PWM: Policy Learning with Multi-Task World Models

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement Learning (RL) has made significant strides in complex tasks but struggles in multi-task settings with different embodiments. World models methods offer scalability by learning a simulation of the environment, but often rely on inefficient gradient-free optimization methods for policy extraction. In contrast, gradient-based methods exhibit lower variance but fail to handle discontinuities. Our work reveals that well-regularized world models can generate smoother optimization landscapes than the actual dynamics, facilitating more effective first-order optimization. We introduce Policy learning with multi-task World Models (PWM), a novel model-based RL algorithm for continuous control. Initially, the world model is pre-trained on offline data, and then policies are extracted from it using first-order optimization in less than 10 minutes per task. PWM effectively solves tasks with up to 152 action dimensions and outperforms methods that use ground-truth dynamics. Additionally, PWM scales to an 80-task setting, achieving up to 27% higher rewards than existing baselines, without relying on costly online planning. Visualizations and code available at `policy-world-model.github.io`.

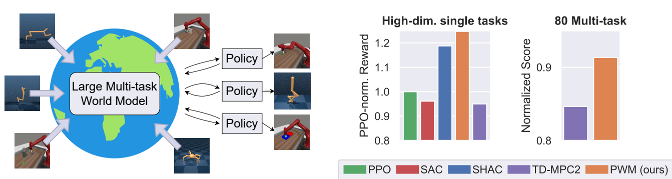


Figure 1: We prpose PWM, a new method for multi-task RL that utilizes pre-trained world models to learn policies for each task. When sufficiently regularized, these world models induce smooth optimization landscapes, which allows for efficient first-order optimization. Our approach can solve tasks in <10 minutes and achieves higher rewards in both single-task and multi-task environments.

1 Introduction

The pursuit of generalizability in machine learning has recently been propelled by the training of large models on substantial datasets (Brown et al., 2020; Kirillov et al., 2023; Bommasani et al., 2021). Such advancements have notably permeated robotics, where multi-task behavior cloning techniques have shown remarkable performance (Zitkovich et al., 2023; Octo Model Team et al., 2024; Goyal et al., 2023; Bousmalis et al., 2023). Nevertheless, these approaches predominantly hinge on near-expert data and struggle with adaptability across diverse robot morphologies due to their dependence on teleoperation (Zitkovich et al., 2023; Octo Model Team et al., 2024; Kumar et al., 2021).

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 (Mnih et al., 2013; Schulman et al., 2017; Haarnoja et al., 2018). Recently, (Hansen et al., 2024) suggested that a potential pathway to multi-task RL is with the world

PWM: Policy Learning with Multi-Task World Models

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement Learning (RL) has made significant strides in complex tasks but struggles in multi-task settings with different embodiments. World models methods offer scalability by learning a simulation of the environment, but often rely on inefficient gradient-free optimization methods for policy extraction. In contrast, gradient-based methods exhibit lower variance but fail to handle discontinuities. Our work reveals that well-regularized world models can generate smoother optimization landscapes than the actual dynamics, facilitating more effective first-order optimization. We introduce Policy learning with multi-task World Models (PWM), a novel model-based RL algorithm for continuous control. Initially, the world model is pre-trained on offline data, and then policies are extracted from it using first-order optimization in less than 10 minutes per task. PWM effectively solves tasks with up to 152 action dimensions and outperforms methods that use ground-truth dynamics. Additionally, PWM scales to an 80-task setting, achieving up to 27% higher rewards than existing baselines, without relying on costly online planning. Visualizations and code available at `policy-world-model.github.io`.

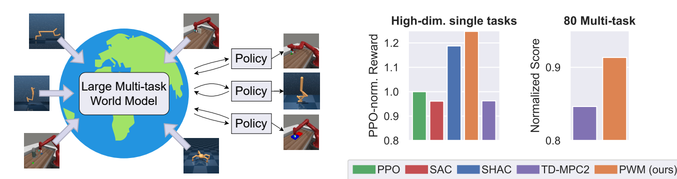


Figure 1: We propose PWM, a new method for multi-task RL that utilizes pre-trained world models to learn policies for each task. When sufficiently regularized, these world models induce smooth optimization landscapes, which allows for efficient first-order optimization. Our approach can solve tasks in <10 minutes and achieves higher rewards in both single-task and multi-task environments.

1 Introduction

The pursuit of generalizability in machine learning has recently been propelled by the training of large models on substantial datasets Brown et al. (2020); Kirillov et al. (2023); Bommasani et al. (2021). Such advancements have notably permeated robotics, where multi-task behavior cloning techniques have shown remarkable performance Zitkovich et al. (2023); Octo Model Team et al. (2024); Goyal et al. (2023); Bousmalis et al. (2023). Nevertheless, these approaches predominantly hinge on near-expert data and struggle with adaptability across diverse robot morphologies due to their dependence on teleoperation Zitkovich et al. (2023); Octo Model Team et al. (2024); Kumar et al. (2021).

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 Mnih et al. (2013); Schulman et al. (2017); Haarnoja et al. (2018). Recently, Hansen et al. (2024) suggested that a potential pathway to multi-task RL is with the world models framework, where a large model learns the environment dynamics and is then combined with Zeroth-order Gradient (ZoG) methods. Despite advancements, ZoG methods struggle with sample inefficiency due to the high variance (Mohamed et al., 2020; Suh et al., 2022; Parmas et al., 2023) and online planning time scales with model size, rendering it infeasible at scale.

