PWM: Policy Learning with Large World Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reinforcement Learning (RL) has achieved impressive results on complex tasks but struggles in multi-task settings with different embodiments. World models offer scalability by learning a simulation of the environment, yet they often rely on inefficient gradient-free optimization methods. We introduce Policy learning with large World Models (PWM), a novel model-based RL algorithm that learns continuous control policies from large multi-task world models. By pre-training the world model on offline data and using it for first-order gradient policy learning, PWM effectively solves tasks with up to 152 action dimensions and outperforms methods using ground-truth dynamics. Additionally, PWM scales to an 80-task setting, achieving up to 27% higher rewards than existing baselines without the need for expensive online planning. Visualizations and code available at `policy-world-model.github.io`

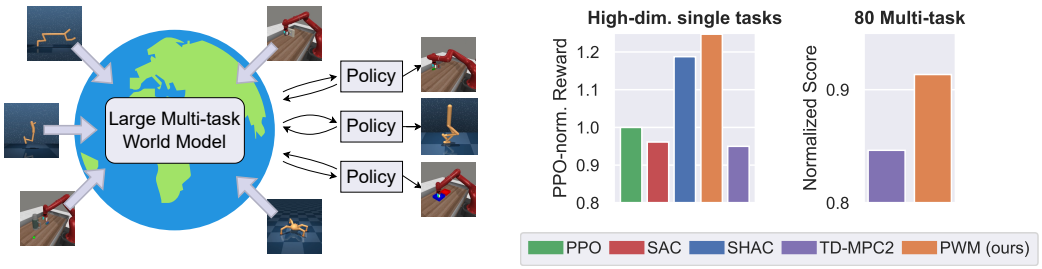


Figure 1: We propose PWM, a novel approach to scalable RL where we harness large multi-task world models to rapidly train model-based RL policies using efficient first-order gradients. When the world models used are regularized and avoid exploding gradients, this results in efficient policy learning in $\leq$ *10 minutes per task*, while achieving higher reward in both single-task and multi-task settings.

1 Introduction

The pursuit of generalizability in machine learning has recently been propelled by the training of large models on substantial datasets [7, 22, 4]. Such advancements have notably permeated robotics, where multi-task behavior cloning techniques have shown remarkable performance [48, 32, 12, 5]. Nevertheless, these approaches predominantly hinge on near-expert data and struggle with adaptability across diverse robot morphologies due to their dependence on teleoperation [48, 32, 24].

In contrast, Reinforcement Learning (RL) offers a robust framework capable of learning from suboptimal data, addressing the aforementioned limitations. However, traditional RL has been focused on single-task experts [29, 39, 13]. Recent work suggests that a potential pathway to multi-task RL is with the world models framework, where a large-scale pre-trained model learns the

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

environment dynamics and is then combined with zeroth-order online planning to solve various tasks [8, 36, 45, 9, 16, 37].

Despite advancements, zeroth-order methods often result in suboptimal solutions due to high variance [30, 40, 34] and online planning time scales with model size, rendering them infeasible at scale [37, 16]. These issues are particularly noticeable in world models with billions of parameters such as GAIA-1 [19] and UniSim [46]. To address these challenges, we suggest a new approach: using world models as scalable, differentiable physics simulators to enable efficient policy learning through First-order Gradients (FoG). They have demonstrated better performance and stability than traditional zeroth-order methods [44, 11]. FoGs have been previously paired with world models approaches [14, 15] but often as complete algorithms that do not study the optimization landscape induced by the world models. [26] studied these landscapes but focused on different learning architectures and did not scale their results. In this work, we analyze world models with differentiable simulation principles and find that the utility of world models lies not in their sheer accuracy but in (1) their smoothness, (2) their optimality gap, and (3) their gradient stability over long-horizon forecasts [40, 18].

Building on these insights, we introduce Policy learning with large World Models (PWM), a novel Model-Based RL (MBRL) algorithm and framework aimed at deriving effective continuous control policies from large, multi-task world models. We utilize pre-trained TD-MPC2 world models to efficiently learn control policies with FoG in <10m per task. Our empirical evaluations on complex locomotion tasks indicate that PWM not only achieves higher reward than baselines but also outperforms methods that use ground-truth simulation dynamics. In a multi-task scenario utilizing a pre-trained 48M parameter world model, PWM achieves up to 27% higher reward than TD-MPC2 without relying on online planning.

This underscores the efficacy of PWM and supports our broader contributions:

1. Through pedagogical examples and ablations, we show that more accurate world models do not translate to better performing policies. Instead of pursuing world model improvements in isolation, we should aim to build world models that result in better policies.
2. When regularized correctly, world models enable efficient first-order optimization. We show that this results in better performing policies and faster training times in comparison to zeroth-order gradient methods.
3. We propose PWM, an algorithm for learning expert policies from large multi-task world models in minutes using First-order Gradients (FoG).

2 Related work

RL approaches can be classified as model-based and model-free which assume and do not assume a model respectively [2]. Most common algorithms for real-world applications such as PPO [39] and SAC [13] are model-free and fall in the category of on-policy and off-policy methods respectively [2]. Both harness an actor-critic architecture where the critic learns the value and the actor optimizes the critic directly to maximize cumulative rewards [23]. Off-policy methods such as SAC typically learn a policy by taking First-order Gradients (FoG) directly from the critic. FoGs optimization typically has lower variance but is affected by discontinuities of the objective [30]. On-policy methods such as PPO harness zeroth-order gradients to learn policies [42]. These gradients are not affected by discontinuities, making them effective for robotic tasks such as locomotion [35] but also exhibit high variance, which leads to slow optimization and suboptimal solutions [30, 40]. Differentiable simulation has been a popular framework to explore the properties of these gradient types [40, 18, 44].

While RL remains mostly focused on single-task policies, the larger robotics community has shifted focus to large multi-task models relying on behavior cloning [10]. RT-1 [6] trains a large Transformer sequence model on a vast dataset to perform low-level object manipulation on a real robot. RT-2 [48] extends this by pre-training an even larger model on internet-scale data to also do planning and grasp affordance. Most recently, Open X [33] and Octo [32] showed that training these approaches on different tasks and embodiment leads to increased performance on single-task single-embodiment evaluations. Unfortunately, the benefits of large models and large datasets have not been sufficiently explored in the context of RL. DreamerV3 [15] is a Model-Based Reinforcement Learning (MBRL) approach which learns a representation of its environment with encoder, decoder, dynamics and reward modules (collectively dubbed *world model*). It showed that scaling to larger models (400M

parameters) leads to higher data efficient and episode rewards. TD-MPC2 [16] continued this trend by employing a 317M parameter world model to learn an online planning multi-task policy capable of achieving good performance on 80 tasks. While showing impressive multi-task scalability, TD-MPC2 fails to solve all tasks it was trained on and exhibits limited scalability due to its use of online planning. With world models increasingly growing in size, we need a policy learning technique capable of scaling to billion parameter world models such as GAIA-1 [19] or UniSim [46].

3 Background

We focus on discrete-time and infinite horizon Reinforcement Learning (RL) scenarios characterized by system states $\boldsymbol{s} \in \mathbb{R}^n = \mathcal{S}$, actions $\boldsymbol{a} \in \mathbb{R}^m = \mathcal{A}$, dynamics function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Combined, these form a Markov Decision Problem (MDP) summarized by the tuple $(\mathcal{S}, \mathcal{A}, f, r, \gamma)$ where γ is the discount factor. Actions at each timestep t are sampled from a stochastic policy $\boldsymbol{a}_t \sim \pi_\theta(\cdot|\boldsymbol{s}_t)$, parameterized by $\boldsymbol{\theta}$. The goal of the policy is to maximize the cumulative discounted rewards:

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) := \max_{\boldsymbol{\theta}} \mathbb{E}_{\substack{\boldsymbol{s}_1 \sim \rho(\cdot) \\ \boldsymbol{a}_t \sim \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}_t)}} \left[\sum_{t=1}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right] \tag{1}$$

where $\rho(\boldsymbol{s}_1)$ is the initial state distribution. Since this maximization over an infinite sum is intractable, in practice we often maximize over a value estimate. The value of a state $\boldsymbol{s}_t$ is defined as the expected reward follow the policy $\pi_{\boldsymbol{\theta}}$

$$V^{\pi}_{\boldsymbol{\psi}}(\boldsymbol{s}_t) := \mathbb{E}_{\boldsymbol{a}_h \sim \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}_h)} \left[\sum_{h=t}^{\infty} \gamma^h r(\boldsymbol{s}_h, \boldsymbol{a}_h) \right] \tag{2}$$

