Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees

Yuping Luo [*†] **Huazhe Xu** [*‡] **Yuanzhi Li** [§] **Yuandong Tian** [¶] **Trevor Darrell** [‡] **Tengyu Ma** [§]

Abstract

Model-based reinforcement learning (RL) is considered to be a promising approach to reduce the sample complexity that hinders model-free RL. However, the theoretical understanding of such methods has been rather limited. This paper introduces a novel algorithmic framework for designing and analyzing model-based RL algorithms with theoretical guarantees. We design a meta-algorithm with a theoretical guarantee of monotone improvement to a local maximum of the expected reward. The meta-algorithm iteratively builds a lower bound of the expected reward based on the estimated dynamical model and sample trajectories, and then maximizes the lower bound jointly over the policy and the model. Instantiating our framework with simplification gives a variant of model-based RL algorithms Stochastic Lower Bounds Optimization (SLBO). Experiments demonstrate that SLBO achieves state-of-the-art performance when only one million or fewer samples are permitted on a range of continuous control benchmark tasks.

1 Introduction

In recent years deep reinforcement learning has achieved strong empirical success, including learning locomotion and manipulation skills in robotics (Levine et al., 2016; Schulman et al., 2015b) . Many of these results are achieved by model-free RL algorithms that often require a massive number of samples. Model-based deep reinforcement learning, in contrast, exploits the information from state observations explicitly — by planning with an estimated dynamical model — and is considered to be a promising approach to reduce the sample complexity.

Despite promising empirical findings, many of *theoretical* properties of model-based deep reinforcement learning are not well-understood. For example, how does the error of the estimated model affect the estimation of the value function and the planning? Can model-based RL algorithms be guaranteed to improve the policy monotonically and converge to a local maximum of the value function? Towards addressing these challenges, the main contribution of this paper is to propose a novel algorithmic framework for model-based deep RL with theoretical guarantees. Our meta-algorithm (Algorithm 1) extends the optimism-in-face-of-uncertainty principle to non-linear dynamical models in a way that requires *no explicit* uncertainty quantification of the dynamical models.

We show that the performance of the policy is guaranteed to monotonically increase, assuming the optimization within each iteration succeeds (see Theorem 2.1.) To the best of our knowledge, this is the first theoretical guarantee of monotone improvement for model-based deep RL.

Finally, inspired by our framework and analysis, we design a variant of model-based RL algorithms Stochastic Lower Bounds Optimization (SLBO). Experiments demonstrate that SLBO achieves

[*]equal contribution

[†]Princeton University, yupingl@cs.princeton.edu

[‡]UC Berkeley. huazhe_xu@eecs.berkeley.edu and trevor@eecs.berkeley.edu

[§]Stanford University. yuanzhili92@gmail.com and tengyuma@stanford.edu

[¶]Facebook AI Research. yuandong@fb.com

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

state-of-the-art performance when only 1M samples are permitted on a range of continuous control benchmark tasks.

2 Algorithmic Framework

As mentioned in the introduction, towards optimizing $V^{\pi,M^\star}$,[6] our plan is to build a lower bound for $V^{\pi,M^\star}$ of the following type and optimize it iteratively:

$$V^{\pi,M^\star} \geq V^{\pi,\widehat{M}} - D(\widehat{M},\pi) \tag{2.1}$$

where $D(\widehat{M},\pi) \in \mathbb{R}_{\geq 0}$ bounds from above the discrepancy between $V^{\pi,\widehat{M}}$ and $V^{\pi,M^\star}$. Building such an optimizable discrepancy bound globally that holds for all $\widehat{M}$ and π turns out to be rather difficult, if not impossible. Instead, we shoot for establishing such a bound over the neighborhood of a reference policy π_{ref}.

$$V^{\pi,M^\star} \geq V^{\pi,\widehat{M}} - D_{\pi_{\text{ref}},\delta}(\widehat{M},\pi), \qquad \forall \pi \text{ s.t. } d(\pi,\pi_{\text{ref}}) \leq \delta \tag{R1}$$

