We extend the MAAC objective with an entropy bonus [7], and perform policy learning by maximizing

$$J_\pi(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r(\hat{s}_t) + \gamma^H Q_{\boldsymbol{\psi}}(\hat{s}_H, a_H)\right] + \beta \mathcal{H}(\pi_{\boldsymbol{\theta}})$$

where $\hat{s}_{t+1} \sim f_{\boldsymbol{\phi}}(\hat{s}_t, a_t)$, $\hat{s}_0 \sim \mathcal{D}$, $a \sim \pi_{\boldsymbol{\theta}}$. We learn the policy by using the pathwise derivative of the model through H steps and the Q-function by sampling multiple trajectories from the same $\hat{s}_0$.

Hence, we learn a maximum entropy policy using pathwise derivative of the model through H steps and the Q-function. We compute the expectation by sampling multiple actions and states from the policy and learned dynamics, respectively.

Q-function Learning. In practice, we train two Q-functions [5] since it has been experimentally proven to yield better results. We train both Q functions by minimizing the Bellman error (Section 2):

$$J_Q(\psi) = \mathbb{E}[(Q_\psi(s_t, a_t) - (r(s_t, a_t) + \gamma Q_\psi(s_{t+1}, a_{t+1})))^2]$$

Similar to [9], we minimize the Bellman residual on states previously visited and imagined states obtained from unrolling the learned model. Finally, the value targets are obtained in the same fashion as the Stochastic Ensemble Value Expansion [1], using H as a horizon for the expansion.

Our method, MAAC, iterates between collecting samples , model training, policy optimization, and Q-function learning. First, we obtain trajectories from the real environment using the latest policy available and append them to a replay buffer $\mathcal{D}_{\text{env}}$, on which the dynamics models are trained until convergence. Then we collect imaginary data from the models: we collect k-step transitions by unrolling the latest policy from a randomly sampled state on $\mathcal{D}_{\text{env}}$. The imaginary data constitutes the $\mathcal{D}_{\text{model}}$, which together with the replay buffer, is used to learn the Q-function and train the policy.

4 Results

Our experimental evaluation aims to examine the following questions: 1) How does MAAC compares against state-of-the-art model-based and model-free methods? 2) Does the gradient error correlate with the derived bound?, 3) Which are the key components of its performance?, and 4) Does it benefit from planning at test time?

In order to answer the posed questions, we evaluate our approach on model-based continuous control benchmark tasks in the MuJoCo simulator [21, 22].

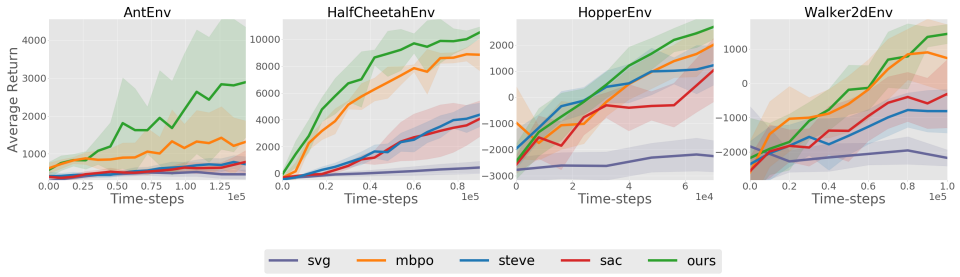

Figure 1: Comparison against state-of-the-art model-free and model-based baselines in four different MuJoCo environments. Our method, MAAC, attains better sample efficiency and asymptotic performance than previous approaches. The gap in performance between MAAC and previous work increases as the environments increase in complexity.

4.1 Comparison Against State-of-the-Art

We compare our method on sample complexity and asymptotic performance against state-of-the-art model-free (MF) and model-based (MB) baselines. Specifically, we compare against the model-free soft actor-critic (SAC) [6] as well as two state-of-the-art model-based baselines: model-based policy-optimization (MBPO) [9] and stochastic ensemble value expansion (STEVE) [1]. The original STEVE algorithm builds on top of the model-free algorithm DDPG [12], however, we implemented it on top of SAC. We also add SVG(1) [8] to comparison, which similar to our method uses the derivative of dynamic models to learn the policy.

