TRAINING DYNAMICS MODELS FOR ACCURATE LONG-HORIZON PREDICTION

Ethan Knight
OpenAI, The Nueva School
ehk@openai.com

Joshua Achiam
OpenAI, UC Berkeley
jachiam@openai.com

ABSTRACT

Reinforcement learning agents can benefit from access to a model of the environment dynamics, for example by planning or generating simulated experience. For many tasks of interest, dynamics models are not known a priori and so must be learned from experience. Current methods for learning dynamics models usually minimize prediction losses covering one or a few time steps; however, these training procedures do not enforce the quality of long-horizon predictions, which limits the scope of model-based methods to short-horizon tasks. We propose instead to train dynamics models with an objective function based on the infinite horizon discounted sum of per-timestep model losses: the total discounted model loss (TDML). Computing this objective poses a challenge because in a naive implementation of a multistep prediction loss, the computation cost scales linearly with the horizon; we propose a method for approximating the TDML in constant time using a model learned by temporal difference methods. We evaluate our approach by comparing the long-horizon prediction accuracy of dynamics models trained using either our TDML approximator or a one-step prediction loss function, and in the preliminary results we report here, we find that models trained with TDML are better at predicting the far future.

1 INTRODUCTION

In model-based reinforcement learning (MBRL), an agent uses a model of its environment dynamics to improve task performance. Models can be used to generate simulated experience (Feinberg et al., 2018; Kaiser et al., 2019), to plan (Silver et al., 2016; Nagabandi et al., 2018), to provide side information for an agent (Weber et al., 2017), or to enforce safety requirements (Dalal et al., 2018; Berkenkamp et al., 2017). When an exact model of the environment is not given a priori, an approximate model of the dynamics must be learned.

Applying approximate models to long-horizon tasks is difficult, because model predictions can diverge from reality due to accumulating errors. For this reason, practitioners have sometimes found it necessary to use prediction horizons that are shorter than the length of the full task (Nagabandi et al., 2018; Kurutach et al., 2018; Kaiser et al., 2019), but this limits the applicability of those methods to cases where short-term optimal behavior correlates well with long-term optimal behavior.

We argue that the difficulty of applying MBRL to long-horizon tasks stems from a misalignment between how dynamics models are trained and how they are used. It is common to train dynamics models with a 1- or n-step prediction loss, where n is small, even when using the models to predict many timesteps into the future. Ideally, we would directly train dynamics models to accurately forward-simulate through the far future, but the natural approach—of constructing an n-step loss function where n is large—is computationally intractable, as the number of forward and backward passes scales with n. To make progress on this problem, we make three contributions:

1. we propose to use a new objective for model-learning: the total discounted model loss (TDML), which accounts for model accuracy over the infinite horizon;
2. we propose a temporal difference algorithm for efficiently approximating TDML in constant time;

3. we demonstrate how the TDML approximator can be used to train accurate dynamics models that perform well at predicting the far future.

To evaluate the effectiveness of our model training approach, we compare the accuracy of long horizon predictions for equivalent models trained by optimizing either the approximate TDML loss or a 1-step loss, as measured by true TDML, for three different model architectures from the model-based RL literature. In our preliminary results, we find that models trained with the approximate TDML loss are more accurate than their 1-step counterparts over long time horizons.

2 PRELIMINARIES

We consider a setting where a deterministic agent π acts in a deterministic environment f, with actions $a_t = \pi(s_t)$ and state transitions $s_{t+1} = f(s_t, a_t)$. Here, states and actions belong to the sets S, A respectively. Although we use determinism to simplify notation, the mathematical framework we will develop is general and extends straightforwardly to stochastic agents and environments.

We aim to learn a model of the transition function, $f_\theta : S \times A \to S$, parameterized by θ, which is a good approximator for f. For convenience, we additionally define the notation $\bar{f}_\theta(s) = f_\theta(s, \pi(s))$ and $\bar{f}(s) = f(s, \pi(s))$. Also let $\bar{f}_\theta{}^n(s)$ or $\bar{f}^n(s)$ denote n successive compositions of $\bar{f}_\theta$ or $\bar{f}$ respectively, which correspond to n steps of rollout in the simulated and real environments.