Here $d(\cdot,\cdot)$ is a function that measures the closeness of two policies, which will be chosen later in alignment with the choice of D. We will mostly omit the subscript δ in D for simplicity in the rest of the paper. We will require our discrepancy bound to vanish when $\widehat{M}$ is an accurate model:

$$\widehat{M} = M^\star \Longrightarrow D_{\pi_{\text{ref}}}(\widehat{M},\pi) = 0, \quad \forall \pi, \pi_{\text{ref}} \tag{R2}$$

The third requirement for the discrepancy bound D is that it can be estimated and optimized in the sense that

$$D_{\pi_{\text{ref}}}(\widehat{M},\pi) \text{ is of the form } \mathop{\mathbb{E}}_{\tau\sim\pi_{\text{ref}},M^\star}[f(\widehat{M},\pi,\tau)] \tag{R3}$$

where f is a known differentiable function. We can estimate such discrepancy bounds for *every* π in the neighborhood of π_{ref} by sampling empirical trajectories $\tau^{(1)},\ldots,\tau^{(n)}$ from executing policy π_{ref} on the real environment $M^\star$ and compute the average of $f(\widehat{M},\pi,\tau^{(i)})$'s. We would have to insist that the expectation cannot be over the randomness of trajectories from π on $M^\star$, because then we would have to re-sample trajectories for every possible π encountered.

Suppose we can establish such an discrepancy bound D (and the distance function d) with properties (R1), (R2), and (R3) then we can devise the following meta-algorithm (Algorithm 1).

Algorithm 1 Meta-Algorithm for Model-based RL

Inputs: Initial policy π_0. Discrepancy bound D and distance function d that satisfy equation (R1) and (R2).
For $k = 0$ to T:

$$\pi_{k+1}, M_{k+1} = \mathop{\text{argmax}}_{\pi\in\Pi,\ M\in\mathcal{M}} \quad V^{\pi,M} - D_{\pi_k,\delta}(M,\pi) \tag{2.2}$$

$$\text{s.t. } d(\pi,\pi_k) \leq \delta \tag{2.3}$$

We remark that the discrepancy bound $D_{\pi_k}(M,\pi)$ in the objective plays the role of learning the dynamical model by ensuring the model to fit to the sampled trajectories.

Our main theorem, proof of which can be found at Appendix B, shows formally that the policy performance in the real environment is non-decreasing under the assumption that the real dynamics belongs to our parameterized family $\mathcal{M}$.

Theorem 2.1. *Suppose that $M^\star \in \mathcal{M}$, that D and d satisfy equation* (R1) *and* (R2)*, and the optimization problem in equation* (2.2) *is solvable at each iteration. Then, Algorithm 1 produces a sequence of policies $\pi_0,\ldots,\pi_T$ with monotonically increasing values:*

$$V^{\pi_0,M^\star} \leq V^{\pi_1,M^\star} \leq \cdots \leq V^{\pi_T,M^\star} \tag{2.4}$$

Moreover, as $k\to\infty$, the value $V^{\pi_k,M^\star}$ converges to some $V^{\bar{\pi},M^\star}$, where $\bar{\pi}$ is a local maximum of $V^{\pi,M^\star}$ in domain Π.

[6] Note that in the introduction we used V^π for simplicity, and in the rest of the paper we will make the dependency on $M^\star$ explicit. All the notations and preliminaries are presented in the appendix.

3 Practical Implementation and Experiments

3.1 Practical implementation

We design with simplification of our framework a variant of model-based RL algorithms, Stochastic Lower Bound Optimization (SLBO). First, we removed the constraints (2.3). Second, we stop the gradient w.r.t M (but not π) from the occurrence of M in $V^{\pi,M}$ in equation (2.2) (and thus our practical implementation is not optimism-driven.)