The results, shown in Fig. 1, highlight the strength of MAAC in terms of performance and sample complexity. MAAC scales to higher dimensional tasks while maintaining its sample efficiency and asymptotic performance. In all the four environments, our method learns faster than previous MB and MF methods. We are able to learn near-optimal policies in the half-cheetah environment in just over 50 rollouts, while previous model-based methods need at least the double amount of data. Furthermore, in complex environments, such as ant, MAAC achieves near-optimal performance within 150 rollouts while other take orders of magnitudes more data.

4.2 Gradient Error

Here, we investigate how the bounds obtained relate to the empirical performance. In particular, we study the effect of the horizon of the model predictions on the gradient error. In order to do so, we construct a double integrator environment; since the transitions are linear and the cost is quadratic for a linear policy, we can obtain the analytic gradient of the expect return.

Figure 2 depicts the $L1$ error of the MAAC objective for different values of the horizon H as well as what would be the error using the true dynamics. As expected, using the true dynamics yields to lower gradient error since the only source comes from the learned Q-function that is weighted down by γ^H. The results using learned dynamics show that the error from the learned dynamics is lower than the one in the Q-function, but it scales poorly with the horizon. For short horizons the error decreases as we increase the horizon. However, large horizons is detrimental since it magnifies the error on the models.

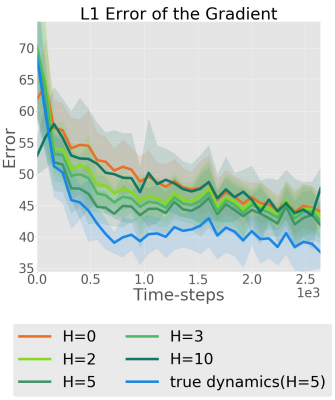

Figure 2: $L1$ error on the policy gradient when using the proposed objective for different values of the horizon H as well as the error from the true dynamics. The results correlate with the assumption that the error in the learned dynamics is lower than the error in the Q-function, as well as they correlate with the derived bounds.

4.3 Ablations

In order to investigate the importance of each of the components of our overall algorithm, we carry out an ablation test. Specifically, we test three different components: 1) not using the model to train the policy, i.e., set $H = 0$, 2) not using the STEVE targets for training the critic, and 3) using a single sample estimate of the path-wise derivative.

The ablation test is shown in Figure 3. The test underpins the importance of backpropagating through the model: setting H to be 0 inflicts a severe drop in the algorithm performance. On the other hand, using the STEVE targets results in slightly more stable training, but it does not have a significant effect. Finally, while single sample estimates can be used in simple environments, they are not accurate enough in higher dimensional environments such as ant.

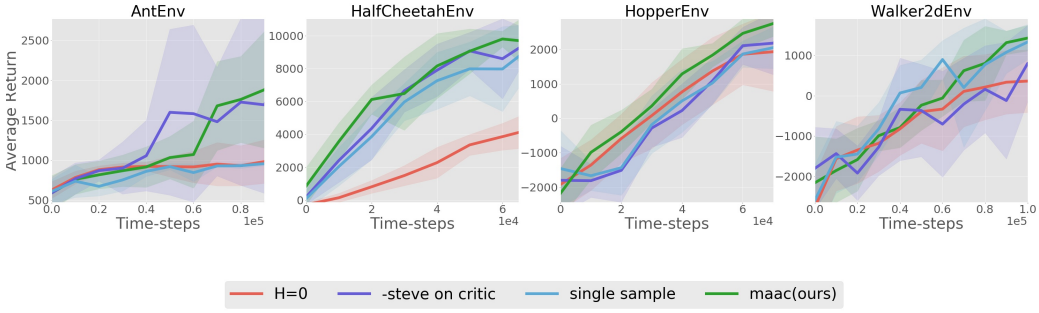

Figure 3: Ablation test of our method. We test the importance of several components of our method: not using the model to train the policy ($H = 0$), not using the STEVE targets for training the Q-function (-STEVE), and using a single sample estimate of the pathwise derivative. Using the model is the component that affects the most the performance, highlighting the importance of our derived estimator.