3 MODEL TRAINING OBJECTIVES

3.1 LOSSES IN THE LITERATURE

The standard one-step objective for model learning, L_1, is defined as:

$$L_1 = \mathop{\mathbb{E}}_{(s_t, a_t, s_{t+1}) \sim \beta} [l(f_\theta(s_t, a_t), s_{t+1})] \tag{1}$$

where β is a replay buffer, and $l : S \times S \to \mathbb{R}_+$is a per-timestep loss function. This objective with $l(x, y) = \|x - y\|_2^2$ is used in Kurutach et al. (2018); Feinberg et al. (2018); Buckman et al. (2018); Nagabandi et al. (2018) among others, and with $l(x, y) = \|x - y\|_2$ in Xu et al. (2018); Clavera et al. (2018). A multi-step variant, called multi-step or n-step loss, was used to train models in Xu et al. (2018) and as a validation metric in Nagabandi et al. (2018):

$$L_n = \mathop{\mathbb{E}}_{(s_{t:t+n}, a_{t:t+n}) \sim \beta} \left[\sum_{i=0}^{n} l(\hat{s}_{t+i}, s_{t+i}) \right], \qquad \hat{s}_{t+i+1} \leftarrow f_\theta(\hat{s}_{t+i}, a_{t+i}), \qquad \hat{s}_t = s_t, \tag{2}$$

where n is typically small (e.g. $n = 2$ in Xu et al. (2018)). Using these losses is common in the literature, even for domains with long horizons (e.g. MuJoCo and Atari). However, we argue that these losses, which we call *few-step objectives*, do not adequately incorporate important details about how models behave when used to forward simulate through the task horizon. We identify two issues in particular that we seek to address:

Distribution mismatch. The inputs shown to the model at training time differ significantly from inputs shown to the model at inference time. At training time, the model is shown trajectories from the real environment; at inference time, the model is shown its own previous predictions as state inputs, and the agent generates actions conditioned on those predictions instead of real states. This induces compounding errors similar to those observed by Bengio et al. (2015) in RNNs, where model prediction errors drive the inference inputs further from the distribution where the model is good. While the multi-step loss ostensibly mitigates this, it still relies on action sequences that were generated by an agent that only sees real states; we refer to this as an "open-loop" loss. Contrast this with a closed-loop variant, where the simulated state sequences are generated by $\hat{s}_{t+i+1} \leftarrow f_\theta(\hat{s}_{t+i}, \pi(\hat{s}_{t+1}))$. We illustrate the distinction in Figure 1.

Component weighting. Standard loss functions like mean-squared-error weight all elements in the loss vector evenly, despite the fact that not all elements will be equally important for accurately forward simulating with the model. This makes the outcome from training with few-step model losses highly dependent on the state representation: if it is easier to learn how to predict irrelevant

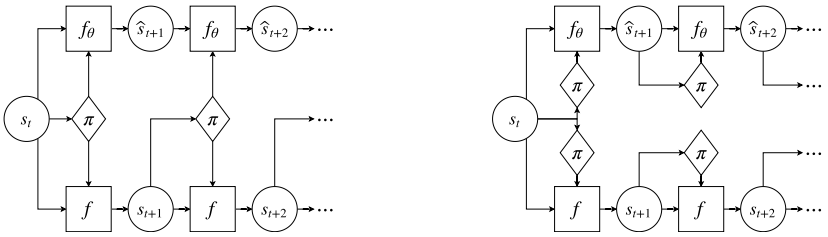

Figure 1: Computation graphs for open-loop n-step loss (left) and closed-loop n-step loss (right)

elements than important ones, the model may perform particularly poorly at inference time. We note that standard tricks like normalizing observations or state deltas, while helpful, do not fully solve the problem.