First-order Gradient (FoG) methods provide a low-variance alternative that have shown superior sample efficiency and asymptotic performance when combined with smooth differentiable simulations (Xu et al., 2022; Georgiev et al., 2024). However, they struggle to optimize through discontinuities (Suh et al., 2022; Georgiev et al., 2024). In this work, we explore the tight coupling between FoG optimization and world models through the lens of differentiable simulation. Counter-intuitively, we find that for gradient-based optimization, we don't want world models to be accurate; instead, we want them to be smooth and have a low optimality gap. This in turn enables efficient FoG optimization.

Building on these insights, we propose Policy learning with multi-task World Models (PWM), an algorithm that can learn policies from offline pre-trained world models in under <10 minutes per task. With this new-found efficiency, we also propose a *new multi-task framework*, where instead of training a full multi-task algorithm, we only train a multi-task world model and then extract a policy for each task. This decoupling of the supervised objective and the RL objective results in more stable, more efficient learning and higher episode rewards. Our empirical evaluations on high-dimensional tasks indicate that PWM not only achieves higher reward than baselines but also outperforms methods that use ground-truth dynamics. In a multi-task scenario utilizing a pre-trained 48M parameter world model from TD-MPC2, 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 result in better 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 methods.
3. We propose PWM, a model-based algorithm for learning continuous control policies from pre-trained multi-task world models that can solve tasks in <10 minutes using FoG optimization.

2 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 (Konda & Tsitsiklis, 1999). 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 $E_{\boldsymbol{\phi}}$ (Hafner et al., 2019; Hansen et al., 2022; 2024;

Under review as a conference paper at ICLR 2025

models framework, where a large model learns the environment dynamics and is then combined with Zeroth-order Gradient (ZoG) methods. Despite advancements, ZoG methods struggle with sample inefficiency due to the high variance Mohamed et al. (2020); Suh et al. (2022); Parmas et al. (2023) and online planning time scales with model size, rendering it infeasible at scale.

First-order Gradient (FoG) methods provide a low-variance alternative that have shown superior sample efficiency and asymptotic performance when combined with smooth differentiable simulations Xu et al. (2022). However, they struggle to optimize through discontinuities Suh et al. (2022); Georgiev et al. (2024). In this work, we explore the tight coupling between FoG optimization and world models through the lens of differentiable simulation. Counter-intuitively, we find that for gradient-based optimization, we don't want world models to be accurate; instead, we want them to be smooth and have a low optimality gap. This in turn enables efficient FoG optimization.

Building on these insights, we propose Policy learning with multi-task World Models (PWM), an algorithm that can learn policies from offline pre-trained world models in under <10 minutes per task. With this new-found efficiency, we also propose a *new multi-task framework*, where instead of training a multi-task algorithm, we only train a multi-task world model and then extract a policy for each task. This decoupling of the supervised and RL objectives results in more stable and efficient learning with higher episode rewards. Our empirical evaluations on high-dim. tasks indicate that PWM not only achieves higher reward than baselines but also outperforms methods that use ground-truth dynamics. In a multi-task scenario utilizing a pre-trained 48M parameter world model from TD-MPC2, 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. **Correlation Between World Model Smoothness and Policy Performance:** Through pedagogical examples and ablations, we demonstrate that smoother, better-regularized world models significantly enhance policy performance. Notably, this results in an inverse correlation between model accuracy and policy performance.
2. **Efficiency of First-Order Gradient (FoG) Optimization:** We show that combining FoG optimization with well-regularized world models enables more efficient policy learning compared to zeroth-order methods. Furthermore, policies learned from world models asymptotically outperform those trained with ground-truth simulation dynamics, emphasizing the importance of the tight relationship between FoG optimization and world model design.
3. **Scalable Multi-Task Algorithm:** Instead of training a single multi-task policy model, we propose PWM, a framework where a multi-task world model is first pre-trained on offline data. Then per-task expert policies are extracted in <10 minutes per task, offering a clear and scalable alternative to existing methods focused on unified multi-task models.

2 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:

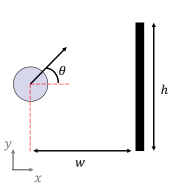
$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) := \max_{\boldsymbol{\theta}} \mathbb{E}_{\substack{\boldsymbol{s}_1 \sim \rho(\cdot) \\ \boldsymbol{a}_t \sim \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}_t)}} \left[\sum_{t=1}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right] \tag{1}$$

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 $\pi_{\boldsymbol{\theta}}$

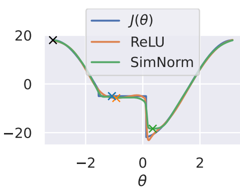
$$V^\pi_{\boldsymbol{\psi}}(\boldsymbol{s}_t) := \mathbb{E}_{\boldsymbol{a}_h \sim \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}_h)} \left[\sum_{h=t}^{\infty} \gamma^h r(\boldsymbol{s}_h, \boldsymbol{a}_h) \right] \tag{2}$$

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 Konda & Tsitsiklis (1999). 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

2



(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 $\hat{\theta}$.

Hafner et al., 2023). 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 (Hafner et al., 2019; 2023) and TD-MPC2 (Hansen et al., 2024). 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*.

3 Policy optimization through 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 (Mohamed et al., 2020). 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 (Kingma et al., 2015) and pathwise derivative (Schulman et al., 2015). 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.

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

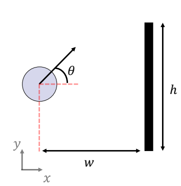
As $\nabla^{[1]}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ in itself is a random variable, we need to estimate it. A popular way to do that in practice is via Monte-Carlo approximation where we are interested in two properties - bias and variance. In Sections 3.1 and 3.2 we tackle each aspect with toy robotic control problem to build intuition. In Section 3.3 we combine our findings to propose a new algorithm.