5 Conclusion

In this work, we present model-augmented actor-critic, MAAC, a reinforcement learning algorithm that uses a learned model by using the pathwise derivative across future timesteps. We theoretically analyzed the objective in terms of the model and value error and we derive a policy improvement expression with respect to those terms. Our algorithm can achieve superior performance and sample efficiency than state-of-the-art model-based and model-free reinforcement learning algorithms. For future work, it would be enticing to deploy the presented algorithm on a real-robotic agent.

References

[1] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *CoRR*, abs/1807.01675, 2018.

[2] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

[3] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. *CoRR*, abs/1809.05214, 2018.

[4] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[5] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

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

[8] 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*, pages 2944–2952, 2015.

[9] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *CoRR*, abs/1906.08253, 2019.

[10] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *IN PROC. 19TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING*, pages 267–274, 2002.

[11] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

[12] 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. *arXiv preprint arXiv:1509.02971*, 2015.

[13] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3):293–321, May 1992.

[14] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *ICLR*, 2019.

[15] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[16] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning, 2019.

[17] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596*, 2017.

[18] J. Peters and S. Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225, Oct 2006.

[19] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

[20] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[21] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[22] 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. *CoRR*, abs/1907.02057, 2019.

A Appendix

In this section, we provide the theoretical proofs of our methods as well as some additional experiments.

A.1 Model-Augmented Actor-Critic Objective

Among model-free methods, actor-critic methods have shown superior performance in terms of sample efficiency and asymptotic performance [6]. However, their sample efficiency remains worse than model-based approaches, and fully off-policy methods still show instabilities comparing to on-policy algorithms [15]. In this paper, we propose a modification of the Q-function parametrization by using the model predictions on the first time-steps after the action is taken. Specifically, we do policy optimization by maximizing the following objective:

$$J_\pi(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r(s_t) + \gamma^H \hat{Q}(s_H, a_H)\right]$$

whereby, $s_{t+1} \sim \hat{f}(s_t, a_t)$ and $a_t \sim \pi_{\boldsymbol{\theta}}(s_t)$. Note that under the true dynamics and Q-function, this objective is the same as the RL objective. Contrary to previous reinforcement learning methods, we optimize this objective by back-propagation through time. Since the learned dynamics model and policy are parameterized as Gaussian distributions, we can make use of the pathwise derivative estimator to compute the gradient, resulting in an objective that captures uncertainty while presenting low variance. The computational graph of the proposed objective is shown in Figure 4.

While the proposed objective resembles n-step bootstrap [20], our model usage fundamentally differs from previous approaches. First, we do not compromise between being off-policy and stability. Typically, n-step bootstrap is either on-policy, which harms the sample complexity, or its gradient estimation uses likelihood ratios, which presents large variance and results in unstable learning [8]. Second, we obtain a strong learning signal by backpropagating the gradient of the policy across multiple steps using the pathwise derivative estimator, instead of the REINFORCE estimator [16, 18]. And finally, we prevent the exploding and vanishing gradients effect inherent to back-propagation through time by the means of the terminal Q-function [11].

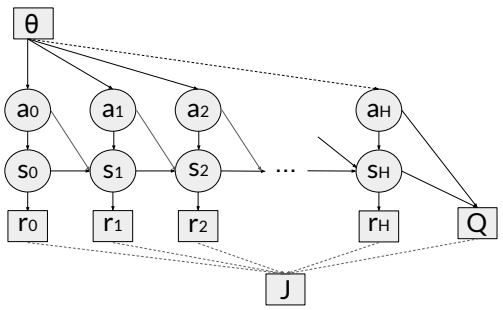

Figure 4: Stochastic computation graph of the proposed objective J_π. The stochastic nodes are represented by circles and the deterministic ones by squares.