3.2 Total Discounted Model Loss

To avoid the distribution mismatch and component weighting issues, we would like a loss function that 1) accounts for closed-loop model behavior, and 2) extends as far as possible into the future. To meet these criteria, we propose an objective function based on a per-timestep loss function that directly extends the multistep closed-loop loss into the infinite horizon via a discount factor $\gamma \in (0, 1)$. We call this per-timestep loss the *total discounted model loss*:

$$D^{f,f_\theta}(s_t, \hat{s}_t) = \sum_{n=0}^{\infty} \gamma^n l(\bar{f}^n(s_t), \bar{f_\theta}^n(\hat{s}_t)). \tag{3}$$

This loss addresses both distribution alignment and component weighting. Since the loss is both closed-loop and spans the entire task length, there is no distribution mismatch between training and inference; additionally, while there is no guarantee that l leads to the best overall result for whatever problem is being addressed, TDML rectifies the issue of component weighting by taking into account far-future states. As long as l is a sensible distance measure, the model is encouraged to have the appropriate behavior.

4 Efficiently Approximating the TDML

Directly evaluating TDML as expressed in Equation 3 is not feasible because the calculation scales with the horizon, which is infinite. However, we can devise a computable approximation for TDML that does not require truncating at an arbitrary fixed horizon, through the use of temporal difference methods. Observe that we can write a Bellman equation for the TDML, as:

$$D^{f,f_\theta}(s_t, \hat{s}_t) = l(s_t, \hat{s}_t) + \gamma D^{f,f_\theta}(\bar{f}(s_t), \bar{f_\theta}(\hat{s}_t)). \tag{4}$$

With the TDML expressed in this form, all of the techniques for deep temporal difference learning (like in DDPG Lillicrap et al. (2015)) can be brought to bear in learning an approximator for it. We denote the approximator, parameterized by ψ, as D_ψ. The basic template for learning the TDML approximator is then:

$$\arg\min_{\psi} \mathop{\mathbb{E}}_{s,\hat{s}\sim\rho} \left[(D_\psi(s, \hat{s}) - y)^2 \right] \tag{5}$$

where

$$y = l(s, \hat{s}) + \gamma D_\psi(\bar{f}(s), \bar{f_\theta}(\hat{s})). \tag{6}$$

However, it is not immediately obvious what distribution ρ should be used to train D_ψ. In order for the backup term y to have accurate values, ρ needs to cover $\bar{f}(s), \bar{f_\theta}(\hat{s})$ for likely $s, \hat{s}$. That is, ρ has to include state-fictitious state pairs for entire "parallel" trajectories. As a result, we generate a distribution over $(s, \hat{s})$ pairs with the following procedure:

- Pick a state s_i in the replay buffer.

Algorithm 1 Model training with Approximate TDML

Given model loss function $l(\cdot, \cdot)$, max rollout length κ, TDML discount γ, model target averaging coefficient τ_f, TDML target averaging coefficient τ_M, model update delay d
Given deterministic policy π, replay buffer β
Initialize TDML network D_ψ, model f_θ and target networks $D_{\psi'} \leftarrow D_\psi$, $f_{\theta'} \leftarrow f_\theta$
for $t = 1$ **to** T **do**
  Sample n trajectories of states τ from β
  Sample integers $k \sim [1, \kappa]$, $i \sim [0, H - k)$
  $y \leftarrow l(s_{i+k}, \bar{f}^k_{\theta'}(s_i)) + \gamma D_{\psi'}(s_{i+k+1}, \bar{f}^{k+1}_{\theta'}(s_i))$
  Update critic with gradient descent using: $\nabla_\psi \frac{1}{n} \sum \left(D_\psi(s_{i+k}, \bar{f}_\theta{}^k(s_i)) - y \right)^2$
  if t mod d **then**
    Update model with gradient descent using: $\nabla_\theta \frac{1}{n} \left(\sum l(s_{t+1}, \bar{f}_\theta(s_t)) + \gamma D_\psi(s_{t+2}, \bar{f}_\theta{}^2(s_t)) \right)$
    Update target networks: $\theta' \leftarrow \tau_f \theta + (1 - \tau_f)\theta'$, $\psi' \leftarrow \tau_M \psi + (1 - \tau_M)\psi'$
  end if
end for

- Select a random integer $k \sim \{1, ..., \kappa\}$ for how far forward to go in the trajectory from s_i.
- Use $(s_{i+k}, \bar{f}_\theta{}^k(s_i))$ as a $(s, \hat{s})$ pair.