We use a multi-step prediction loss for learning the models with ℓ_2 norm. For a state s_t and action sequence $a_{t:t+h}$, we define the h-step prediction $\hat{s}_{t+h}$ as $\hat{s}_t = s_t$, and for $h \geq 0$, $\hat{s}_{t+h+1} = \widehat{M}_\phi(\hat{s}_{t+h}, a_{t+h})$, The *$H$-step loss* is then defined as

$$\mathcal{L}_\phi^{(H)}((s_{t:t+h}, a_{t:t+h}); \phi) = \frac{1}{H}\sum_{i=1}^{H} \|(\hat{s}_{t+i} - \hat{s}_{t+i-1}) - (s_{t+i} - s_{t+i-1})\|_2. \tag{3.1}$$

We note that the term $V^{\pi_\theta, \mathrm{sg}(\widehat{M}_\phi)}$ depends on both the parameter θ and the parameter ϕ but there is no gradient passed through ϕ, whereas $\mathcal{L}_\phi^{(H)}$ only depends on the ϕ. We optimize equation (2.2) by alternatively maximizing $V^{\pi_\theta, \mathrm{sg}(\widehat{M}_\phi)}$ and minimizing $\mathcal{L}_\phi^{(H)}$: for the former, we use TRPO with samples from the estimated dynamical model $\widehat{M}_\phi$ (by treating $\widehat{M}_\phi$ as a fixed simulator), and for the latter we use standard stochastic gradient methods. Algorithm 2 gives a pseudo-code for the algorithm. The n_{model} and n_{policy} iterations are used to balance the number of steps of TRPO and Adam updates within the loop indexed by n_{inner}.

Algorithm 2 Stochastic Lower Bound Optimization (SLBO)

1: Initialize model network parameters ϕ and policy network parameters θ
2: Initialize dataset $\mathcal{D} \leftarrow \emptyset$
3: **for** n_{outer} iterations **do**
4: $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{$ collect n_{collect} samples from real environment using π_θ with noises $\}$
5: $\quad$ **for** n_{inner} iterations **do** $\quad \triangleright$ optimize (2.2) with stochastic alternating updates
6: $\quad\quad$ **for** n_{model} iterations **do**
7: $\quad\quad\quad$ optimize (3.1) over ϕ with sampled data from $\mathcal{D}$ by one step of Adam
8: $\quad\quad$ **for** n_{policy} iterations **do**
9: $\quad\quad\quad \mathcal{D}' \leftarrow \{$ collect n_{trpo} samples using $\widehat{M}_\phi$ as dynamics $\}$
10: $\quad\quad\quad$ optimize π_θ by running TRPO on $\mathcal{D}'$

Power of stochasticity and connection to standard MB RL: We identify the main advantage of our algorithms over standard model-based RL algorithms is that we alternate the updates of the model and the policy within an outer iteration. By contrast, most of the existing model-based RL methods only optimize the models once (for a lot of steps) after collecting a batch of samples (see Algorithm 3 for an example). The stochasticity introduced from the alternation with stochastic samples seems to dramatically reduce the overfitting (of the policy to the estimated dynamical model) in a way similar to that SGD regularizes ordinary supervised training. Another way to view the algorithm is that the model obtained from line 7 of Algorithm 2 at different inner iteration serves as an ensemble of models. We do believe that a cleaner and easier instantiation of our framework (with optimism) exists, and the current version, though performing very well, is not necessarily the best implementation.

3.2 Experimental Results

We evaluate our algorithm SLBO (Algorithm 2) on five continuous control tasks from rllab (Duan et al., 2016), including Swimmer, Half Cheetah, Humanoid, Ant, Walker. All environments that we test have a maximum horizon of 500, while environments with longer horizons are commonly harder to train. More details can be found in Appendix C.1.

Baselines. We compare our algorithm with 3 other algorithms including: (1) Soft Actor-Critic (SAC) (Haarnoja et al., 2018), the state-of-the-art model-free off-policy algorithm in sample efficiency; (2) Trust-Region Policy Optimization (TRPO) (Schulman et al., 2015a), a policy-gradient based

algorithm; and (3) Model-Based TRPO, a standard model-based algorithm described in Algorithm 3. Details of these algorithms can be found in Appendix C.4.