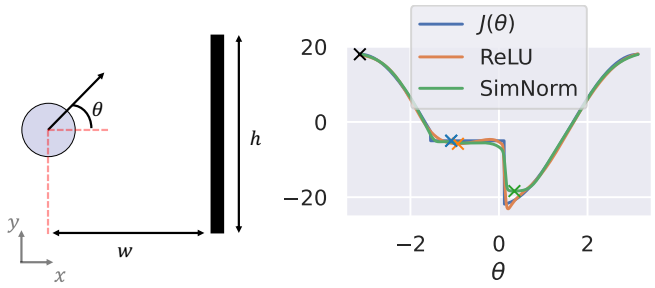
When V is approximated with a learned model with parameters $\boldsymbol{\psi}$ and $\pi_{\boldsymbol{\theta}}$ attempts to maximize some function of V, we arrive at the popular and successful actor-critic architecture [23]. Additionally, in MBRL it is common to also learn approximations of f and r, which we denote as $F_{\boldsymbol{\phi}}$ and $R_{\boldsymbol{\phi}}$, respectively. It has also been shown to be beneficial to encode the true state $\boldsymbol{s}$ into a latent state $\boldsymbol{z}$ using a learned encoder $E_{\boldsymbol{\phi}}$ [14, 17, 16, 15]. Putting together all of these components we can define a model-based actor-critic algorithm to consist of the tuple $(\pi_{\boldsymbol{\theta}}, V_{\boldsymbol{\psi}}, E_{\boldsymbol{\phi}}, F_{\boldsymbol{\phi}}, R_{\boldsymbol{\phi}})$ which can describe popular approaches such as Dreamer [14] and TD-MPC2 [16]. Notably, we make an important distinction between the types of components. We refer to $E_{\boldsymbol{\phi}}$, $F_{\boldsymbol{\phi}}$ and $R_{\boldsymbol{\phi}}$ as the *world model components* since they are a supervised learning problem with fixed targets. On the other hand, π_θ and V_ψ optimize for moving targets which is fundamentally more challenging and we refer to them as the *policy components*.

4 Policy optimization through learned world models

This paper builds on the insight that since access to $F_{\boldsymbol{\phi}}$ and $R_{\boldsymbol{\phi}}$ is assumed through a pre-trained world-model, we have the option to optimize Eq. 1 via *First-order Gradient (FoG) optimization* which exhibit lower gradient variance, more optimal solutions and improved sample efficiency [30]. In our setting, these types of gradients are obtained by directly differentiating the expected terms of Eq. 1 as shown in Eq. 3. Note that this gradient estimator is also known as reparameterized gradient [21] and pathwise derivative [38]. While we use the explicit $\nabla^{[1]}$ notation below, we later drop it for simplicity as all gradient types in this work are first-order gradients.

$$\nabla^{[1]}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) := \mathbb{E}_{\substack{\boldsymbol{s}_1 \sim \rho(\cdot) \\ \boldsymbol{a}_h \sim \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}_h)}} \left[\nabla_{\boldsymbol{\theta}} \left(\sum_{t=1}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right) \right] \tag{3}$$

As $\nabla^{[1]}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ in itself is a random variable, we need to estimate it. A popular way to do that in practice is via Monte-Carlo approximation where we are interested in two properties - bias and variance. In Sections 4.1 and 4.2 we tackle each aspect with toy robotic control problem to build intuition. In Section 4.3 we combine our findings to propose a new algorithm.



(a) Ball-wall visualization. **(b)** Problem landscape.

Model	Model error	Opt. gap
True	0.0	16.85
ReLU	0.71	16.05
SimNorm	1.13	3.47

(c) Model error and optimality gap.

Figure 2: Ball-wall pedagogical example. The **left** figure visualizes the problem. The middle figure shows the problem landscape induced by each model. $J(\theta)$ shows the true underlying function and the two other are MLPs with different activation functions. We minimize each of these problems using gradient descent and starting at $\theta = -\pi$ (marker $\times$). The colored crosses represent the solutions converged to for each model. The **right** table shows the model approximation error during training and the optimality gap $|J(\theta^*) - J(\hat{\theta})|$ between the global minimum θ^* and the solution found for each model $\hat{\theta}$.

4.1 Learning through contact

FoGs are unbiased $\mathbb{E}\left[\bar{\nabla}^{[1]} J(\boldsymbol{\theta})\right] := \mathbb{E}\left[\sum_{n=1}^{N} \hat{\nabla}_n^{[1]} J(\boldsymbol{\theta})\right] = \nabla J(\boldsymbol{\theta})$, only if both the dynamics f and rewards r are Lipschitz-smooth [40]. However, many robotic problems involving contact are inherently non-smooth, which breaks these conditions and results in gradient sample error where $\mathbb{E}\left[\bar{\nabla}^{[1]} J(\boldsymbol{\theta})\right] \neq \nabla J(\boldsymbol{\theta})$ under finite number of samples N. Instead of directly optimizing the true, discontinuous objective, it is advantageous to optimize a smooth surrogate, such as a model learned by a regularized deep neural network.

To illustrate this concept, we use a toy problem where a ball is thrown toward a wall at a fixed velocity as shown in Figure 2a. The objective is to find the optimal initial angle θ such that we maximize forward distance. In this simplified pedagogical example, we assume that the ball "sticks" to the wall, creating a discontinuous optimization landscape (Figure 2b). We compare the performance of two models in approximating this objective: a 2-layer Multi-Layer Perceptron (MLP) with ReLU activation and another MLP with SimNorm activation [16] in the intermediate layers. SimNorm normalizes a latent vector $\boldsymbol{z}$ by projecting it into simplices with dimension V using a softmax operator. Given an input vector $\boldsymbol{z}$, SimNorm can be expressed as a mapping into L vectors:

$$\text{SimNorm}(\boldsymbol{z}) := [\boldsymbol{g}_1, ..., \boldsymbol{g}_L], \quad \boldsymbol{g}_i = \text{Softmax}(\boldsymbol{z}_{i:i+V}) \tag{4}$$

We train the MLPs and observe the smoothing effects of the learned models in Figure 2b. While the MLP smooths the problem landscape, it also introduces a local minimum when attempting to optimize with gradient descent starting from (e.g.) $\theta = -\pi$, leading to a large optimality gap (difference between the solution and the optimal solution: $\|\hat{\theta} - \theta^*\|$). In contrast, the SimNorm MLP has additional regularization which reduces the optimality hap, at the expense of model accuracy (Table 2c). Therefore, we believe that regularized learned models can reduce gradient sample error, and thus the optimality gap, enabling more efficient FoG optimization in non-smooth environments. Further details in Appendix A.

4.2 Learning with chaotic dynamics

While first-order gradient estimators (FoGs) have lower variance per step, they can accumulate significant variance over long-horizon rollouts [27]. [40] link this variance to the smoothness of models and the length of the prediction horizon: $\text{Var}\left[\nabla J^{[1]}\right] \propto \|\nabla f(s,a)\|^{2H}$. At sufficiently high H, the high variance renders FoGs ineffective in chaotic systems. Chaotic systems are characterized by their sensitivity to initial conditions, where small perturbations can lead to exponentially divergent trajectories, making long-term prediction particularly challenging. The double pendulum, also known as the Acrobot [31], is a classic example of such a system (Figure 3).

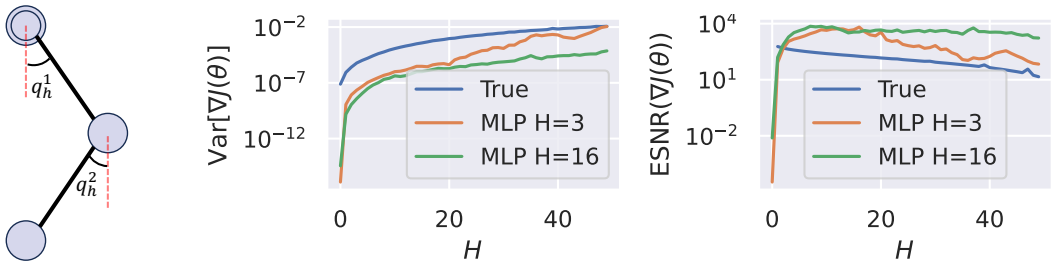


Figure 3: Double pendulum pedagogical example. The middle figure evaluates the variance of policy gradient estimates over $N = 100$ Monte-Carlo samples for varying horizons H. The right figure shows the same data but plots the Expected Signal-to-Noise ratio (ESNR) with higher values translating to more useful gradients. These results suggests that world models trained over long horizon trajectories provide more useful gradients. Note that $H = 3$ and $H = 16$ in the figure legends refer to the training horizon of the models.