This gives us the following optimization problem for training the TDML approximator D_ψ:

$$\arg\min_\psi \mathop{\mathbb{E}}_{\substack{k \sim \{1,\ldots,\kappa\} \\ s_i, s_{i+k}, s_{i+k+1} \sim \beta}} \left[\left(D_\psi(s_{i+k}, \bar{f}_\theta{}^k(s_i)) - y \right)^2 \right] \tag{7}$$

where

$$y = l(s_{i+k}, \bar{f}_\theta{}^k(s_i)) + \gamma D_\psi(s_{i+k+1}, \bar{f}_\theta{}^{k+1}(s_i)) \tag{8}$$

5 Model Learning with Approximate TDML Objective

If we had access to the true TDML function, we could train models with the objective:

$$\begin{aligned} L_{\text{TDML}} &= \mathop{\mathbb{E}}_{s_t, s_{t+1} \sim \beta} \left[D^{f, f_\theta}(s_{t+1}, \bar{f}_\theta(s_t)) \right] \\ &= \mathop{\mathbb{E}}_{s_t, s_{t+1}, s_{t+2} \sim \beta} \left[l(s_{t+1}, \bar{f}_\theta(s_t)) + \gamma D^{f, f_\theta}(s_{t+2}, \bar{f}_\theta{}^2(s_t)) \right]. \end{aligned}$$

Using our approximator instead, we arrive at the objective:

$$L_{\text{TDML}} \approx \mathop{\mathbb{E}}_{s_t, s_{t+1}, s_{t+2} \sim \beta} \left[l(s_{t+1}, \bar{f}_\theta(s_t)) + \gamma D_\psi(s_{t+2}, \bar{f}_\theta{}^2(s_t)) \right]. \tag{9}$$

Although we have now given an objective for model-learning in terms of our TDML approximator, the picture for training the model is not complete. The TDML approximator is only valid for a fixed model, so after the model is updated, it will also need to be updated.

We take the approach of TD3 (Fujimoto et al., 2018), alternating between minimizing the "critic" D_ψ error and minimizing the "actor" f_θ error, incorporating a delay d which specifies the number of TDML approximator optimization steps per model optimization step. We also use target networks, as in TD3 (Fujimoto et al., 2018) and DDPG (Lillicrap et al., 2015), to stabilize the optimization procedure. See Algorithm 1 for the full procedure.

6 Does TDML work in practice?

Since we aim to isolate the model-training problem core to MBRL, we strip away confounding factors and focus on just the training procedure. We create a fixed replay buffer of interaction

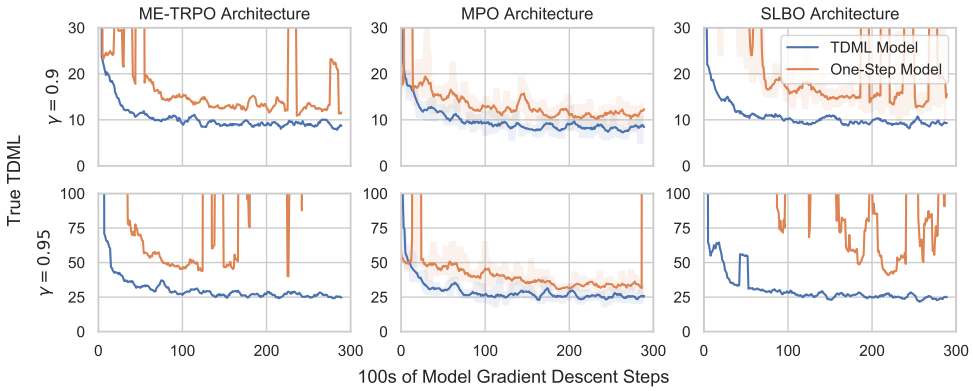

Figure 2: Comparison between one-step l_2 loss training (One-Step Model) and Approximate TDML training (TDML Model). Each column uses a model architecture from the literature, and each row corresponds to a different TDML discount factor. Plots are smoothed over a 1000 SGD step window. The shaded region represents the minimum and maximum values within that window.

data using a trained TD3 agent (Fujimoto et al., 2018) on the `Walker2d-v2` domain in OpenAI Gym (Brockman et al., 2016) to collect a buffer of 1000 trajectories, for a total of 10^6 transitions. Although the agent is deterministic, the trajectories vary due to random starting states.