The result is shown in Figure 1. In Fig 1, our algorithm shows superior convergence rate (in number of samples) than all the baseline algorithms while achieving better final performance with 1M samples. Specifically, we mark model-free TRPO performance after 8 million steps by the dotted line in Fig 1 and find out that our algorithm can achieve comparable or better final performance in one million steps. We also study the performance of SLBO and baselines with 4 million training samples in C.5. Ablation study of multi-step model training can be found in Appendix C.5.[7]

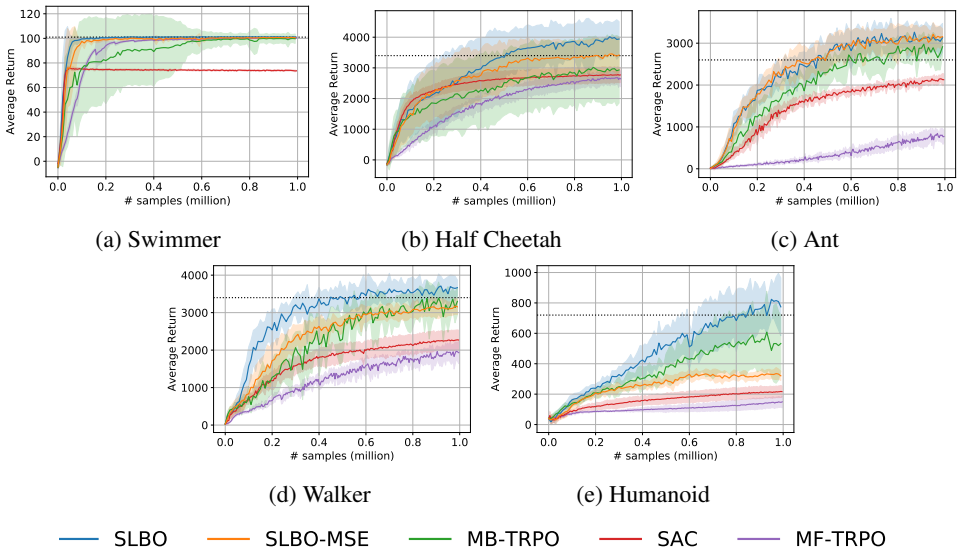


Figure 1: Comparison between SLBO (ours), SLBO with squared ℓ^2 model loss (SLBO-MSE), vanilla model-based TRPO (MB-TRPO), model-free TRPO (MF-TRPO), and Soft Actor-Critic (SAC). We average the results over 10 different random seeds, where the solid lines indicate the mean and shaded areas indicate one standard deviation. The dotted reference lines are the total rewards of MF-TRPO after 8 million steps.

References

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016.

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.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

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*, pp. 1928–1937, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

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*, pp. 5026–5033. IEEE, 2012.

[7] Videos demonstrations are available at https://sites.google.com/view/algombrl/home.

A Notations and Preliminaries

We denote the state space by $\mathcal{S}$, the action space by $\mathcal{A}$. A policy $\pi(\cdot|s)$ specifies the conditional distribution over the action space given a state s. A dynamical model $M(\cdot|s,a)$ specifies the conditional distribution of the next state given the current state s and action a. We will use $M^\star$ globally to denote the unknown true dynamical model. Our target applications are problems with the continuous state and action space, although the results apply to discrete state or action space as well. When the model is deterministic, $M(\cdot|s,a)$ is a dirac measure. In this case, we use $M(s,a)$ to denote the unique value of s' and view M as a function from $\mathcal{S}\times\mathcal{A}$ to $\mathcal{S}$. Let $\mathcal{M}$ denote a (parameterized) family of models that we are interested in, and Π denote a (parameterized) family of policies.

Unless otherwise stated, for random variable X, we will use p_X to denote its density function.

Let S_0 be the random variable for the initial state. Let $S_t^{\pi,M}$ to denote the random variable of the states at steps t when we execute the policy π on the dynamic model M stating with S_0. Note that $S_0^{\pi,M}=S_0$ unless otherwise stated. We will omit the subscript when it's clear from the context. We use A_t to denote the actions at step t similarly. We often use τ to denote the random variable for the trajectory $(S_0,A_1,\dots,S_t,A_t,\dots)$. Let $R(s,a)$ be the reward function at each step. We assume R is *known* throughout the paper, although R can be also considered as part of the model if unknown. Let γ be the discount factor.