We analyze the variance of gradient estimators in the double pendulum using both the true dynamics and a SimNorm-activated MLP model. The MLP model was trained for auto-regressive prediction horizons of $H = 3$ and $H = 16$ until convergence on a large dataset. Figure 3 shows that both learned models exhibit reduced variance compared to the true dynamics. However, as noted by [34], variance alone is insufficient for drawing definitive conclusions about gradient quality. Instead, they propose analyzing gradients via their Expected Signal-to-Noise Ratio (ESNR), defined as:

$$\text{ESNR}(\nabla J(\boldsymbol{\theta})) = \mathbb{E}\left[\frac{\sum \mathbb{E}\left[\nabla^{[1]} J(\boldsymbol{\theta})\right]^2}{\sum \text{Var}\left[\nabla^{[1]} J(\boldsymbol{\theta})\right]}\right] \tag{5}$$

In Figure 3, we observe that learned models exhibit higher ESNR than the true dynamics, providing more useful gradients. Notably, the training horizon plays a critical role, with the $H = 16$ model sustaining a higher ESNR over higher H. We conclude that learned world models offer more informative policy gradients than the true system dynamics. Further details in Appendix B.

4.3 PWM: an efficient policy learning method

Given the results from the previous subsection, we propose to view world models not as components of RL methods but instead as scalable differentiable physics simulators which provide gradients with low sample error and variance. It is worth noting that approaches such as TD-MPC2 [16] do not exploit these properties but rather choose to optimize policies via DDPG-style gradients: $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \mathbb{E}_{\boldsymbol{a}\sim\pi(\cdot|s)}[\nabla_{\boldsymbol{\theta}} Q(\boldsymbol{s}, \boldsymbol{a})]$.

We propose a new method and framework for learning policies from large multi-task world models.

Framework. Assuming availability of data from multiple tasks, we first train a multi-task world model to predict future states and rewards. Then for each task we want to solve, we learn a single policy in minutes using FoG optimization. The policy is then deployed to solve the task and optionally finetune its world model and policy. We dub this PWM: Policy optimization through World Models.

Algorithm 1: PWM: Policy Learning with large World Models

Given: Multi-task dataset $\mathcal{B}$
Given: γ: discount rate
Given: $\alpha_{\boldsymbol{\theta}}, \alpha_{\boldsymbol{\psi}}, \alpha_{\boldsymbol{\phi}}$: learning rates
Initialize learnable parameters $\boldsymbol{\theta}, \boldsymbol{\psi}, \boldsymbol{\phi}$
▷ `Pre-train world model once`
for *N epochs* **do**
  $\boldsymbol{s}_{1:H}, \boldsymbol{a}_{1:H}, r_{1:H}, \boldsymbol{e} \sim \mathcal{B}$
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha_{\boldsymbol{\phi}} \mathcal{L}_{wm}(\boldsymbol{\phi})$ ▷ `Eq. 10`
end
▷ `Train policy on task embedding` $\boldsymbol{e}$
for *M epochs* **do**
  $\boldsymbol{s}_1 \sim \mathcal{B}$
  $\boldsymbol{z}_1 = E_{\boldsymbol{\phi}}(\boldsymbol{s}_1, \boldsymbol{e})$
  for *h=[1, ..., H]* **do** ▷ `Rollout`
    $\boldsymbol{a}_h \sim \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{z}_h)$
    $r_h = R_\phi(\boldsymbol{z}_h, \boldsymbol{a}_h, \boldsymbol{e})$
    $\boldsymbol{z}_{h+1} = F_\phi(\boldsymbol{z}_h, \boldsymbol{a}_h, \boldsymbol{e})$
  end
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \mathcal{L}_\pi(\boldsymbol{\theta})$ ▷ `Eq. 7-9`
  $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \alpha_{\boldsymbol{\psi}} \mathcal{L}_V(\boldsymbol{\psi})$ ▷ `Eq. 6`
end

Method. For policy learning, we propose on-policy actor-critic approach where the actor is trained via FoG back-propagated through the world model, while the critic is trained via TD(λ). The key to our approach is that training is done in a batched fashion where multiple trajectories are imagined in parallel. The actor loss function is akin to 1 but features rewards over a fixed horizon H, terminal value bootstrapping and usage of the learned world model components:

$$\mathcal{L}_\pi(\boldsymbol{\theta}) := \mathbb{E}_{\substack{\boldsymbol{s}_1 \sim \rho(\cdot) \\ \boldsymbol{a}_h \sim \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{z}_h)}} \left[\sum_{h=1}^{H-1} \gamma^h R_\phi(\boldsymbol{z}_h, \boldsymbol{a}_h) + \gamma^H V_{\boldsymbol{\psi}}(\boldsymbol{z}_H) \right] \quad \text{where} \quad \begin{matrix} \boldsymbol{z}_1 = E_{\boldsymbol{\phi}}(\boldsymbol{s}_1) \\ \boldsymbol{z}_{t+1} = F_{\boldsymbol{\phi}}(\boldsymbol{z}_t, \boldsymbol{a}_t) \end{matrix} \tag{6}$$

The critic is trained in a model-free fashion using TD(λ) over an H-step rollout in latent space $\boldsymbol{z}$ as seen in other similar on-policy methods [41, 14, 44]:

$$V_h(\boldsymbol{z}_t) := \sum_{n=t}^{t+h-1} \gamma^{n-t} R_\phi(\boldsymbol{z}_n, \boldsymbol{a}_n) + \gamma^{t+h} V_{\boldsymbol{\psi}}(\boldsymbol{z}_{t+h}) \tag{7}$$

$$\hat{V}(\boldsymbol{z}_t) := (1-\lambda) \left[\sum_{h=1}^{H-t-1} \lambda^{h-1} V_h(\boldsymbol{z}_t) \right] + \lambda^{H-t-1} V_H(\boldsymbol{z}_t) \tag{8}$$

$$\mathcal{L}_V(\boldsymbol{\psi}) := \sum_{h=t}^{t+H} \left\| V_{\boldsymbol{\psi}}(\boldsymbol{z}_h) - \hat{V}(\boldsymbol{z}_h) \right\|_2^2 \tag{9}$$

We use an ensemble of 3 critics to reduce variance. To enable FoG optimization, it is important to use a well-regularized world model. We use the $\big(E_\phi(\boldsymbol{s}, \boldsymbol{e}), F_\phi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{e}), R_\phi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{e})\big)$ model proposed by TD-MPC2 [16] with learnable task embeddings $\boldsymbol{e}$. It is trained in an auto-regressive fashion by sampling data from a buffer with loss function:

$$\mathcal{L}_{\text{wm}}(\boldsymbol{\phi}) = \mathbb{E}_{(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}', \boldsymbol{e})_{0:H} \sim \mathcal{B}} \left[\sum_{t=0}^{H} \gamma^t \big(\| \boldsymbol{z}_{t+1} - sg(E_\phi(\boldsymbol{s}_{t+1}, \boldsymbol{e})) \|_2^2 + CE(\hat{r}_t, r_t) \big) \right] \tag{10}$$

where $sg(\cdot)$ is the stop-gradient operator and CE is the cross-entropy loss function. Reward prediction is formulated as a discrete regression problem in log-transformed space. Furthermore, $E_{\boldsymbol{\phi}}$ and $F_{\boldsymbol{\phi}}$ use SimNorm activation (Eq. 4) in their output layers. All trainable models are fully-connected MLPs with LayerNorm [3] and Mish activation [28]. The complete algorithm is shown in Algorithm 1. Further implementation details and hyper-parameters are listed in Appendix C.

5 Experimental results

5.1 Contact-rich single tasks

We first assess our proposed method on complex continuous control tasks with up to $\mathcal{A} = \mathbb{R}^{152}$ using the differentiable simulator dflex [44]. Hopper, Ant, Anymal, Humanoid and muscle-actuated (SNU) Humanoid (Figure 4) are tasked to maximize forward velocity [1]. We compare against SHAC [44],

[1] While these tasks are inspired by dm_control [43], they are also fundamentally more challenging as they have to maximize an unbounded forward velocity, not achieve a velocity target.