The horizon H in our proposed objective allows us to trade off between the accuracy of our learned model and the accuracy of our learned Q-function. Hence, it controls the degree to which our algorithm is model-based or well model-free. If we were not to trust our model at all ($H = 0$), we would end up with a model-free update; for $H = \infty$, the objective results in a shooting objective. Note that we will perform policy optimization by taking derivatives of the objective, hence we require accuracy on the derivatives of the objective and not on its value. The following lemma provides a bound on the gradient error in terms of the error on the derivatives of the model, the Q-function, and the horizon H.

Lemma A.1 (Gradient Error)**.** *Let $\hat{f}$ and $\hat{Q}$ be the learned approximation of the dynamics f and Q-function Q, respectively. Assume that Q and $\hat{Q}$ have $L_q/2$-Lipschitz continuous gradient and f and $\hat{f}$ have $L_f/2$-Lipschitz continuous gradient. Let $\epsilon_f = \max_t \|\nabla \hat{f}(\hat{s}_t, \hat{a}_t) - \nabla f(s_t, a_t)\|_2$ be the error on the model derivatives and $\epsilon_Q = \|\nabla \hat{Q}(\hat{s}_H, \hat{a}_H) - \nabla Q(s_H, a_H)\|_2$ the error on the Q-function derivative. Then the error on the gradient between the learned objective and the true objective can be bounded by:*

$$\mathbb{E}\left[\|\nabla_{\boldsymbol{\theta}} J_\pi - \nabla_{\boldsymbol{\theta}} \hat{J}_\pi\|_2\right] \leq c_1(H)\epsilon_f + c_2(H)\epsilon_Q$$

Proof. Let $J_\pi(\boldsymbol{\theta})$ and $\hat{J}_\pi(\hat{\boldsymbol{\theta}})$ be the expected return of the policy $\pi_{\boldsymbol{\theta}}$ under our objective and under the RL objective, respectively. Then, we can write the MSE of the gradient as

$$\begin{aligned}&\mathbb{E}[\|\nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \hat{J}_\pi(\boldsymbol{\theta})\|_2] = \mathbb{E}[\|\nabla_{\boldsymbol{\theta}}(M - \hat{M}) + |\nabla_{\boldsymbol{\theta}}\gamma^H(Q - \hat{Q})\|_2] \\ &\leq \mathbb{E}[\|\nabla_{\boldsymbol{\theta}}(M - \hat{M})\|_2] + \mathbb{E}[\|\nabla_{\boldsymbol{\theta}}\gamma^H(Q - \hat{Q})\|_2]\end{aligned}$$

whereby, $M = \sum_{t=0}^{H} \gamma^t r(s_t)$ and $\hat{M} = \sum_{t=0}^{H} \gamma^t r(\hat{s}_t)$.

We will denote as ∇ the gradient w.r.t the inputs of network, $x_t = (s_t, a_t)$ and $\hat{x}_t = (\hat{s}_t, \hat{a}_t)$; where $\hat{a}_t \sim \pi_{\boldsymbol{\theta}}(\hat{s}_t)$. Notice that since $f\hat{f}$ and π are Lipschitz and their gradient is Lipschitz as well, we have that $\nabla_{\boldsymbol{\theta}}\hat{x}_t \leq K^t$, where K depends on the Lipschitz constants of the model and the policy. Without loss of generality, we assume that K is larger than 1. Now, we can bound the error on the Q as