3.1 Learning through contact

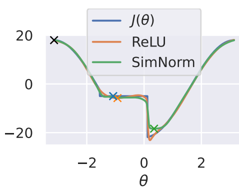
FoGs are unbiased $\mathbb{E}\left[\bar{\nabla}^{[1]} J(\boldsymbol{\theta})\right] := \mathbb{E}\left[\sum_{n=1}^{N} \hat{\nabla}^{[1]}_n J(\boldsymbol{\theta})\right] = \nabla J(\boldsymbol{\theta})$, only if both the dynamics f and rewards r are Lipschitz-smooth (Suh et al., 2022). However, many robotic 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

Under review as a conference paper at ICLR 2025



(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 $\hat{\theta}$.

$\boldsymbol{s}$ into a latent state $\boldsymbol{z}$ using a learned encoder $E_{\boldsymbol{\phi}}$ Hafner et al. (2019); Hansen et al. (2022; 2024); Hafner et al. (2023). 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 Hafner et al. (2019; 2023) and TD-MPC2 Hansen et al. (2024). 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*.

3 Policy optimization through 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 Mohamed et al. (2020). 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 Kingma et al. (2015) and pathwise derivative Schulman et al. (2015). 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 3.1 and 3.2 we tackle each aspect with toy robotic control problem to build intuition. In Section 3.3 we combine our findings to propose a new algorithm.

3.1 Learning through contact

FoGs are unbiased $\mathbb{E}\left[\bar{\nabla}^{[1]} J(\boldsymbol{\theta})\right] := \mathbb{E}\left[\sum_{n=1}^{N} \hat{\nabla}^{[1]}_n J(\boldsymbol{\theta})\right] = \nabla J(\boldsymbol{\theta})$, only if both the dynamics f and rewards r are Lipschitz-smooth Suh et al. (2022). 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

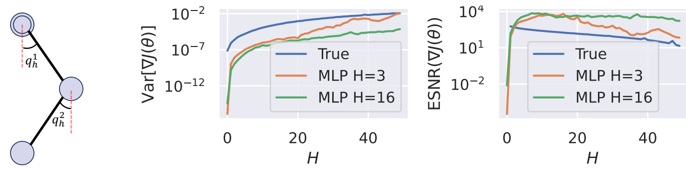


Figure 3: Double pendulum pedagogical example. The middle figure evaluates the variance of policy gradient estimates over $N = 100$ Monte-Carlo samples for varying horizons H. The right figure shows the same data but plots the Expected Signal-to-Noise ratio (ESNR) with higher values translating to more 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.

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 (Hansen et al., 2024) 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:

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

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). This inverse correlation between the optimality gap and model error is known as objective mismatch (Lambert et al., 2020). 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.

3.2 Learning with chaotic dynamics

While FoGs have lower variance per step, they can accumulate significant variance over long-horizon rollouts (Metz et al., 2021). (Suh et al., 2022) link this variance to the smoothness of models and the length of the prediction horizon: $\mathrm{Var}\left[\nabla J^{[1]}\right] \propto \|\nabla f(s,a)\|^{2H}$. 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 (Murray & Hauser, 1991), is a classic example of such a system (Figure 3).

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 (Parmas et al., 2023), 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:

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

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.

Under review as a conference paper at ICLR 2025

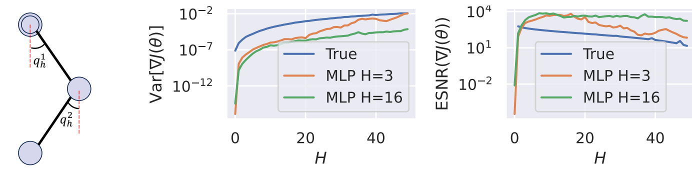


Figure 3: Double pendulum pedagogical example. The middle figure evaluates the variance of policy gradient estimates over $N = 100$ Monte-Carlo samples for varying horizons H. The right figure shows the same data but plots the Expected Signal-to-Noise ratio (ESNR) with higher values translating to more 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.

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 Hansen et al. (2024) in the intermediate layers. SimNorm normalizes a latent vector $\boldsymbol{z}$ by projecting it into simplices with dimension P using a softmax operator. Given an input vector $\boldsymbol{z}$, SimNorm can be expressed as a mapping into L vectors:

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

We trained both MLPs and observed their effects on smoothing the optimization landscape (Figure 2b). The ReLU-activated MLP smooths the landscape but introduces a local minimum that hinders gradient descent, particularly when starting from $\theta = -\pi$, resulting in a large optimality gap (difference between the solution and the optimal solution: $\|\hat{\theta} - \theta^*\|$). In contrast, the SimNorm-activated MLP has additional regularization which reduces the optimality gap, at the expense of model accuracy (Table 2c). This example highlights that more accurate models do not always lead to better policies, as noted by Lambert et al. (2020). Our findings extend this by showing that for FoG optimization, prioritizing smoothness over accuracy can lead to improved results. Further details are provided in Appendix A.

3.2 Learning with chaotic dynamics

While FoGs have lower variance per step, they can accumulate significant variance over long-horizon rollouts Metz et al. (2021). Suh et al. (2022) link this variance to the smoothness of models and the length of the prediction horizon: $\mathrm{Var}\left[\nabla J^{[1]}\right] \propto \|\nabla f(s,a)\|^{2H}$. 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 Murray & Hauser (1991), is a classic example of such a system (Figure 3).