Figure 4: High-dimensional single-task environments (left to right): Hopper, Ant, Anymal, Humanoid and SNU Humanoid. Our method successfully learns tasks with up to $m = 152$ continuous action dimensions. Additional 80 multi-task environments used in this paper are listed in Appendix E

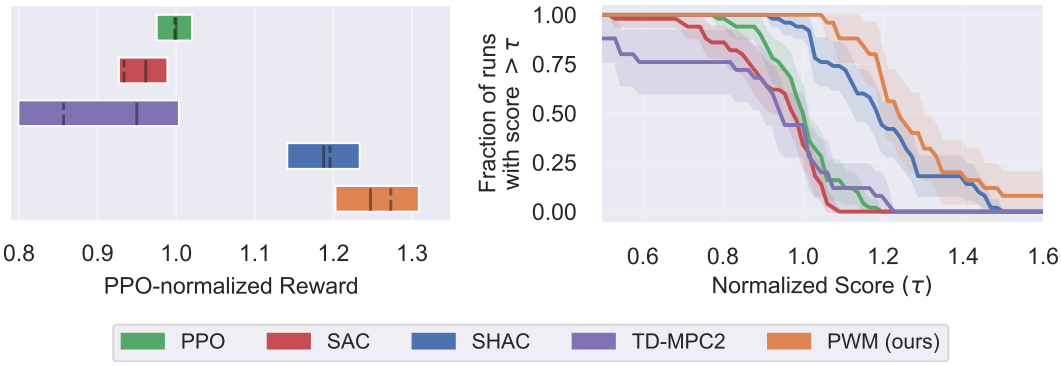


Figure 5: Aggregate results from high-dimensional locomotion tasks where each agent is trained to solve just that task (i.e. specialist). The **left** figure summarizes rewards achieved at the end of training using 50% IQM for the solid lines and 95% CI as suggested by [1], as well as mean for the dashed lines. We see that PWM achieves higher rewards than our main baselines TD-MPC2 and SHAC. The **right** figure shows score distributions across all tasks which lets us understand the performance variability of each approach. PWM exhibits a similar curve to SHAC but different than TD-MPC2, due to the policy learning approach.

a method with a similar actor-critic architecture but uses gradients directly from the differentiable simulation. This allows us to understand whether world models induce better landscapes for policy learning. Furthermore, we compare against TD-MPC2 which uses the same world model but learns a policy in a model-free fashion and actively plans at inference time. This comparison allows us to understand whether first-order gradients can learn better policies. We additionally include prominent model-free baselines PPO [39] and SAC [13]. TD-MPC2 follows the hyper-parameters from the original work [16], while all other baselines follow [44].

We conduct this experiment across 5 tasks with 5 seeds each where PWM and TD-MPC2 use the same pre-trained world models and are left to learn a policy and finetune their world models online. This is done to enable fair comparison to SHAC which directly has access to the simulation model and does not require any training. The results in Figure 5 reveal that (1) PWM is able to learn policies with higher reward than SHAC asymptotically, indicating that regularized world models induce smooth optimization landscapes than the true (discontinuous) dynamics. Furthermore (2) our method is able to learn policies with higher rewards than TD-MPC2 without the need for online planning and with the same compute time budget. However, PWM does not scale well to the highest dimensional task. More experiment details and results are included in Appendix D.

5.2 Multi-task world-model

We analyze the scalability of our proposed framework and method to large multi-task pre-trained world models. We evaluate on two settings: (1) 30 continuous control dm_control tasks [43] ranging from $m = 1$ to $m = 6$ and (2) 80 tasks which include 50 additional manipulation tasks from MetaWorld [47] with $n = 39$ and $m = 4$. These two multi-task settings were introduced as *MT30* and *MT80* by [16]. In conducting our experiments, we harness the same data and world model architecture as TD-MPC2. The data consists of 120k and 40k trajectories per dm_control and MetaWorld task, respectively generated by 3 random seeds of TD-MPC2 runs. The world models we use are the 48M parameter models introduced in [16].

To train PWM, we first pre-train the world models on the dataset in a similar fashion to TD-MPC2 but with training $H = 16$ and $\gamma = 0.99$ for better first-order gradients as highlighted in Section 4.2. Then we train a PWM policy on each particular task using the offline datasets for 10k gradient steps which take 9.3 minutes on an Nvidia RTX6000 GPU. We evaluate task performance for 10 seeds for each task and aggregate results in Figure 6. We compare against TD-MPC2 which learns a multi-task policy while pre-training its world model and relies on online planning at inference. We can see that PWM learns behavior achieving higher reward than TD-MPC2 while also being significantly faster at inference time. We further compare our multi-task PWM policy to online-trained single-task experts SAC [13] and DreamerV3 [15]. Figure 6 reveals that multi-task PWM, while disadvantaged, performs comparably to the single-task experts without requiring any environment interaction and only training policies for ≤ 10 minutes per task. Additional results in Appendix E.

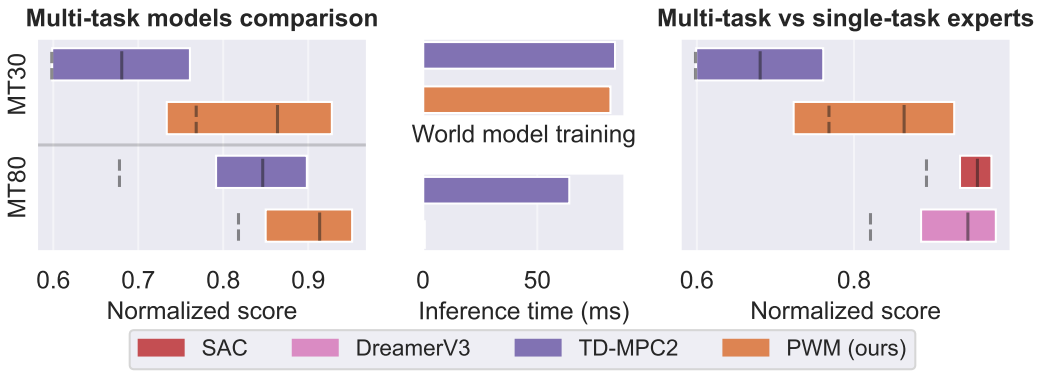
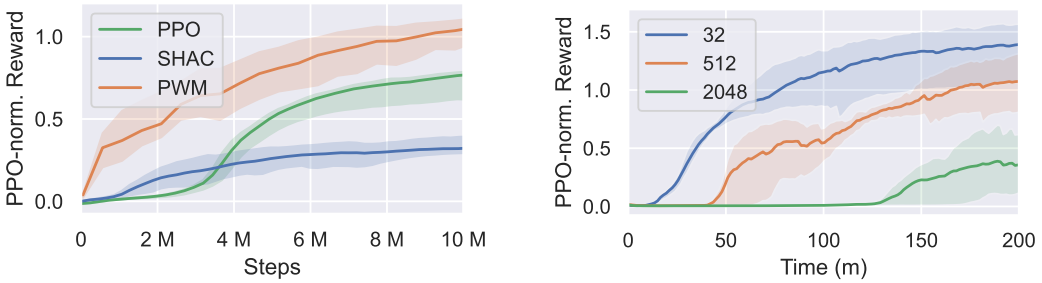


Figure 6: Multi-task results. The **left** figure shows results of multi-task agents in the 30 and 80 task set settings which include environments from dm_control [43] and MetaWorld [47]. The results show 50% IQM with the solid lines and mean with the dashed lines. The bars represent 95% CI. In both settings PWM achieves higher reward than TD-MPC2 without the need for online planning. The **middle** figure compares the training and inference times of TD-MPC2 and PWM for the 48M parameter model. We can see that PWM has significantly lower inference time as it does not use online planning. The **right** figure shows a comparison between multi-task PWM and single-task experts SAC and DreamerV3 on the MT30 task set. Notably, PWM is able to mat ch the performance of SAC and DreamerV3 without requiring any environment interaction.



(a) Contact stiffness ablation.

(b) Policy batch size ablation.

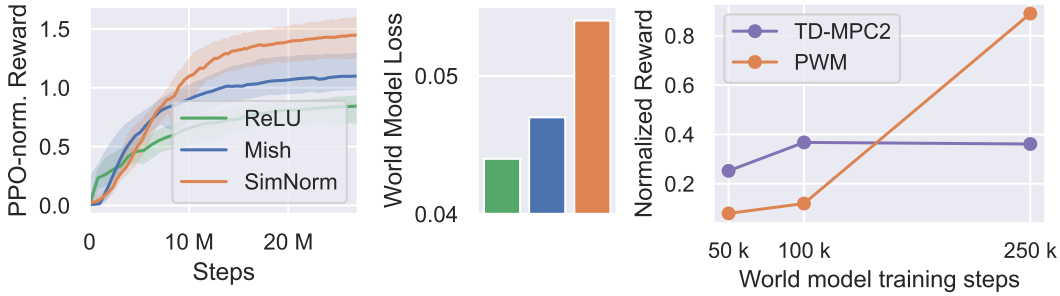
Figure 7: Left figure shows contact stiffness ablation where we increase contact stiffness on the Hopper task and analyze the effects on policy learning. The results indicate that stiff (but realistic) contact has adverse effects on SHAC which uses the simulation model to learn. Meanwhile, PPO and PWM remain unaffected with PWM still obtaining 17% more reward than PPO asymptotically. The **right figure shows a policy batch size ablation** on the Any task where we vary only the batch size used to train the policy components of PWM. Unfortunately we observe that PWM provides best result within a unit of time by using small batch sizes. Both figures show 50% IQM and 95% CI over 5 random seeds.