We train various model architectures from the literature, specifically those used in ME-TRPO (Kurutach et al., 2018), MPO (Clavera et al., 2018), and SLBO (Xu et al., 2018), using both the one-step l_2 loss and the TDML learning procedure (with the l_2 loss as the one-step loss function l). In cases where ensembling was used in the original work, we train a single model. We do not use observation normalization or delta state prediction (in initial experiments, these tricks had some effect on performance, but their use confounds the relationship we are interested in exploring).

In order to measure the quality of long-horizon predictions, we compute the *true* total discounted model loss, L_{TDML}, for both the one-step-trained and approximate TDML-trained models. We evaluate L_{TDML} by sampling a batch of 16 states $s_t \sim \beta$, querying the real transition function f and learned model f_θ in parallel for 1000 steps (the full horizon in `Walker2d-v2`) starting from s_t, and then calculating the average TDML for the batch of trajectories according to Equation 3. We experiment with two different TDML discount factors (γ) of 0.9 and 0.95, and find that our results hold over discount factors. Finally, since the TDMLs of one-step models explode over long horizons (see Appendix B), we clip the models' one-step predictions between large values not encountered in the real environment (-10^6 and 10^6).

Our results are summarized in Figure 2 and in Appendix A. Hyperparameters are listed in Appendix C. Notably, training with approximate TDML not only decreases the true TDML significantly faster, but in long-horizon simulations, the models also do not reach areas in state space with unrealistic values or NaNs. This indicates that optimizing for the TDML metric improves the overall stability of the model.

We note that these architectures were tuned by the authors to perform well using one-step error, and that other architectures tuned for TDML could potentially perform even better.

7 Conclusion

In this work, we discussed challenges in learning environment models that are accurate over long horizons, and we proposed to use a new objective function, the total discounted model loss (TDML), to address these challenges. We also proposed an efficient method for approximating the TDML by exploiting a temporal difference equation, and a method for training a dynamics model using the TDML approximator as the objective function. Our experiments show that training with approximate TDML produces models with more accurate long-horizon predictions than training with a mean-squared error objective. In future work, we plan to explore the use of TDML training as a subroutine in model-based RL algorithms for control.

REFERENCES

Joshua Achiam, Ethan Knight, and Pieter Abbeel. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pp. 908–918, 2017.

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

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*, pp. 8234–8244, 2018.

Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. *arXiv preprint arXiv:1809.05214*, 2018.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

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. *arXiv preprint arXiv:1803.00101*, 2018.

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

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*, pp. 2944–2952, 2015.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pp. 971–980, 2017.

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

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.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017.

Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*, 2018.

APPENDIX A: TRAINING CURVES

Below we give figures that detail model training for various model architectures and settings for γ. In each sub-plot, the title corresponds to the metric being tracked (so, the plot marked "TDML" is what we measure for performance). For TDML Models, "Loss" refers to the approximate TDML.