Let $V^{\pi,M}$ be the value function on the model M and policy π defined as:

$$V^{\pi,M}(s)=\mathop{\mathbb{E}}_{\substack{\forall t\geq 0, A_t\sim\pi(\cdot|S_t)\\ S_{t+1}\sim M(\cdot|S_t,A_t)}}\left[\sum_{t=0}^{\infty}\gamma^t R(S_t,A_t)\mid S_0=s\right] \tag{A.1}$$

We define $V^{\pi,M}=\mathbb{E}\left[V^{\pi,M}(S_0)\right]$ as the expected reward-to-go at Step 0 (averaged over the random initial states). Our goal is to maximize the reward-to-go on the true dynamical model, that is, $V^{\pi,M^\star}$, over the policy π. For simplicity, throughout the paper, we set $\kappa=\gamma(1-\gamma)^{-1}$ since it occurs frequently in our equations. Every policy π induces a distribution of states visited by policy π:

Definition A.1. For a policy π, define $\rho^{\pi,M}$ as the discounted distribution of the states visited by π on M. Let ρ^π be a shorthand for $\rho^{\pi,M^\star}$ and we omit the superscript $M^\star$ throughout the paper. Concretely,we have $\rho^\pi=(1-\gamma)\sum_{t=0}^{\infty}\gamma^t\cdot p_{S_t^\pi}$

B Main Theorem

Proof of Theorem 2.1. Since D and d satisfy equation (R1), we have that

$$V^{\pi_{k+1},M^\star}\geq V^{\pi_{k+1},M_{k+1}}-D_{\pi_k}(M_{k+1},\pi_{k+1})$$

By the definition that π_{k+1} and M_{k+1} are the optimizers of equation (2.2), we have that

$$V^{\pi_{k+1},M_{k+1}}-D_{\pi_k}(M_{k+1},\pi_{k+1})\geq V^{\pi_k,M^\star}-D_{\pi_k}(M^\star,\pi_k)=V^{\pi_k,M^\star}\qquad\text{(by equation R2)}$$

Combing the two equations above we complete the proof of equation (2.4).

For the second part of the theorem, by compactness, we have that a subsequence of π_k converges to some $\bar{\pi}$. By the monotonicity we have $V^{\pi_k,M^\star}\leq V^{\bar{\pi},M^\star}$ for every $k\geq 0$. For the sake of contradiction, we assume $\bar{\pi}$ is a not a local maximum, then in the neighborhood of $\bar{\pi}$ there exists π' such that $V^{\pi',M^\star}>V^{\bar{\pi},M^\star}$ and $d(\bar{\pi},\pi')<\delta/2$. Let t be such that π_t is in the $\delta/2$-neighborhood of $\bar{\pi}$. Then we see that $(\pi',M^\star)$ is a better solution than (π_{t+1},M_{t+1}) for the optimization problem (2.2) in iteration t because $V^{\pi',M^\star}>V^{\bar{\pi},M^\star}\geq V^{\pi_{t+1},M^\star}\geq V^{\pi_{t+1},M_{t+1}}-D_{\pi_t}(M_{t+1},\pi_{t+1})$. (Here the last inequality uses equation (R1) with π_t as π_{ref}.) The fact $(\pi',M^\star)$ is a strictly better solution than (π_{t+1},M_{t+1}) contradicts the fact that (π_{t+1},M_{t+1}) is defined to be the optimal solution of (2.2) . Therefore $\bar{\pi}$ is a local maximum and we complete the proof.

□

C Implementation Details

C.1 Environment Setup

We benchmark our algorithm on six tasks based on physics simulator Mujoco (Todorov et al., 2012). We use rllab's implementation (Duan et al., 2016) [8] to interact with Mujoco. All the environments we use have a maximum horizon of 500 steps. We remove all contact information from observation. To compute reward from states, we put the velocity of center of mass into the states.