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 Parmas et al. (2023), 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:

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

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.

3.3 PWM: Policy learning with multi-task World Models

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 (Hansen et al., 2024) do not exploit these properties but rather choose to optimize policies via DDPG-style gradients: $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \mathbb{E}_{\boldsymbol{a}\sim\pi(\cdot|s)}[\nabla_{\boldsymbol{\theta}} Q(\boldsymbol{s}, \boldsymbol{a})]$.

Algorithm 1: PWM: Policy learning with multi-task 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. 6`
  $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} - \alpha_{\boldsymbol{\psi}} \mathcal{L}_V(\boldsymbol{\psi})$ ▷ `Eq. 7-9`
end

We propose a new method and framework for efficiently 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.

Method. For policy learning, we propose on-policy actor-critic approach inspired by differentiable simulation approaches (Xu et al., 2022) 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 Eq. 1 but features rewards over a fixed horizon H, terminal value bootstrapping and usage of the learned world model components:

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

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 (Sutton & Barto, 2018; Hafner et al., 2019; Xu et al., 2022):

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

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

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

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 (Hansen et al., 2024) 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 (Ba et al., 2016) and Mish activation (Misra, 2019). The complete algorithm is shown in Algorithm 1. Further implementation details can be found in Appendix C.

Under review as a conference paper at ICLR 2025

3.3 PWM: Policy learning with multi-task World Models

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 Hansen et al. (2024) do not exploit these properties but rather choose to optimize policies via DDPG-style gradients: $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \mathbb{E}_{\boldsymbol{a}\sim\pi(\cdot|s)}[\nabla_{\boldsymbol{\theta}} Q(\boldsymbol{s}, \boldsymbol{a})]$.

Algorithm 1: PWM: Policy learning with multi-task 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**
  $\boldsymbol{s}_{1:H}, \boldsymbol{a}_{1:H}, r_{1:H}, \boldsymbol{e} \sim \mathcal{B}$
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha_{\boldsymbol{\phi}} \nabla \mathcal{L}_{wm}(\boldsymbol{\phi})$ ▷ `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
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \nabla \mathcal{L}_\pi(\boldsymbol{\theta})$ ▷ `Eq. 6`
  $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} - \alpha_{\boldsymbol{\psi}} \nabla \mathcal{L}_V(\boldsymbol{\psi})$ ▷ `Eq. 7-9`
end

We propose a new method and framework for efficiently 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.

Method. For policy learning, we propose on-policy actor-critic approach inspired by differentiable simulation approaches Xu et al. (2022) 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 Eq. 1 but features rewards over a fixed horizon H, terminal value bootstrapping and usage of the learned world model components:

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

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 Sutton & Barto (2018); Hafner et al. (2019); Xu et al. (2022):

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

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

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

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 Hansen et al. (2024) 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 Ba et al. (2016) and Mish activation Misra (2019). The complete algorithm is shown in Algorithm 1. Further implementation details can be found in Appendix C.

5

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

4 Experimental results

4.1 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 (Xu et al., 2022). Hopper, Ant, Anymal, Humanoid and muscle-actuated (SNU) Humanoid (Figure 4) are tasked to maximize forward velocity. We compare against SHAC (Xu et al., 2022), a method with a similar actor-critic architecture as PWM but uses ground-truth dynamics and rewards from the simulation, instead of learning them. 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 (Schulman et al., 2017) and SAC (Haarnoja et al., 2018).

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.

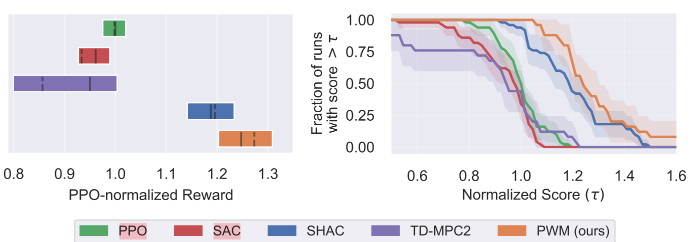


Figure 5: Aggregate results from high-dimensional locomotion tasks where each agent is trained to solve just that task (i.e. specialist). The **left** figure summarizes rewards achieved at the end of training using 50% IQM for the solid lines and 95% CI as suggested by (Agarwal et al., 2021), as well 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.

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

4 Experimental results

4.1 Contact-rich single tasks

The aim of this section is to understand whether smooth world models create better optimization landscapes than ground-truth dynamics, facilitating efficient FoG optimization. We study this on 5 continuous control tasks (Figure 4 with up to $\mathcal{A} = \mathbb{R}^{152}$ using the differentiable simulator dflex Xu et al. (2022). Comparisons include SHAC Xu et al. (2022), which uses ground-truth dynamics and rewards with a similar actor-critic architecture to PWM. Furthermore, we compare against two world model approaches. DreamerV3 Hafner et al. (2023) learns its world model via reconstruction, its actor via ZoG optimization and critic via Model-based Value Expansion (MVE) Feinberg et al. (2018). TD-MPC2 Hansen et al. (2024) uses the same world model as PWM but learns a policy in a model-free fashion and actively plans at inference time. We additionally include model-free baselines PPO Schulman et al. (2017) and SAC Haarnoja et al. (2018). This comparison allows us to understand whether (1) FoG-based optimization can learn better policies asymptotically and (2) whether smooth world models induce better optimization landscapes for FoG optimization.

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.