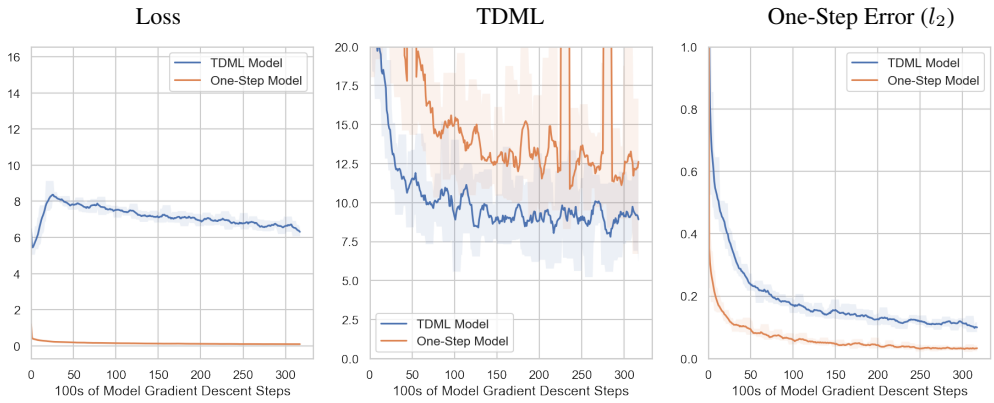

Figure 3: ME-TRPO training comparison with $\gamma = 0.9$

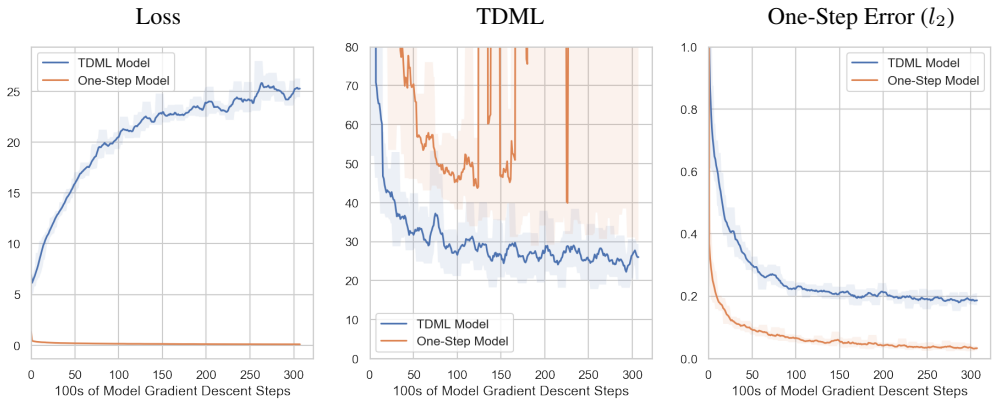

Figure 4: ME-TRPO training comparison with $\gamma = 0.95$

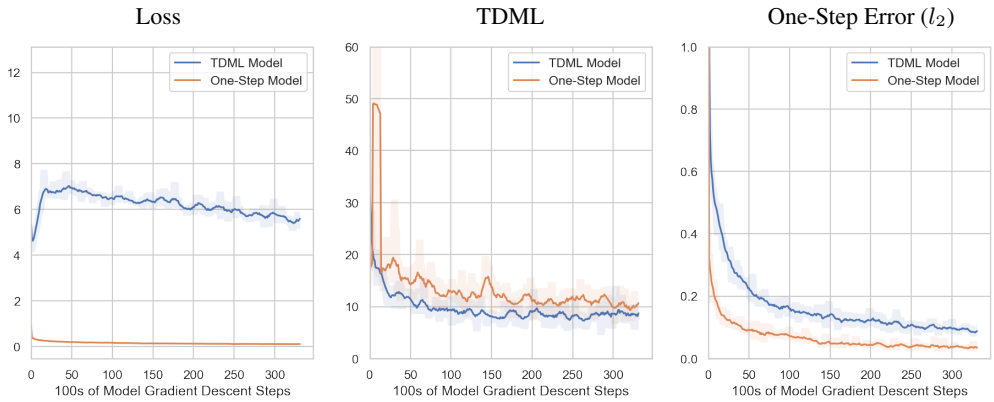

Figure 5: MPO training comparison with $\gamma = 0.9$

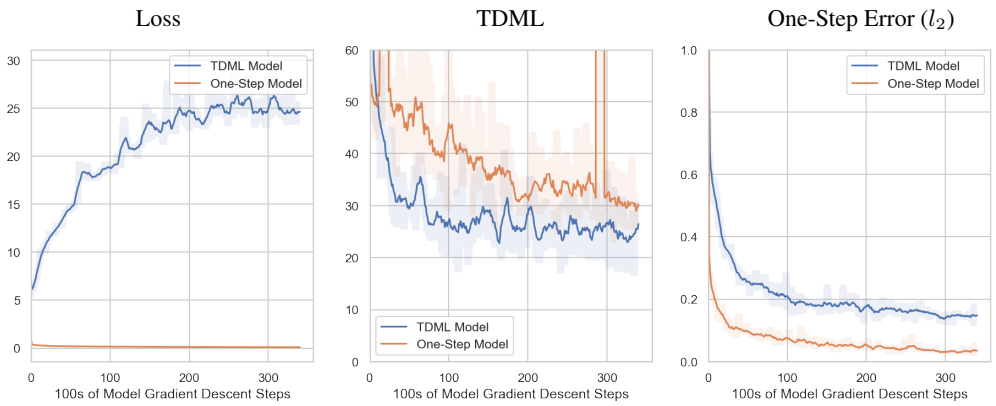

Figure 6: MPO training comparison with $\gamma = 0.95$

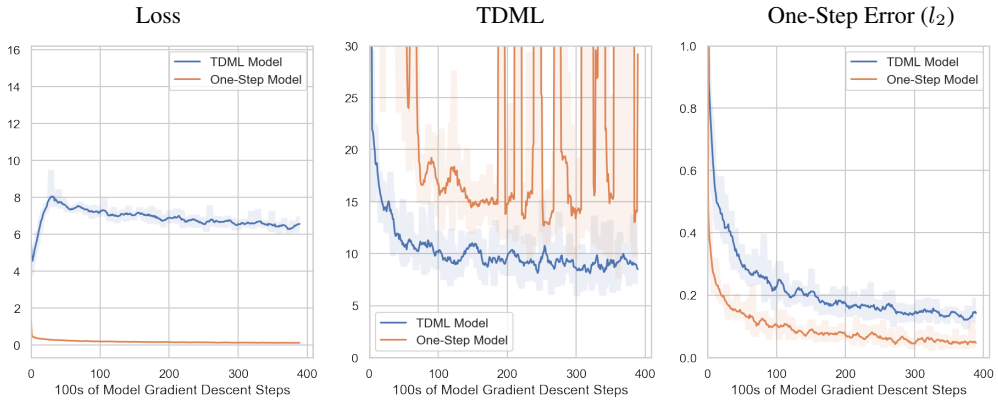