$$\begin{aligned}\|\nabla_{\boldsymbol{\theta}}(Q - \hat{Q})\|_2 =& \|\nabla Q \nabla_{\boldsymbol{\theta}} x_H - \nabla\hat{Q}\nabla_{\boldsymbol{\theta}}\hat{x}_H\|_2 \\ &= \|(\nabla Q - \nabla\hat{Q})\nabla_{\boldsymbol{\theta}} x_H - \nabla\hat{Q}(\nabla_{\boldsymbol{\theta}}\hat{x}_H - \nabla_{\boldsymbol{\theta}} x_H)\|_2 \\ &\leq \|\nabla Q - \nabla\hat{Q}\|_2\|\nabla_{\boldsymbol{\theta}} x_H\|_2 + \|\nabla\hat{Q}\|_2\|\nabla_{\boldsymbol{\theta}}\hat{x}_H - \nabla_{\boldsymbol{\theta}} x_H\|_2 \\ &\leq \epsilon_Q\|\nabla_{\boldsymbol{\theta}} x_H\|_2 + L_Q\|\nabla_{\boldsymbol{\theta}}\hat{x}_H - \nabla_{\boldsymbol{\theta}} x_H\|_2 \\ &\leq \epsilon_Q K^H + L_Q\|\nabla_{\boldsymbol{\theta}}\hat{x}_H - \nabla_{\boldsymbol{\theta}} x_H\|_2\end{aligned}$$

Now, we will bound the term $\|\nabla_{\boldsymbol{\theta}}\hat{s}_{t+1} - \nabla_{\boldsymbol{\theta}} s_{t+1}\|_2$:

$$\begin{aligned}\|\nabla_{\boldsymbol{\theta}}\hat{s}_{t+1} - \nabla_{\boldsymbol{\theta}} s_{t+1}\|_2 &= \|\nabla_s\hat{f}\nabla_{\boldsymbol{\theta}}\hat{s}_t + \nabla_a\hat{f}\nabla_{\boldsymbol{\theta}}\hat{a}_t - \nabla_s f\nabla_{\boldsymbol{\theta}} s_t - \nabla_a f\nabla_{\boldsymbol{\theta}} a_t\|_2 \\ &\leq \|\nabla_s\hat{f}\nabla_{\boldsymbol{\theta}}\hat{s}_t - \nabla_s f\nabla_{\boldsymbol{\theta}} s_t\|_2 + \|\nabla_a\hat{f}\nabla_{\boldsymbol{\theta}}\hat{a}_t - \nabla_a f\nabla_{\boldsymbol{\theta}} a_t\|_2 \\ &\leq \epsilon_f\|\nabla_{\boldsymbol{\theta}}\hat{s}_t\|_2 + L_f\|\nabla_{\boldsymbol{\theta}}\hat{s}_t - \nabla_{\boldsymbol{\theta}} s_t\|_2 + L_f\|\nabla_{\boldsymbol{\theta}}\hat{a}_t - \nabla_{\boldsymbol{\theta}} a_t\| + \epsilon_f\|\nabla_{\boldsymbol{\theta}}\hat{a}_t\|_2 \\ &\leq \epsilon_f\|\nabla_{\boldsymbol{\theta}}\hat{s}_t\|_2 + (L_f + L_f L_\pi)\|\nabla_{\boldsymbol{\theta}}\hat{s}_t - \nabla_{\boldsymbol{\theta}} s_t\|_2 + \epsilon_f\|\nabla_{\boldsymbol{\theta}}\hat{a}_t \\ &= \epsilon_f\|\nabla_{\boldsymbol{\theta}}\hat{x}_t\|_2 + (L_f + L_f L_\pi)\|\nabla_{\boldsymbol{\theta}}\hat{s}_t - \nabla_{\boldsymbol{\theta}} s_t\|_2\end{aligned}$$

Hence, applying this recursion we obtain that

$$\|\nabla_{\boldsymbol{\theta}}\hat{x}_{t+1} - \nabla_{\boldsymbol{\theta}} x_{t+1}\|_2 \leq \epsilon_f \sum_{k=0}^{t}(L_f + L_f L_\pi)^{t-k}\|\nabla_{\boldsymbol{\theta}}\hat{x}_k\|_2 \leq \epsilon_f \frac{L^{t+1} - 1}{L - 1}K^t$$

where $L = L_f + L_f L_\pi$. Then, the error in the gradient in the previous term is bounded by

$$\|\nabla_{\boldsymbol{\theta}}(Q - \hat{Q})\|_2 \leq \epsilon_Q K^H + L_Q\epsilon_f \frac{L^H - 1}{L - 1}K^H$$