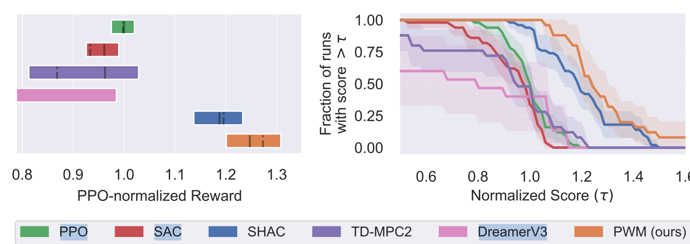


Figure 5: Aggregate results from high-dimensional locomotion tasks where each agent is trained to solve just that task (i.e. specialist). The **left** figure summarizes rewards achieved at the end of training using 50% IQM for the solid lines and 95% CI as suggested by Agarwal et al. (2021), as well 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. DreamerV3 results are incomplete.

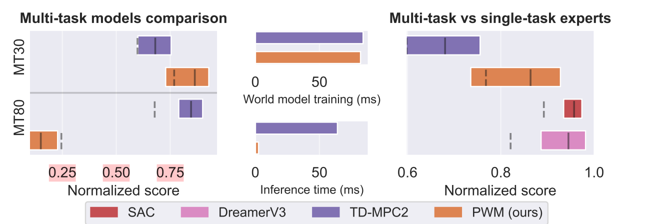


Figure 6: Multi-task results. The **left** figure shows results of multi-task agents in the 30 and 80 task set settings which include environments from dm_control (Tunyasuvunakool et al., 2020) and MetaWorld (Yu et al., 2020). The results show 50% IQM with the solid lines and mean with the dashed lines. 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. PWM has significantly lower inference time as it does not plan online. The **right** figure shows a comparison between multi-task PWM and TD-MPC2 and single-task experts SAC and DreamerV3 on the MT30 task set. Notably, PWM is able to match the performance of SAC and DreamerV3.

4.2 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 (Tunyasuvunakool et al., 2020) ranging from $m = 1$ to $m = 6$ and (2) 80 tasks which include 50 additional manipulation tasks from MetaWorld (Yu et al., 2020) with $n = 39$ and $m = 4$. These two multi-task settings were introduced as *MT30* and *MT80* by (Hansen et al., 2024). 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 (Hansen et al., 2024) with slight modifications to make them differentiable (Appendix C).

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 3.2. 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. While the fast per-task training is enabled by FoG optimization, we also find that training a single multi-task policy produces poor results as shown in Appendix F. We further compare our multi-task PWM policy to online-trained single-task experts SAC (Haarnoja et al., 2018) and DreamerV3 (Hafner et al., 2023). 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.

4.3 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 (Suh et al., 2022). We run the same experiment as Section 4.1, 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 4.1. 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 generalizlbe the differentiable simulations.

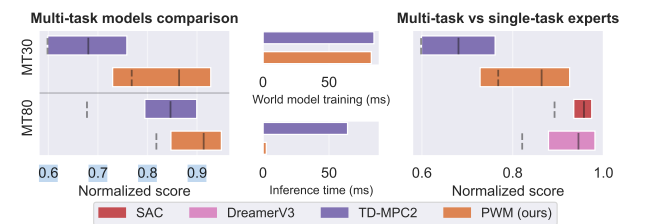


Figure 6: Multi-task results. The **left** figure shows results of multi-task agents in the 30 and 80 task set settings which include environments from dm_control Tunyasuvunakool et al. (2020) and MetaWorld Yu et al. (2020). The results show 50% IQM with the solid lines and mean with the dashed lines. 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. PWM has significantly lower inference time as it does not plan online. The **right** figure shows a comparison between multi-task PWM and TD-MPC2 and single-task experts SAC and DreamerV3 on the MT30 task set. Notably, PWM is able to match the performance of SAC and DreamerV3.

4.2 Multi-task world-model

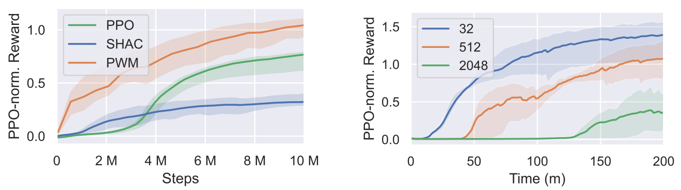
We analyze the scalability of our proposed framework and method to large multi-task pre-trained world models. We evaluate on two settings: (1) 30 continuous control dm_control tasks Tunyasuvunakool et al. (2020) ranging from $m = 1$ to $m = 6$ and (2) 80 tasks which include 50 additional manipulation tasks from MetaWorld Yu et al. (2020) with $n = 39$ and $m = 4$. These two multi-task settings were introduced as *MT30* and *MT80* by Hansen et al. (2024). 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 Hansen et al. (2024) with slight modifications to make them differentiable (Appendix C).

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 3.2. 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. While the fast per-task training is enabled by FoG optimization, we also find that training a single multi-task policy produces poor results as shown in Appendix F. We further compare our multi-task PWM policy to online-trained single-task experts SAC Haarnoja et al. (2018) and DreamerV3 Hafner et al. (2023). 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.

4.3 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.

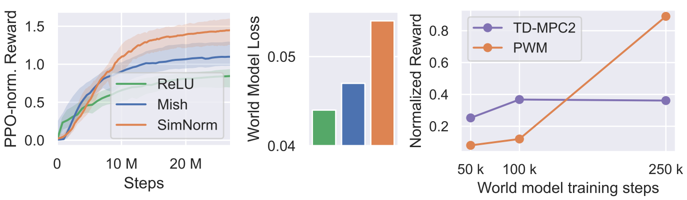
We increase the **contact stiffness** to be more realistic but also more stiff contact gives gradients with high sample error Suh et al. (2022). We run the same experiment as Section 4.1, but only for the Hopper task and present the aggregate results from 5 random seeds in Figure 7a where we normalize rewards by the maximum reward achieved by PPO in Section 4.1. 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 generalizable the differentiable simulations.