Figure 7: SLBO training comparison with $\gamma = 0.9$

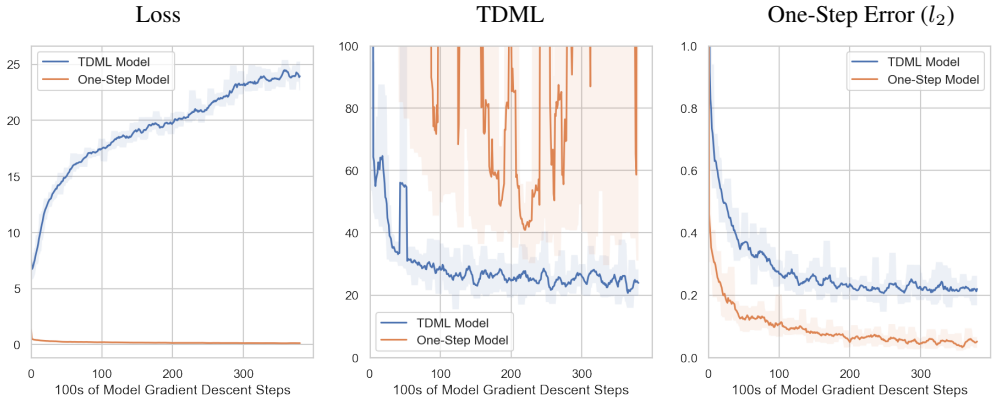

Figure 8: SLBO training comparison with $\gamma = 0.95$

APPENDIX B: MODEL VARIANTS

To demonstrate the need for clipping model predictions when using one-step loss, we also ran a slate of experiments where the model predictions were not clipped. We found that the TDML of one-step models tended to explode catastrophically when not clipped:

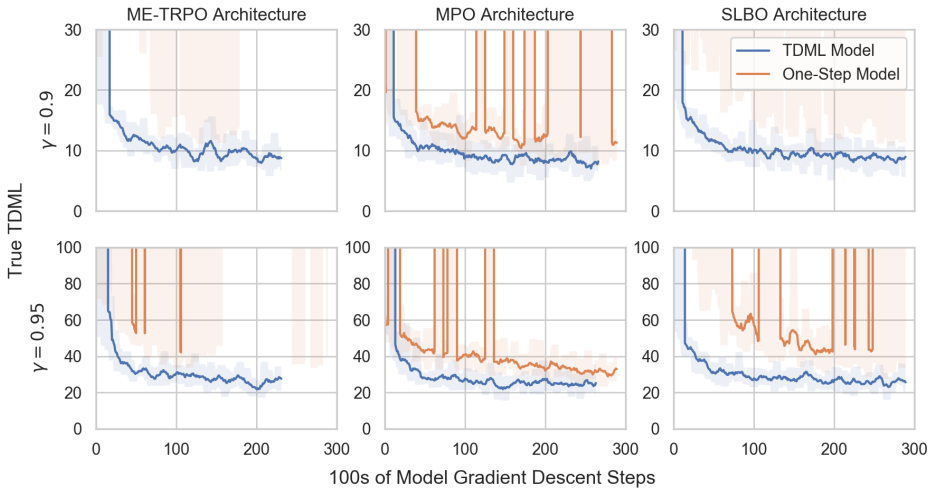

Figure 9: Results for models trained without state prediction clipping.

To further investigate rollout explosion, we also ran a separate slate of experiments using $\tanh$ activations instead of relu nonlinearities, since $\tanh$ saturates at extreme values and therefore is less prone to explosion.

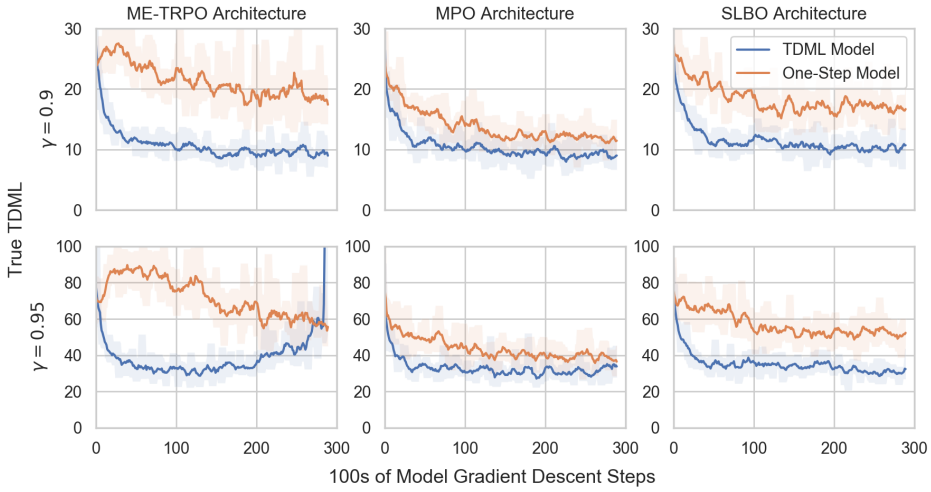

Figure 10: Results for models trained using $\tanh$ nonlinearities.

We found that the models trained using approximate TDML still beat out one-step models. However, the TDML of one model trained with approximate TDML diverged, which could be related to divergence issues explored with other temporal-difference algorithms (Achiam et al., 2019). In later work, we plan to investigate how work related to improving stability of TD-methods can improve training with approximate TDML.

APPENDIX C: HYPERPARAMETERS

All models used the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 10^{-3}.

Architecture	Layers	Activation	Batch size	L2 λ	Weight norm?
ME-TRPO (Kurutach et al., 2018)	[1024, 1024]	ReLU	1000	0	No
MPO (Clavera et al., 2018)	[512, 512, 512]	ReLU	500	0	Yes
SLBO (Xu et al., 2018)	[500, 500]	ReLU	256	10^{-5}	No

Table 1: Model architectures tested

The TDML network was also trained with the Adam optimizer with a learning rate of $2 \cdot 10^{-4}$ (see Algorithm 1).

Layers	Activation	Batch size	Update delay d	κ	τ_f	τ_M
[1024, 1024]	SeLU (Klambauer et al., 2017)	2048	10	5	0.005	1

Table 2: Hyperparameters for TDML network, M