In order to bound the model term we need first to bound the rewards since

$$\|\nabla_{\boldsymbol{\theta}}(M - \hat{M})\|_2 \leq \sum_{t=0}^{H}\gamma^t\|\nabla_{\boldsymbol{\theta}}(r(s_t) - r(\hat{s}_t))\|_2$$

Similar to the previous bounds, we can bound now each reward term by

$$\|\nabla_{\boldsymbol{\theta}}(r(s_t) - r(\hat{s}_t))\|_2 \leq \epsilon_f L_r \frac{L^{t+1} - 1}{L - 1}K^t$$

With this result we can bound the total error in models

$$\|\nabla_{\boldsymbol{\theta}}(M - \hat{M})\|_2 \leq \sum_{t=0}^{H-1}\gamma^t\epsilon_f L_r \frac{L^{t+1} - 1}{L - 1}K^t = \frac{L\epsilon_f}{(L-1)}\left(\frac{(\gamma KL)^H - 1}{\gamma KL - 1} - \frac{(\gamma K)^H - 1}{\gamma K - 1}\right)$$

Then, the gradient error has the form

$$\begin{aligned}\mathbb{E}[\|\nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\hat{J}_\pi(\boldsymbol{\theta})\|_2] \leq \frac{L\epsilon_f}{(L-1)}\left(\frac{(\gamma KL)^H - 1}{\gamma KL - 1} - \frac{(\gamma K)^H - 1}{\gamma K - 1}\right) + \epsilon_Q(\gamma K)^H + L_Q\epsilon_f\frac{L^H - 1}{L - 1}(\gamma K)^H \\ = \epsilon_f c_1(H) + \epsilon_Q c_2(H)\end{aligned}$$

□

The result in Lemma A.1 stipulates the error of the policy gradient in terms of the maximum error in the model derivatives and the error in the Q derivatives. The functions c_1 and c_2 are functions of the horizon and depend on the Lipschitz constants of the model and the Q-function. Note that we are just interested in the relation between both sources of error, since the gradient magnitude will be scaled by the learning rate, or by the optimizer, when applying it to the weights.

A.2 Monotonic Improvement

In the previous section, we presented our objective and the error it incurs in the policy gradient with respect to approximation error in the model and the Q function. However, the error on the gradient is not indicative of the effect of the desired metric: the average return. Here, we quantify the effect of the modeling error on the return. First, we will bound the KL-divergence between the policies resulting from taking the gradient with the true objective and the approximated one. Then we will bound the performance in terms of the KL.

Lemma A.2 (Total Variation Bound)**.** *Under the assumptions of the Lemma A.1, let $\boldsymbol{\theta} = \boldsymbol{\theta}_o + \alpha \nabla_{\boldsymbol{\theta}} J_\pi$ be the parameters resulting from taking a gradient step on the exact objective, and $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_o + \alpha \nabla_{\boldsymbol{\theta}} \hat{J}_\pi$ the parameters resulting from taking a gradient step on approximated objective, where $\alpha \in \mathbb{R}^+$. Then the following bound on the total variation distance holds*

$$\max_s D_{TV}(\pi_\theta || \pi_{\hat{\theta}}) \leq \alpha c_3 (\epsilon_f c_1(H) + \epsilon_Q c_2(H))$$

Proof. The total variation distance can be bounded by the KL-divergence using the Pinsker's inequality

$$D_{TV}(\pi_{\boldsymbol{\theta}} \| \pi_{\hat{\boldsymbol{\theta}}}) \leq \sqrt{\frac{D_{KL}(\pi_{\boldsymbol{\theta}} \| \pi_{\hat{\boldsymbol{\theta}}})}{2}}$$