(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.



(a) World model ablation.

(b) World model vs policy sample efficiency.

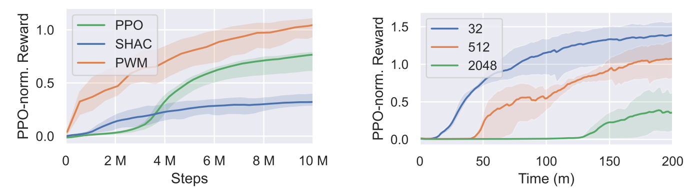
Figure 8: The **left figure ablates the activation functions** of the world model used to learn policies on the Ant task. We progressively add more regularization to the world model via changes to the activation function and observe an inverse correlation between world model loss and policy reward. This 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.

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% 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 4.1 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. Additional results in Appendix F.

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

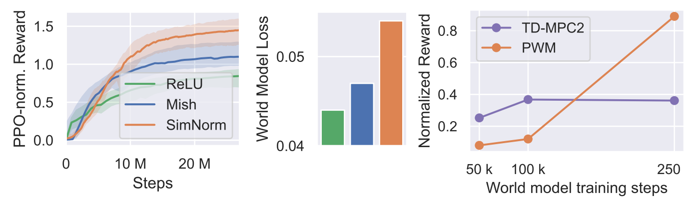
Under review as a conference paper at ICLR 2025



(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.



(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.

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% 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 4.1 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. Additional results in Appendix F.

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

8

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 TD-MPC2 but also require better trained world models in order to obtain high reward.

5 Related work

Reinforcement learning (RL) strategies are divided into model-based and model-free approaches, with the latter not assuming a model of the environment (Arulkumaran et al., 2017). Model-free approaches, such as PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018) do not require a model of the environment and represent on-policy and off-policy methods respectively. These algorithms use an actor-critic structure, where the critic assesses the policy while the actor updates it through gradient-based optimization to maximize rewards (Konda & Tsitsiklis, 1999).

Gradient estimator types. In the absence of direct access to dynamics and reward functions, it is common to use the Policy Gradients Theorem (Sutton et al., 1999), a zeroth-order method, to estimate gradients. Although robust to discontinuities, this method exhibits high variance, leading to sample inefficiency (Mohamed et al., 2020). In contrast, first-order gradients (FoG) offer lower variance by differentiating through the objective but struggle with discontinuities (Suh et al., 2022). Differentiable simulations have risen as a tool to study the properties of gradient estimators (Howell et al., 2022; Metz et al., 2021) and have produced model-based algorithms which use FoG optimization through physics to learn high-performing policies (Xu et al., 2022; Georgiev et al., 2024).

Multi-task models. While traditional RL focuses on single-task policies, the broader robotics field is increasingly adopting large multi-task models through behavior cloning (Firoozi et al., 2023). Recent efforts like Open X (Padalkar et al., 2023) and Octo (Octo Model Team et al., 2024) have demonstrated improved performance across various tasks and embodiments by leveraging large models and datasets. However, the potential of these large-scale approaches in RL remains largely unexplored. While GATO (Reed et al., 2022) attempted to scale model-free RL across multiple tasks, it faced challenges with sample inefficiency and required significant fine-tuning. Conversely, TD-MPC2 (Hansen et al., 2024) successfully scaled a 317M parameter world model for online planning across 80 tasks. While showing impressive multi-task scalability, it failed to solve all tasks and exhibits limited scalability due to online planning. Our work builds on the world model architecture proposed by TD-MPC2 but employs FoG optimization for policy learning and extracts per-tasks policies. DreamerV3 (Hafner et al., 2023) also integrates world models with FoG but focuses on online learning without addressing multi-task scenarios. Our work delves deeper into the relationship between world models and policy learning, exploring the essential characteristics of world models that facilitate efficient optimization.

6 Conclusion

In this study, we analyzed world models through policy gradient estimation and identified an inverse correlation between the accuracy of world models and episode rewards. We concluded that world models should prioritize smoothness and a smaller optimality gap over accuracy to enhance policy performance. Building on these insights, we propose Policy learning through Multi-task World Models (PWM), a MBRL algorithm that integrates smooth world models with first-order gradient (FoG) optimization. Our evaluations showed that PWM can outperform existing methods, including those with access to ground-truth simulation dynamics, in learning high-reward policies for high-dimensional tasks. To scale to a multi-task settings, we propose a framework where world models are pre-trained offline and treated as differentiable simulations. Our results demonstrate that PWM can be used to learn expert policies in <10 minutes per task, achieving higher rewards without the need for expensive online planning. With ample data and large, smooth world models, we believe this approach has significant potential for scalability.

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 TD-MPC2 world models used are difficult to train at scale due to their auto-regressive formulation.

Under review as a conference paper at ICLR 2025

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 TD-MPC2 but also require better trained world models in order to obtain high reward.

5 Related work

Reinforcement learning (RL) strategies are divided into model-based and model-free approaches, with the latter not assuming a model of the environment Arulkumaran et al. (2017). Model-free approaches, such as PPO Schulman et al. (2017) and SAC Haarnoja et al. (2018) do not require a model of the environment and represent on-policy and off-policy methods respectively. These algorithms use an actor-critic structure, where the critic assesses the policy while the actor updates it through gradient-based optimization to maximize rewards Konda & Tsitsiklis (1999).