5.3 Ablations

We perform 4 ablations on the complex single task experiments in order to understand the nuances of first-order optimization through world models with PWM.

We increase the **contact stiffness** to be more realistic but also more stiff contact gives gradients with high sample error [40]. We run the same experiment as Section 5.1, but only for the Hopper task and present the aggregate results from 5 random seeds in Figure 7a where we normalize rewards by the maximum reward achieved by PPO in Section 5.1. We see that while PPO and PWM rewards remain similar to prior results, while SHAC performance decreases by 48%. This shows that regularized world models are robust to stiff contact models and thus more generalizble the differentiable simulations.

The **second ablation explores batch sizes** for policy learning with first-order gradients. Contrary to model-free methods which can scale to large batch sizes, we find that FoG techniques like PWM benefit from smaller batch sizes. We explore this on the Ant task in Figure 7b where we plot 50%



(a) World model ablation. **(b)** World model vs policy sample efficiency.

Figure 8: The **left figure ablates the activation functions** of the world model used to learn policies on the Ant task. We progressively add more regularization to the world model via changes to the activation function and observe an inverse correlation between world model loss and policy reward. This indicates that we should not construct world models for accuracy but for policy learning. The **right figure investigates the policy sample efficiency** on 5 dm_control tasks. We use the same data to pre-train world models for varying amount of gradient steps and then train the policy for 50k gradient steps and compare against TD-MPC2 (without planning). The results indicate that PWM policies are significantly more sample efficient but also require better trained world models. All results shown are 50% IQM with 95% CI across 5 random seeds.

IQM rewards over 5 random seeds. While larger batch sizes allow us to generate more data within a unit of time, that does not necessarily translate to learning better policies.

Next we **ablate the world model regularization**. We perform the same experiment as Section 5.1 on the Ant task but now pre-train 3 different world models. (1) with ReLU activation func., (2) with Mish activation func. and (3) with Mish activation func. and SimNorm activation func. at the output layers of E_ϕ and F_ϕ. Figure 8a reveals that while less regularization results in lower world model error, that does not translate to learning better policies. Surprisingly, less regularized world models enable policies to start faster (up to 1M steps) but plateau to a suboptimal policy.

To understand the **policy sample efficiency** of PWM while controlling for the world model, we perform an ablation where we pre-train the same world model for [50k, 100k, 250k] gradient steps on an offline dataset. Then we fix the world model and train only the policy components on the same dataset for 50k gradient steps and measure the reward. We do this for 3 random seeds and 5 dm_control tasks. We repeat the same experiment for TD-MPC2 but disable its planning component in order to understand the learning dynamics of each methods' policy components. The results in Figure 8b show that the PWM policy components are significantly more sample efficient than TDMPC2 but also require better trained world models in order to obtain high reward.

6 Conclusion

In this work, we introduced Policy learning with large World Models (PWM), a novel MBRL approach that utilizes large multi-task world models as differentiable physics simulators for efficient policy training using First-order Gradients (FoG). Our evaluations demonstrated that PWM can learn policies with higher rewards than existing methods, even if they have access to simulation ground-truth models. Furthermore, the PWM framework paves a pathway to scalibly learn high-performing policies from large multi-task world models, achieving higher rewards than our main baseline TDMPC2 without the need for online planning.

Limitations. Despite its demonstrated efficacy, PWM has notable limitations. Firstly, performance relies heavily on the availability of substantial pre-existing data to train the world model, which might not always be feasible, especially in novel or low-data environments. Secondly, although PWM facilitates fast and cost-effective policy training, it necessitates re-training for each new task, which could limit its applicability in scenarios requiring rapid adaptation to diverse tasks. Lastly, the current TDMPC2 world models used are difficult to train at scale due to their autoregressive formulation.

In summary, PWM pushes the boundaries of multi-task policy learning from world models. Future research directions can explore learning from images and

A closing remark on societal impact. Our proposed method and contributions are foundational in nature and not likely to have any immediate negative societal impact or risks. However, we acknowledge that embodied AI agents in general pose substantial challenges to society, including but not limited to: disruption of the labor market, safe and reliable deployment of robots alongside humans, and potential for misuse by bad actors.

References

[1] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[5] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

[6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[9] Y. Feng, N. Hansen, Z. Xiong, C. Rajagopalan, and X. Wang. Finetuning offline world models in the real world. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, 2023.

[10] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *arXiv preprint arXiv:2312.07843*, 2023.

[11] I. Georgiev, K. Srinivasan, J. Xu, E. Heiden, and A. Garg. Adaptive horizon actor-critic for policy learningin contact-rich differentiable simulation. In *International Conference on Machine Learning*. PMLR, 2024.

[12] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[13] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[14] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[15] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[16] N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control. 2024.

[17] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. 2022.

[18] T. A. Howell, S. Le Cleac'h, J. Z. Kolter, M. Schwager, and Z. Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 9, 2022.

[19] A. Hu, L. Russell, H. Yeo, Z. Murez, G. Fedoseev, A. Kendall, J. Shotton, and G. Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

[20] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 38–44. IEEE, 2016.

[21] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

[22] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[23] V. Konda and J. Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[24] A. Kumar, J. Hong, A. Singh, and S. Levine. Should i run offline reinforcement learning or behavioral cloning? In *International conference on learning representations*, 2021.

[25] M. Lee, J. Lee, and D. Lee. Differentiable dynamics simulation using invariant contact mapping and damped contact force. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11683–11689. IEEE, 2023.

[26] M. Ma, T. Ni, C. Gehring, P. D'Oro, and P.-L. Bacon. Do transformer world models give better policy gradients? *arXiv preprint arXiv:2402.05290*, 2024.

[27] L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

[28] D. Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[30] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

[31] R. M. Murray and J. E. Hauser. *A case study in approximate linearization: The acrobat example*. Electronics Research Laboratory, College of Engineering, University of . . . , 1991.

[32] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[33] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[34] P. Parmas, T. Seno, and Y. Aoki. Model-based reinforcement learning with scalable composite policy gradient estimators. In *International Conference on Machine Learning*, pages 27346–27377. PMLR, 2023.

[35] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[36] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[37] I. Schubert, J. Zhang, J. Bruce, S. Bechtle, E. Parisotto, M. Riedmiller, J. T. Springenberg, A. Byravan, L. Hasenclever, and N. Heess. A generalist dynamics model for control. *arXiv preprint arXiv:2305.10912*, 2023.

[38] J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[40] H. J. Suh, M. Simchowitz, K. Zhang, and R. Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pages 20668–20696. PMLR, 2022.

[41] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[42] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[43] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[44] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2021.

[45] Y. Xu, N. Hansen, Z. Wang, Y.-C. Chan, H. Su, and Z. Tu. On the feasibility of cross-task transfer with model-based reinforcement learning. 2023.

[46] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W.-C. Ma, A. J. Yang, and R. Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023.

[47] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[48] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023.

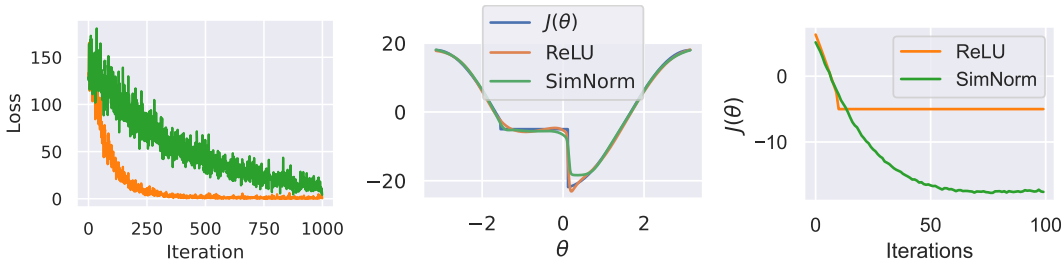


Figure 10: Extended ball-wall toy example results. The **left** figure shows the model losses as they are trained to approximate the target function $J(\theta$. The middle figure shows the output of the trained model across the spectrum of θ as well as the true target. This is the function we attempt to minimize in the **right** figure. We can see that when using the MLP with ReLU activation functions, the optimizer quickly gets stuck in local minima while the model using SimNorm activation function is able to find a solution closer to the true one.