Then if we assume third order smoothness on our policy, by the Fisher information metric theorem then

$$D_{KL}(\pi_{\boldsymbol{\theta}} \| \pi_{\hat{\boldsymbol{\theta}}}) = \tilde{c} \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2^2 + (\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2^3)$$

Given that $\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2 = \alpha \|\nabla_{\boldsymbol{\theta}} J_\pi - \nabla_{\boldsymbol{\theta}} \hat{J}_\pi\|_2$, for a small enough step the following inequality holds

$$D_{KL}(\pi_{\boldsymbol{\theta}} \| \pi_{\hat{\boldsymbol{\theta}}}) \leq \alpha^2 \tilde{c} (\epsilon_f c_1(H) + \epsilon_Q c_2(H))^2 =$$

Combining this bound with the Pinsker inequality

$$D_{TV}(\pi_{\boldsymbol{\theta}} \| \pi_{\hat{\boldsymbol{\theta}}}) \leq \alpha \sqrt{\frac{\tilde{c}}{2}} (\epsilon_f c_1(H) + \epsilon_Q c_2(H)) = \alpha c_3 (\epsilon_f c_1(H) + \epsilon_Q c_2(H))$$

□

The previous lemma results in a bound on the distance between the policies originated from taking a gradient step using the true dynamics and Q-function, and using its learned counterparts. Now, we can derive a similar result from [10] to bound the difference in average returns.

Theorem A.1 (Monotonic Improvement)**.** *Under the assumptions of the Lemma A.1, be $\boldsymbol{\theta}'$ and $\hat{\boldsymbol{\theta}}$ as defined in Lemma A.2, and assuming that the reward is bounded by $r_{\max}$. Then the average return of the $\pi_{\hat{\boldsymbol{\theta}}}$ satisfies*

$$J_\pi(\hat{\boldsymbol{\theta}}) \geq J_\pi(\boldsymbol{\theta}) - \frac{2\alpha r_{\max}}{1-\gamma} \alpha c_3 (\epsilon_f c_1(H) + \epsilon_Q c_2(H))$$

Proof. Given the bound on the total variation distance, we can now make use of the monotonic improvement theorem to establish an improvement bound in terms of the gradient error. Let $J_\pi(\boldsymbol{\theta})$ and $J_\pi(\hat{\boldsymbol{\theta}})$ be the expected return of the policy $\pi_{\boldsymbol{\theta}}$ and $\pi_{\hat{\boldsymbol{\theta}}}$ under the true dynamics. Let ρ and $\hat{\rho}$ be

the discounted state marginal for the policy $\pi_{\boldsymbol{\theta}}$ and $\pi_{\hat{\boldsymbol{\theta}}}$, respectively

$$
\begin{aligned}
|J_\pi(\boldsymbol{\theta}) - J_\pi(\hat{\boldsymbol{\theta}})| &= |\sum_{s,a} \rho(s)\pi_{\boldsymbol{\theta}} r(s,a) - \hat{\rho}(s)\pi_{\hat{\boldsymbol{\theta}}} r(s,a)| \\
&\leq |\sum_{s,a} \rho(s)\pi_{\boldsymbol{\theta}}(a|s) r(s,a) - \hat{\rho}(s)\pi_{\hat{\boldsymbol{\theta}}}(a|s) r(s,a)| \\
&\leq r_{\max} |\sum_{s,a} \rho(s)\pi_{\boldsymbol{\theta}}(a|s) - \hat{\rho}(s)\pi_{\hat{\boldsymbol{\theta}}}(a|s)| \\
&\leq \frac{2r_{\max}}{1-\gamma} \max_s \sum_a |\pi_{\boldsymbol{\theta}}(a|s) - \pi_{\hat{\boldsymbol{\theta}}}(a|s)| \\
&= \frac{2r_{\max}}{1-\gamma} \max_s D_{TV}(\pi_{\boldsymbol{\theta}} \| \pi_{\hat{\boldsymbol{\theta}}})
\end{aligned}
$$

Then, combining the results from Lemma A.2 we obtain the desired bound. □

Hence, we can provide explicit lower bounds of improvement in terms of model error and function error. Theorem A.1 extends previous work of monotonic improvement for model-free policies [19, 10], to the model-based and actor critic set up by taking the error on the learned functions into account. From this bound one could, in principle, derive the optimal horizon H that minimizes the gradient error. However, in practice, approximation errors are hard to determine and we treat H as an extra hyper-parameter. In section 4.2, we experimentally analyze the error on the gradient for different estimators and values of H.