Gradient estimator types. In the absence of direct access to dynamics and reward functions, it is common to use the Policy Gradients Theorem Sutton et al. (1999), a zeroth-order method, to estimate gradients. Although robust to discontinuities, this method exhibits high variance, leading to sample inefficiency Mohamed et al. (2020). In contrast, first-order gradients (FoG) offer lower variance by differentiating through the objective but struggle with discontinuities Suh et al. (2022). Differentiable simulations have risen as a tool to study the properties of gradient estimators Howell et al. (2022); Metz et al. (2021) and have produced model-based algorithms which use FoG optimization through physics to learn high-performing policies Xu et al. (2022); Georgiev et al. (2024).

Multi-task models. While traditional RL focuses on single-task policies, the broader robotics field is increasingly adopting large multi-task models through behavior cloning Firoozi et al. (2023). Recent efforts like Open X Padalkar et al. (2023) and Octo Octo Model Team et al. (2024) have demonstrated improved performance across various tasks and embodiments by leveraging large models and datasets. However, the potential of these large-scale approaches in RL remains largely unexplored. While GATO Reed et al. (2022) attempted to scale model-free RL across multiple tasks, it faced challenges with sample inefficiency and required significant fine-tuning. Conversely, TD-MPC2 Hansen et al. (2024) successfully scaled a 317M parameter world model for online planning across 80 tasks. While showing impressive multi-task scalability, it failed to solve all tasks and exhibits limited scalability due to online planning. Our work builds on the world model architecture proposed by TD-MPC2 but employs FoG optimization for policy learning and extracts per-tasks policies. DreamerV3 Hafner et al. (2023) also integrates world models with FoG but focuses on online learning without addressing multi-task scenarios. Our work delves deeper into the relationship between world models and policy learning, exploring the essential characteristics of world models that facilitate efficient optimization.

6 Conclusion

In this study, we analyzed world models through policy gradient estimation and identified an inverse correlation between the accuracy of world models and episode rewards. We concluded that world models should prioritize smoothness and a smaller optimality gap over accuracy to enhance policy performance. Building on these insights, we propose Policy learning through Multi-task World Models (PWM), a MBRL algorithm that integrates smooth world models with first-order gradient (FoG) optimization. Our evaluations showed that PWM can outperform existing methods, including those with access to ground-truth simulation dynamics, in learning high-reward policies for high-dimensional tasks. To scale to a multi-task settings, we propose a framework where world models are pre-trained offline and treated as differentiable simulations. Our results demonstrate that PWM can be used to learn expert policies in <10 minutes per task, achieving higher rewards without the need for expensive online planning. With ample data and large, smooth world models, we believe this approach has significant potential for scalability.

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 TD-MPC2 world models used are difficult to train at scale due to their auto-regressive formulation.

9

Reproducibility statement. Code, training data and checkpoints are made available at `policy-world-model.github.io`. We rely on dflex, MetaWorld, DMControl and MuJoCo for simulation which are publicly available under MIT and Apache 2.0 licenses. We use multi-task data from TD-MPC2 which is publicly available. Implementation details and full list of hyper-parameters are available in Appendix C.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *arXiv preprint arXiv:2312.07843*, 2023.

Ignat Georgiev, Krishnan Srinivasan, Jie Xu, Eric Heiden, and Animesh Garg. Adaptive horizon actor-critic for policy learningin contact-rich differentiable simulation. In *International Conference on Machine Learning*. PMLR, 2024.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pp. 694–710. PMLR, 2023.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. 2024.

Taylor A Howell, Simon Le CleacâĂŹh, J Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 9, 2022.

Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 38–44. IEEE, 2016.

Under review as a conference paper at ICLR 2025

Reproducibility statement. Code, training data and checkpoints are made available at `policy-world-model.github.io`. We rely on dflex, MetaWorld, DMControl and MuJoCo for simulation which are publicly available under MIT and Apache 2.0 licenses. We use multi-task data from TD-MPC2 which is publicly available. Implementation details and full list of hyper-parameters are available in Appendix C.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value expansion for efficient model-free reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 2018.

Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *arXiv preprint arXiv:2312.07843*, 2023.

Ignat Georgiev, Krishnan Srinivasan, Jie Xu, Eric Heiden, and Animesh Garg. Adaptive horizon actor-critic for policy learningin contact-rich differentiable simulation. In *International Conference on Machine Learning*. PMLR, 2024.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pp. 694–710. PMLR, 2023.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. 2024.

Taylor A Howell, Simon Le CleacâĂŹh, J Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 9, 2022.

10

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should i run offline reinforcement learning or behavioral cloning? In *International conference on learning representations*, 2021.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.

Minji Lee, Jeongmin Lee, and Dongjun Lee. Differentiable dynamics simulation using invariant contact mapping and damped contact force. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11683–11689. IEEE, 2023.

Luke Metz, C Daniel Freeman, Samuel S Schoenholz, and Tal Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

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

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

Richard M Murray and John Edmond Hauser. *A case study in approximate linearization: The acrobat example*. Electronics Research Laboratory, College of Engineering, University of âĂę, 1991.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

Paavo Parmas, Takuma Seno, and Yuma Aoki. Model-based reinforcement learning with scalable composite policy gradient estimators. In *International Conference on Machine Learning*, pp. 27346–27377. PMLR, 2023.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015.

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

Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pp. 20668–20696. PMLR, 2022.