A Ball-wall example details

This section provides more details on the ball-wall example used to showcase the issues of optimizing through contact in Section 4.1. In constructing this toy example we chose a simple physical system that exhibits contact discontinuities. Inspired by Suh et al. [40], we constructed a simple problem of a point mass (ball) being thrown forward (x direction) at a fixed velocity v. The optimization parameter of interest is the initial angle θ and the goal is to maximize forward distance traveled (in 2D). For simplicity we assume that the ball sticks to the wall (without complex contact) which can be expressed as:

$$x = f(\theta) = \begin{cases} x_0 + v\cos(\theta)t + \frac{1}{2}gt^2 & \text{if } y_{\text{contact}} > h \\ w & else \end{cases} \tag{11}$$

where $g = 9.81$ is gravity, h and w are the height and width of the wall, (x_0, y_0) is the starting position, $v = 10$ is the starting velocity and $t = 2$ is time. y_{contact} is the height at the time of contact t_{contact} which are both given by solving Eq. 11 for $f(\theta) = w$:

$$t_{\text{contact}} = \frac{-vcos(\theta) + \sqrt{v^2\cos^2(\theta) + aw}}{a} \qquad y_{\text{contact}} = y_0 + v\sin(\theta)t + \frac{1}{2}gt^2$$

We visualize the toy example in Figure 9 to aid reading. With $f(\theta)$ defined, we attempt to learn it with two Multi-Layer Perceptrons (MLPs). We configure them to have 2 hidden layers of 32 neurons each. The first MLP uses ReLU activation functions, while the latter uses SimNorm activation functions as defined in Eq. 4. Both models are initialized with identical random parameters and are trained with the ADAM optimizer with learning rate $\alpha = 2 \times 10^{-3}$ for 100 epochs using a batch size of $B = 50$. The data we use to train the models was 1000 uniform samples of $f(\theta)$ within $\theta \in [-\pi, \pi]$. Figure 9 shows the training losses of the models, induced optimization landscapes and the losses when attempting to maximize the models as you would do in an RL setting.

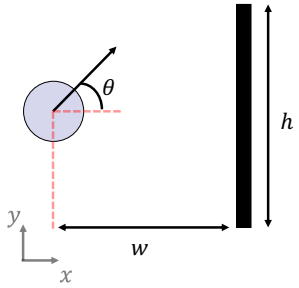


Figure 9: Pedagogical ball-wall toy problem visualized.

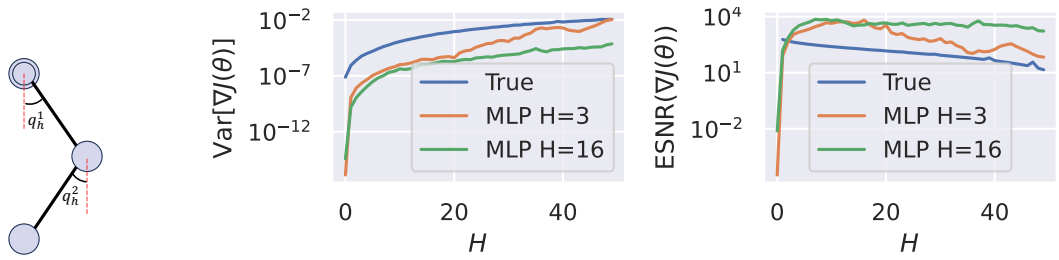


Figure 11: Double pendulum pedagogical example. The middle figure evaluates the variance of policy gradient estimates over $N = 100$ Monte-Carlo samples for varying horizons H. The right figure shows the same data but plots the Expected Signal-to-Noise ratio (ESNR) with higher values translating to more useful gradients. These results suggests that world models trained over long horizon trajectories provide more useful gradients.

B Double pendulum example details

The double pendulum (also known as Acrobot [31]) is a classic under-actuated chaotic system. It is characterized by its sensitivity to initial conditions where even small perturbations result in large gradient variance with long horizon (> 20) trajectories. We chose this system to analyze variance and expected signal-to-noise ratio (ENSR) in Section 4.2 as it is the easiest problem exhibiting chaosness. We model this toy problem similar to DMControl [43] in our differentiable simulator, dflex as we need ground truth gradients for comparison. The first link with angle θ_1 is fixed to the base and not actuated. The second link with angle θ_2 is the only control input via $\dot{\theta}_2$. The state of the system id calculated as:

$$\boldsymbol{s} = [\cos(\theta_1), \sin(\theta_1), \cos(\theta_2), \sin(\theta_2), \dot{\theta}_1, \dot{\theta}_2]$$

The objective of this toy example is to bring and balance the pendulum upwards which we achieve by formulating a reward:

$$r(\boldsymbol{s}, \boldsymbol{a}) = -\theta_1^2 - \theta_2^2 - 0.1\dot{\theta}_2^2$$

Next we train world models to approximate the dynamics and reward above. For this we collect data with the SHAC algorithm [44] over 3 different runs for a total of 24,000 episodes of length 240 timesteps. Maximum episode reward achieved during data collection was -942.95. Then we train two TDMPC2 [16] world models on the collected data. We use the 5M parameter model which features a latent state of dimension of 512, encoder E_ϕ with one hidden dimension of 256, dynamics model MLP with 2 hidden layers with 512 neurons and a rewards model of the same design. We keep the same hyper-parameters as per the origin work by Hensen et al. but use $\gamma = 0.99$ which we found to reduce variance substantially. We train two models with different training horizons $H = 3$ and $H = 16$ for 100k batch samples and a batch size of 1024.

With the trained models, we now compare the variance of stochastic gradients provided by the true dynamics of the simulation and the two trained models. We do this by loading the best policy learned by SHAC during data collection and executing a $H = 50$ rollout across the 3 models. We ensure that the same actions are taken for each evaluated models and collect 100 Monte Carlo samples. In addition to variance, we report ESNR as suggested by [34] and defined in 5. Higher ESNR translate to more useful gradients and we naturally should expect values to decrease with increased H. We reported the results in Figure 3 but also duplicate them in Figure 11 for convenience and ease of reading.

C Implementation details and hyper-parameters

The section details several implementation details of PWM that we thought are not crucial for understanding the proposed approach in Section 4.3 but are important for replicating the results.

1. **Reward binning** - the reward model we use in PWM is formulated as a discrete regression problem where $\mathbb{R}$ rewards are discretized into a predefined number of bins. Similar to [16, 25], we do this to enable robustness to reward scale and multi-task-ness. In particular, we perform two-hot encoding using SymLog and SymExp operators which are mathematically defined as:

$$\text{SymLog}(x) = \text{sign}(x)\log(1 + |x|) \qquad \text{SymExp}(x) = \text{sign}(x)(e^{|x|} - 1)$$

   Two-hot encoding is then performed with:

```
def two_hot(x):
    x = clamp(symlog(x), vmin, vmax)
    bin_idx = floor((x - vmin) / bin_size)
    bin_offset = (x - vmin) / bin_size - bin_idx
    soft_two_hot = zeros(x.size(0), num_bins)
    soft_two_hot[bin_idx] = 1 - bin_offset
    soft_two_hot[bin_odx + 1] = bin_offset
    return soft_two_hot
```

   Inverting this operation to get back to scalar rewards would usually involve $\text{SymExp}(x)$ but note that the $\text{sign}(x)$ operator is not differentiable and would therefore not work for FoG. Instead, we chose to omit the $\text{SymExp}(x)$ operation which technically now returns pseudo-rewards but also gradients which we found sufficient for policy learning:

```
def two_hot_inversion(x):
    vals = linspace(vmin, vmax, num_bins)
    x = softmax(x)
    x = torch.sum(x * vals, dim=-1)
    return x
```

2. **Critic training** - while Algorithm 1 function to similar results as presented in 5, we found it beneficial to split the critic training data from a single rollout into several smaller mini-batches and over them for multiple gradient steps. In our implementation we split the data into 4 mini-batches and perform 8 gradient steps over them with uniform sampling. With a $H = 16$ and batch size 64, this translates to a critic batch size of 256.
3. **Minimum policy noise** - Due to the larger amount of gradient steps needed, we noticed that PWM's actor tends to collapse to a deterministic policy rapidly. As such, we found it beneficial to include a lower bound on the standard deviation of the action distribution in order to maintain stochasticity in the optimization process. We have used 0.24 throughout this paper. While similar results would be possible by adding an entropy term [39], we found our current solution sufficient
4. **World model fine-tuning** - Throughout all of our experiments we found that the offline data used to train PWM's world model to be crucial to learning a good policy. In very high-dimensional tasks such as Humanoid SNU, collecting extensive data is a difficult task. As such, in these tasks we found it beneficial to online fine-tune the world model. We do this on all single-task experiments of Section 5.1 using the default hyper-parameters and a replay buffer of size 1024.

D Contact-rich single task experiment details

In Section 5.1, we explore 5 locomotion tasks with increasing complexity. They are described below and shown in Figure 4.