C.2 Network Architecture and Model Learning

We use the same reward function as in rllab, except that all the coefficients C_{contact} in front of the contact force s are set to 0 in our case. We refer the readers to (Duan et al., 2016) Supp Material 1.2 for more details. All actions are projected to the action space by clipping. We normalize all observations by $s' = \frac{s-\mu}{\sigma}$ where $\mu, \sigma \in \mathbb{R}^{d_{\text{observation}}}$ are computed from all observations we collect from the real environment. Note that μ, σ may change as we collect new data. Our policy will always produce an action a in $[-1, 1]^{d_{\text{action}}}$ and the action a', which is fed into the environment, is scaled linearly by $a' = \frac{1-a}{2} a_{\text{min}} + \frac{1+a}{2} a_{\text{max}}$, where $a_{\text{min}}, a_{\text{max}}$ are the min or max values allowed at each entry.

C.3 SLBO Details

The dynamical model is represented by a feed-forward neural network with two hidden layers, each of which contains 500 hidden units. The activation function at each layer is ReLU. We use Adam to optimize the loss function with learning rate 10^{-3} and L_2 regularization 10^{-5}. The network does not predict the next state directly; instead, it predicts the normalized difference of $s_{t+1} - s_t$. The normalization scheme and statistics are the same as those of observations: We maintain μ, σ from collected data in the real environment and may change them as we collect more, and the normalized difference is $\frac{s_{t+1}-s_t-\sigma}{\mu}$.

The policy network is a feed-forward network with two hidden layers, each of which contains 32 hidden units. The policy network uses $\tanh$ as activation function and outputs a Gaussian distribution $\mathcal{N}(\mu(s), \sigma^2)$ where σ a state-independent trainable vector.

During our evaluation, we use $H = 2$ for multi-step model training and the batch size is given by $\frac{256}{H} = 128$, i.e., we enforce the model to see 256 transitions at each batch.

We run our algorithm $n_{\text{outer}} = 100$ iterations. We collect $n_{\text{train}} = 10000$ steps of real samples from the environment at the start of each iteration using current policy with Ornstein-Uhlunbeck noise (with parameter $\theta = 0.15, \sigma = 0.3$) for better exploration. At each iteration, we optimize dynamics model and policy alternatively for $n_{\text{inner}} = 20$ times. At each iteration, we optimize dynamics model for $n_{\text{model}} = 100$ times and optimize policy for $n_{\text{policy}} = 40$ times.

C.4 Baselines

TRPO. TRPO hyperparameters are listed at Table 1, which are the same as OpenAI Baselines' implementation. These hyperparameters are fixed for all experiments where TRPO is used, including ours, MB-TRPO and MF-TRPO. We do not tune these hyperparameters. We also normalize observations as our algorithm and OpenAI Baselines do.

We use a neural network as the value function to reduce variance, which has 2 hidden layers of units 64 and uses $\tanh$ as activation functions. We use Generalized Advantage Estimator (GAE) Schulman et al. (2015b) to estimate advantages. Both TRPO used in our algorithm and that in model-free algorithm share the same set of hyperparameters.

SAC. For fair comparison, we do not use a large policy network (2 hidden layers, one of which has 256 hidden units) as the authors suggest, but use exactly the same policy network as ours. All other hyperparameters are kept the same as the authors'. Note that Q network and value network have 256

[8] commit b3a2899 in `https://github.com/rll/rllab/`

Table 1: TRPO Hyperparameters.

Hyperparameters	Values
batch size	4000
max KL divergence	0.01
discount γ	0.99
GAE λ	0.95
CG iterations	10
CG damping	0.1

hidden units at each hidden layers, which is more than TRPO's. We refer the readers to Haarnoja et al. (2018) Appendix D for more details.

MB-TRPO. Model-Based TRPO (MB-TRPO) is similar to our algorithm SLBO but does not optimize model and policy alternatively during one iteration. We do not tune the hyperparameter n_{model} since any number beyond a certain threshold would bring similar results. For n_{policy} we try $\{100, 200, 400, 800\}$ on Ant and find $n_{\text{policy}} = 200$ works best. As suggested by Section C.5, we use 0.005 as the coefficient of entropy bonus for all experiments.