A.3 Ablations

In order to show the significance of each component of MAAC, we conducted more ablation studies. The results are shown in Figure 5. Here, we analyze the effect of training the Q-function with data coming from just the real environment, not learning a maximum entropy policy, and increasing the batch size instead of increasing the amount of samples to estimate the value function.

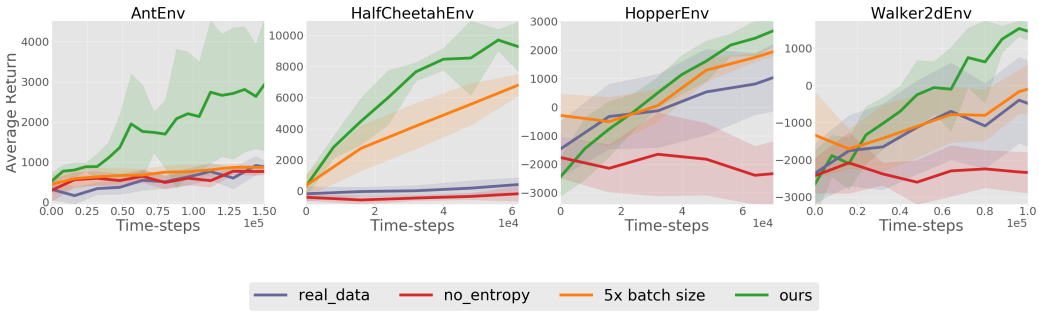

Figure 5: We further test the significance of some components of our method: not use the dynamics to generate data, and only use real data sampled from environments to train policy and Q-functions (real_data), remove entropy from optimization objects (no_entropy), and using a single sample estimate of the pathwise derivative but increase the batch size accordingly (5x batch size). Considering entropy and using dynamic models to augment data set are both very important.

A.4 Execution Time Comparison

A.5 Model Predictive Control

One of the key benefits of methods that combine model-based reinforcement learning and actor-critic methods is that the optimization procedure results in a stochastic policy, a dynamics model and a Q-function. Hence, we have all the components for, at test time, refine the action selection by the means of model predictive control (MPC). Here, we investigate the improvement in performance of planning at test time. Specifically, we use the cross-entropy method with our stochastic policy as our initial distributions. The results, shown in Table 2, show benefits in online planning in complex

	Iteration (s)	Training Model (s)	Optimization (s)	MBPO Iteration (s)
HalfCheetahEnv	1312	486	738	708
HopperEnv	845	209	517	723

Table 1: This table shows the time that different parts of MAAC need to train for one iteration after 6000 time steps, averaged across 4 seeds. We also add the time needed for MBPO for one iteration here for comparison.

	AntEnv	HalfCheetahEnv	HopperEnv	Walker2dEnv
MAAC+MPC	$3.97\text{e}3 \pm 1.48\text{e}3$	$1.09\text{e}4 \pm 94.5$	$2.8\text{e}3 \pm 11$	$1.76\text{e}3 \pm 78$
MAAC	$3.06\text{e}3 \pm 1.45\text{e}3$	$1.07\text{e}4 \pm 253$	$2.77\text{e}3 \pm 3.31$	$1.61\text{e}3 \pm 404$

Table 2: Performance at test time with (maac+mpc) and without (maac) planning of the converged policy using the MAAC objective.

domains; however, its improvement gains are more timid in easier domains, showing that the learned policy has already interiorized the optimal behaviour.