1. **Hopper**, a single-legged robot jumping only in one axis with $n = 11$ and $m = 3$.
2. **Ant**, a four-legged robot with $n = 37$ and $m = 8$.

Hyper-parameter	Value
Policy components	
Horizon (H)	16
Batch size	64
$\alpha_{\boldsymbol{\theta}}$	5×10^{-4}
$\alpha_{\boldsymbol{\psi}}$	5×10^{-4}
Actor grad norm	1
Critic grad norm	100
Actor hidden layers	$[400, 200, 100]$
Critic hidden layers	$[400, 200]$
Number of critics	3
λ	0.95
γ	0.99
Critic batch split	4
Critic iterations	8
World model components (48M)	
Latent state ($\boldsymbol{z}$) dimension	768
Horizon (H)	16
Batch size	1024
$\alpha_{\boldsymbol{\phi}}$	3×10^{-4}
World model grad norm	20.0
SimNorm V	8
Reward bins	101
Encoder $E_{\boldsymbol{\phi}}$ hidden layers	$[1792, 1792, 1792]$
Dynamics $F_{\boldsymbol{\phi}}$ hidden layers	$[1792, 1792]$
Reward $R_{\boldsymbol{\phi}}$ hidden layers	$[1792, 1792]$
Task encoding dimension	96

Table 1: Table of hyper-parameters used in PWM, shared across all tasks.

Figure 12: Locomotion environments (left to right): Hopper, Ant, Anymal, Humanoid and SNU Humanoid.

3. **Anymal**, a more sophisticated quadruped with $n = 49$ and $m = 12$ modeled after [20].
4. **Humanoid**, a classic contact-rich environment with $n = 76$ and $m = 21$ which requires extensive exploration to find a good policy.
5. **SNU Humanoid**, a version of Humanoid lower body where instead of joint torque control, the robot is controlled via $m = 152$ muscles intended to challenge the scaling capabilities of algorithms.

All tasks share the same common main objective - maximize forward velocity v_x:

We additionally use auxiliary rewards R_{height} to incentivise the agent to, R_{angle} to keep the agents' normal vector point up, $R_{heading}$ to keep the agent's heading pointing towards the direction of running and a norm over the actions to incentivise energy-efficient policies. For most algorithms, none of these rewards apart from the last one are crucial to succeed in the task. However, all of them aid learning policies faster.

$$R_{height} = \begin{cases} h - h_{term} & if h \geq h_{term} \\ -200(h - h_{term})^2 & if h < h_{term} \end{cases}$$

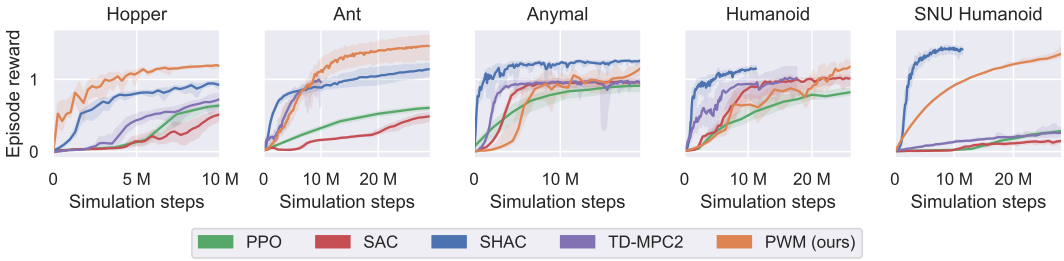


Figure 13: Learning curves for each environment. This figure shows 50% IQM and 95% CI across 5 random seeds for each task in the dflex simulation suite. Rewards are normalized by the maximum reward achieved by PPO (usually $\geq$ 100M steps). While PWM remains competitive with SHAC for most tasks, it does not scale well to the 152 action dimension SNU Humanoid.

Environment	Reward
Hopper	$v_x + R_{height} + R_{angle} - 0.1\|\boldsymbol{a}\|_2^2$
Ant	$v_x + R_{height} + 0.1R_{angle} + R_{heading} - 0.01\|\boldsymbol{a}\|_2^2$
Anymal	$v_x + R_{height} + 0.1R_{angle} + R_{heading} - 0.01\|\boldsymbol{a}\|_2^2$
Humanoid	$v_x + R_{height} + 0.1R_{angle} + R_{heading} - 0.002\|\boldsymbol{a}\|_2^2$
Humanoid STU	$v_x + R_{height} + 0.1R_{angle} + R_{heading} - 0.002\|\boldsymbol{a}\|_2^2$

Table 2: Rewards used for each task bench-marked in Section 5

$$R_{angle} = 1 - \left(\frac{\theta}{\theta_{term}}\right)^2$$

$R_{angle} = \|\boldsymbol{q}_{forward} - \boldsymbol{q}_{agent}\|_2^2$ is the difference between the heading of the agent $\boldsymbol{q}_{agent}$ and the forward vector $\boldsymbol{q}_{agent}$. h is the height of the CoM of the agent and θ is the angle of its normal vector. h_{term} and θ_{term} are parameters that we set for each environment depending on the robot morphology. Similar to other high-performance RL applications in simulation, we find it crucial to terminate episode early if the agent exceeds these termination parameters. However, it is worth noting that AHAC is still capable of solving all tasks described in the paper without these termination conditions, albeit slower.

All results presented in Figure 13 are for 5 random seeds using the simulator in a vectorized fashion with 64 parallel environments for all approaches, except PPO which uses 1024. We note that while simulation steps appear high, all of these experiments are executed $\leq$ 2 hours on an Nvidia RTX6000 GPU. In addition the the learning curves of Figure 13, we also present tabular results below:

	Hopper	Ant	Anymal	Humanoid	SNU Humanoid
PPO	1.00 ± 0.11	1.00 ± 0.12	1.00 ± 0.03	1.00 ± 0.05	1.00 ± 0.09
SAC	0.87 ± 0.16	0.95 ± 0.08	0.98 ± 0.06	1.04 ± 0.04	0.88 ± 0.11
TDMPC2	0.85 ± 0.37	1.0 ± 0.49	0.98 ± 0.48	1.03 ± 0.45	0.26 ± 0.12
SHAC	1.02 ± 0.03	1.16 ± 0.13	1.26 ± 0.04	1.15 ± 0.04	1.44 ± 0.08
PWM	1.20 ± 0.29	1.46 ± 0.31	1.16 ± 0.24	1.19 ± 0.025	1.08 ± 0.24

Table 3: Tabular results of the asymptotic rewards achieved by each algorithm across all tasks. The results presented are PPO-normalised 50 % IQM and standard deviation across 5 random seeds. Most algorithms have been trained until convergence.

We note that TDMPC2 and PWM use pre-trained world models on 20480 episodes of each task. The world models are trained for 100k gradient steps and the same world models (specific to each task) are loaded into both approaches. The data consists of trajectories of varying policy quality generated with the SHAC algorithm. Trajectories include near-0 episode rewards as well as the highest reward achieved by SHAC. Note that we also run an early termination mechanism in these tasks which is done to accelerate learning and iteration.

	Hopper	Ant	Anymal	Humanoid	SNU Humanoid
PPO	4742 ± 521	6605 ± 793	12029 ± 360	7293 ± 365	4114 ± 370
SAC	4126 ± 759	6275 ± 528	11788 ± 722	7285 ± 292	3620 ± 453
TDMPC2	4027 ± 1768	6591 ± 2708	11787 ± 4702	7476 ± 3268	1121 ± 525
SHAC	4837 ± 142	7662 ± 859	15157 ± 481	8387 ± 292	5924 ± 329
PWM	5680 ± 2303	9672 ± 2012	13927.74 ± 2882	8661 ± 1792	4585 ± 958

Table 4: Tabular results of the asymptotic (end of training) rewards achieved by each algorithm across all tasks. The results presented are 50 % IQM and standard deviation across 10 random seeds. All algorithms have been trained until convergence.

E Multi-task experiments additional results

In this section we provide additional results on multi-task experiments. While we find it beneficial to train the world model at the same horizon as the policy learning, it is not strictly necessary to achieve good performance. In Figure 14 we present an ablation where we compare PWM world models pre-trained on horizons $H = 3$ and $H = 16$ and policies trained only with $H = 16$. These results reveal that $H = 16$ trained world models have only marginally higher scores. On deeper inspection, most of increased scores come form dm_control tasks which are harder than MetaWorld tasks on average. Therefore if training new world models, we advise using higher H; however if other pre-trained world models exist with suboptimal H, they will probably be also useful.