Algorithm 3 Model-Based Trust Region Policy Optimization (MB-TRPO)

initialize model network parameters ϕ and policy network parameters θ
initialize dataset $\mathcal{D} \leftarrow \emptyset$
for n_{outer} iterations **do**
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{$ collect n_{collect} samples from real environment using π_θ with noises $\}$
    for n_{model} iterations **do**
        optimize (3.1) over ϕ with sampled data from $\mathcal{D}$ by one step of Adam
    for n_{policy} iterations **do**
        $\mathcal{D}' \leftarrow \{$ collect n_{trpo} samples using $\widehat{M}_\phi$ as dynamics $\}$
        optimize π_θ by running TRPO on $\mathcal{D}'$

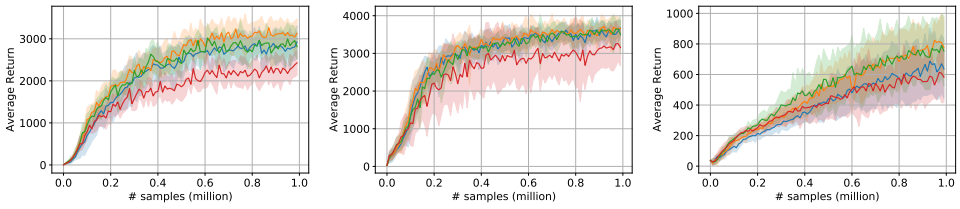
SLBO. We tune multi-step model training parameter $H \in \{1, 2, 4, 8\}$, entropy regularization coefficient $\lambda \in \{0, 0.001, 0.003, 0.005\}$ and $n_{\text{policy}} \in \{10, 20, 40\}$ on Ant and find $H = 2, \lambda = 0.005, n_{\text{policy}} = 40$ work best, then we fix them in all environments, though environment-specific hyperparameters may work better. The other hyperparameters, including $n_{\text{inner}}, n_{\text{model}}$ and network architecture, are never tuned. We observe that at the first several iterations, the policy overfits to the learnt model so a reduction of n_{policy} at the beginning can further speed up convergence but we omit this for simplicity.

C.5 Ablation Study

Multi-step model training. We compare multi-step model training with single-step model training and the results are shown on Figure 2. Note that $H = 1$ means we use single-step model training. We observe that small H (e.g., 2 or 4) can be beneficial, but larger H (e.g., 8) can hurt. We hypothesize that smaller H can help the model learn the uncertainty in the input and address the error-propagation issue to some extent.

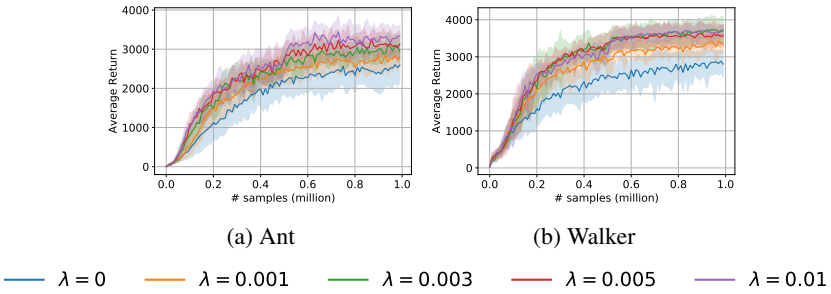
Entropy regularization. An additional component we apply to SLBO is the commonly-adopted entropy regularization in policy gradient method (Mnih et al., 2016), which was found to significantly boost the performance in our experiments (ablation study in Appendix C.5). Specifically, an additional entropy term is added to the objective function in TRPO. We hypothesize that entropy bonus helps exploration, diversifies the collected data, and thus prevents overfitting.

Figure 3 shows that entropy reguarization can improve both sample efficiency and final performance. More entropy regularization leads to better sample efficiency and higher total rewards. We observe that in the late iterations of training, entropy regularization may hurt the performance thus we stop using entropy regularization in the second half of training.



(a) Ant (b) Walker (c) Humanoid

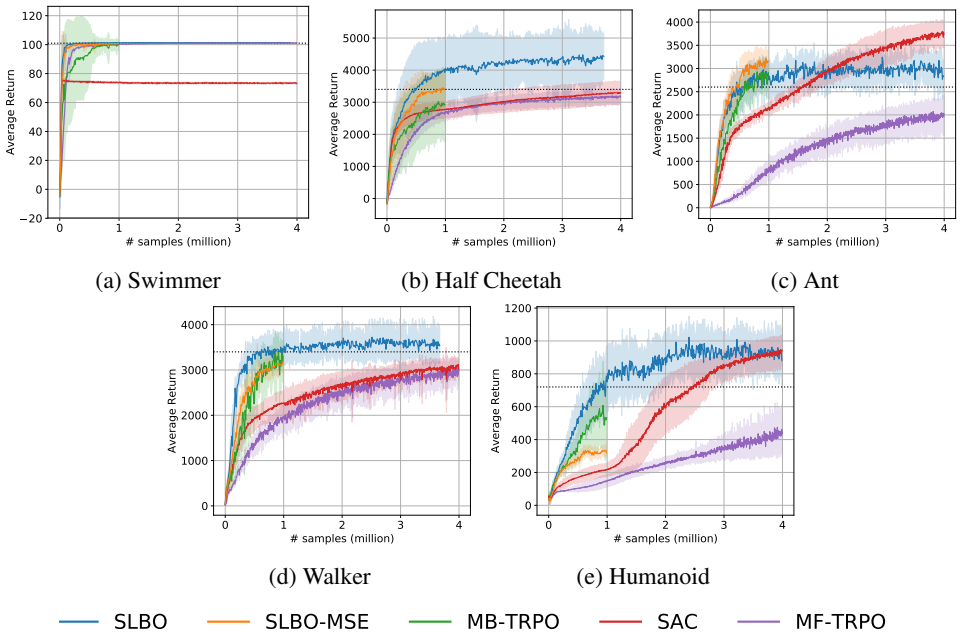
Figure 2: Ablation study on multi-step model training. All the experiments are average over 10 random seeds. The x-axis shows the total amount of real samples from the environment. The y-axis shows the averaged return from execution of our learned policy. The solid line is the mean of the total rewards from each seed. The shaded area is one-standard deviation.



(a) Ant (b) Walker

Figure 3: Ablation study on entropy regularization. λ is the coefficient of entropy regularization in the TRPO's objective. All the experiments are averaged over 10 random seeds. The x-axis shows the total amount of real samples from the environment. The y-axis shows the averaged return from execution of our learned policy. The solid line is the mean of the total rewards from each seed. The shaded area is one-standard deviation.

SLBO with 4M training steps. Figure 4 shows that SLBO is superior to SAC and MF-TRPO in Swimmer, Half Cheetah, Walker and Humanoid when 4 million samples or fewer samples are allowed. For Ant environment , although SLBO with less than one million samples reaches the performance of MF-TRPO with 8 million samples, SAC's performance surpasses SLBO after 2 million steps of training. Since model-free TRPO almost stops improving after 8M steps and our algorithms uses TRPO for optimizing the estimated environment, we don't expect SLBO can significantly outperform the reward of TRPO at 8M steps. The result shows that SLBO is also satisfactory in terms of asymptotic convergence (compared to TRPO.) It also indicates a better planner or policy optimizer instead of TRPO might be necessary to further improve the performance.



(a) Swimmer (b) Half Cheetah (c) Ant

(d) Walker (e) Humanoid

Figure 4: Comparison among SLBO (ours), SLBO with squared ℓ^2 model loss (SLBO-MSE), vanilla model-based TRPO (MB-TRPO), model-free TRPO (MF-TRPO), and Soft Actor-Critic (SAC) with more samples than in Figure 1. SLBO, SAC, MF-TRPO are trained with 4 million real samples. We average the results over 10 different random seeds, where the solid lines indicate the mean and shaded areas indicate one standard deviation. The dotted reference lines are the total rewards of MF-TRPO after 8 million steps.