Figure 14: Horizon ablation of the world model

Figures 15 and 16 give scores for individual tasks for TDMPC2 and PWM across both the MT30 and MT80 task sets. We can observe that most of the increased performance of PWM is in dm_control tasks which are on average more difficult than MetaWorld.

F TD(λ) ablation

Here we replace the actor objective of PWM with TD(λ)l, instead of TD(N). We evaluate this on 3 single-task experiments - Hopper, Ant and Anymal. Figure 17 shows the results using 3 random seeds and 50% IQM. We can see that PWM overall achieves higher rewards, which we believe is to more simple and less noisy gradients obtained via TD(N). Additionally, we found that TD(λ) uses approx. 10% more computation due to having to compute more gradients. However, it is worth noting that TD(λ) learning curves appear more stable with higher dimensional tasks such as Anymal and we believe this ablation requires more large-scale studying.

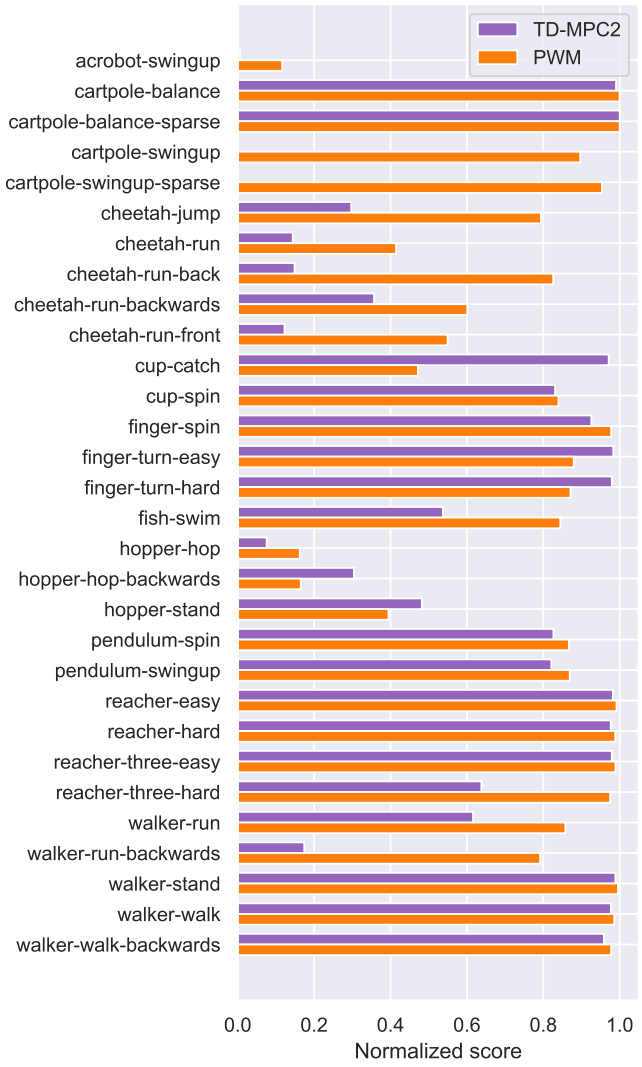


Figure 15: Individual task results for MT30 task set.

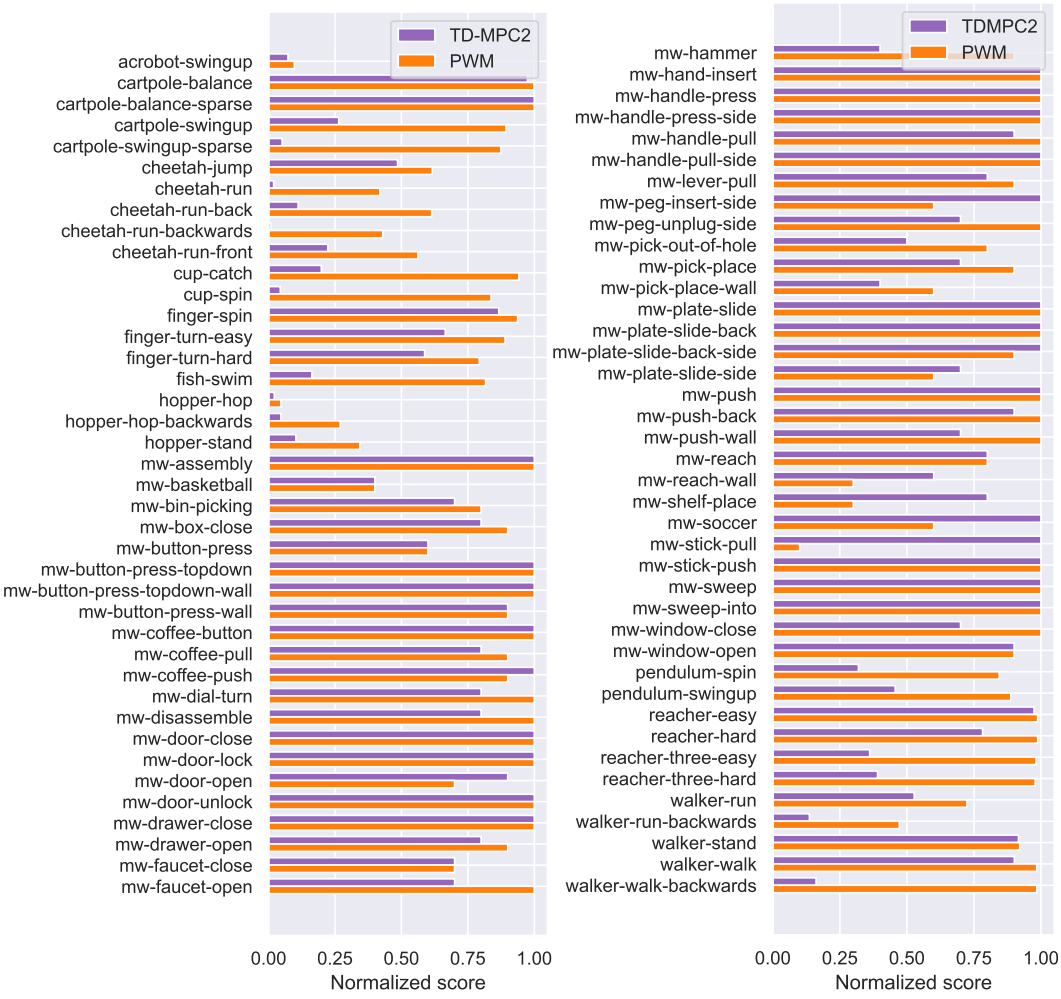


Figure 16: Individual task results for MT80 task set.

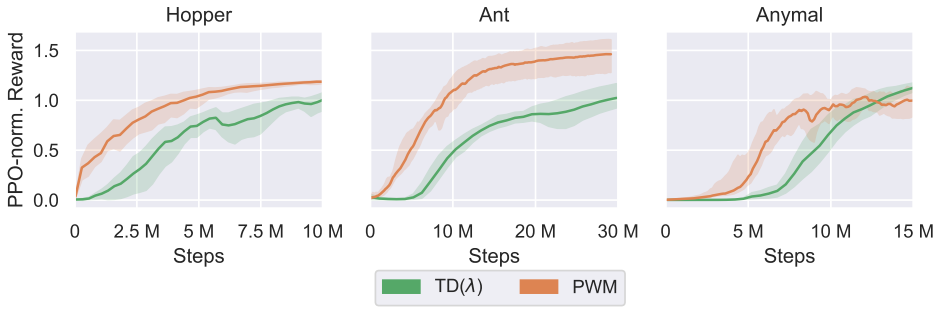


Figure 17: TD(λ) ablation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, we believe that our main claims are supported by the experimental results in Section 5: our method learns better policies than baselines in both single-task and multi-task settings.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of our method in detail in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Our contributions are empirical in nature; we do not have any notable theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Yes. We base our experiments on publicly available simulation environments, models, and datasets, and we provide extensive implementation details in the appendices.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Yes. As mentioned previously, we primarily use publicly available simulation environments, models, and datasets, and we provide extensive implementation details in the appendices. We are committed to releasing code that reproduces all of our main experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide extensive implementation details in the appendices, including all relevant hyperparameters.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report error bars to communicate statistical significance whenever appropriate, as well as rliable [1] metrics.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide detailed information about computing resources and training times for every major experiment. A key selling point of our method is its relatively small computational requirements.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and firmly believe that our research conforms in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our contributions are foundational in nature and unlikely to have any immediate societal impact. However, we do include limited discussion on this in Section 6.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not believe that our models pose any risk for misuse at this time.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We use multiple external assets in this work, all of which are properly credited. Licenses are respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our main contributions are algorithmic, but described in detail and we are committed to releasing well-documented code alongside our paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not conduct research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not conduct research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.