Under review as a conference paper at ICLR 2025

Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 38–44. IEEE, 2016.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should i run offline reinforcement learning or behavioral cloning? In *International conference on learning representations*, 2021.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.

Minji Lee, Jeongmin Lee, and Dongjun Lee. Differentiable dynamics simulation using invariant contact mapping and damped contact force. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11683–11689. IEEE, 2023.

Luke Metz, C Daniel Freeman, Samuel S Schoenholz, and Tal Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

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

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

Richard M Murray and John Edmond Hauser. *A case study in approximate linearization: The acrobat example*. Electronics Research Laboratory, College of Engineering, University of âĂę, 1991.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

Paavo Parmas, Takuma Seno, and Yuma Aoki. Model-based reinforcement learning with scalable composite policy gradient estimators. In *International Conference on Machine Learning*, pp. 27346–27377. PMLR, 2023.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015.

11

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2022.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

Under review as a conference paper at ICLR 2025

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

Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pp. 20668–20696. PMLR, 2022.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2022.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

12

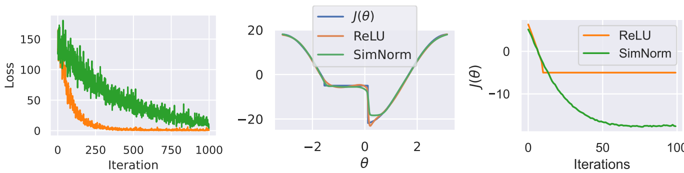


Figure 10: Extended ball-wall toy example results. The **left** figure shows the model losses as they are trained to approximate the target function $J(\theta$. The middle figure shows the output of the trained model across the spectrum of θ as well as the true target. This is the function we attempt 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 3.1. In constructing this toy example we chose a simple physical system that exhibits contact discontinuities. Inspired by Suh et al. (Suh et al., 2022), 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.

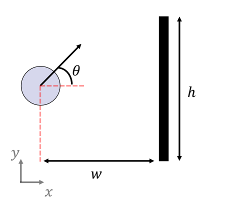


Figure 9: Pedagogical ball-wall toy problem visualized.

Under review as a conference paper at ICLR 2025

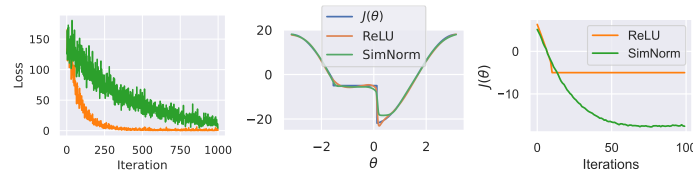


Figure 10: Extended ball-wall toy example results. The **left** figure shows the model losses as they are trained to approximate the target function $J(\theta)$. The middle figure shows the output of the trained model across the spectrum of θ as well as the true target. This is the function we attempt 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 3.1. In constructing this toy example we chose a simple physical system that exhibits contact discontinuities. Inspired by Suh et al. (Suh et al., 2022), 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:

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

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 10 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.

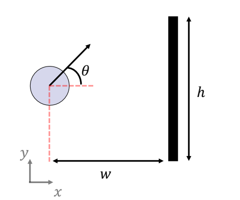


Figure 9: Pedagogical ball-wall toy problem visualized.

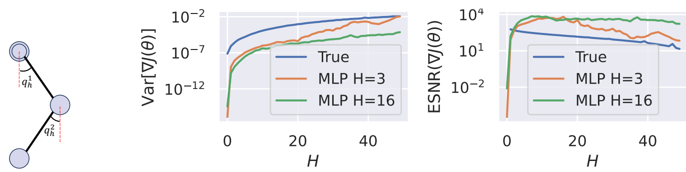


Figure 11: Double pendulum pedagogical example. The middle figure evaluates the variance of policy gradient estimates over $N = 100$ Monte-Carlo samples for varying horizons H. The right figure shows the same data but plots the Expected Signal-to-Noise ratio (ESNR) with higher values translating to more 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 (Murray & Hauser, 1991)) 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 3.2 as it is the easiest problem exhibiting chaosness. We model this toy problem similar to DMControl (Tunyasuvunakool et al., 2020) 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 (Xu et al., 2022) 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 (Hansen et al., 2024) 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 (Parmas et al., 2023) 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.

Under review as a conference paper at ICLR 2025

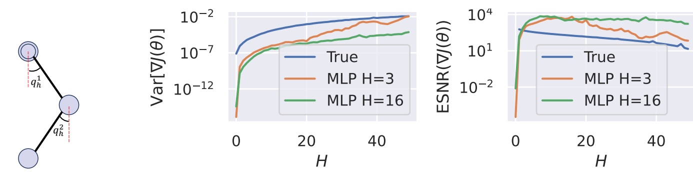


Figure 11: Double pendulum pedagogical example. The middle figure evaluates the variance of policy gradient estimates over $N = 100$ Monte-Carlo samples for varying horizons H. The right figure shows the same data but plots the Expected Signal-to-Noise ratio (ESNR) with higher values translating to more 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 (Murray & Hauser, 1991)) 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 3.2 as it is the easiest problem exhibiting chaosness. We model this toy problem similar to DMControl (Tunyasuvunakool et al., 2020) 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 (Xu et al., 2022) 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 (Hansen et al., 2024) 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 (Parmas et al., 2023) 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.

14

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 3.3 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 (Hansen et al., 2024; Lee et al., 2023), 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:

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

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 $\mathrm{SymExp}(x)$ but note that the $\mathrm{sign}(x)$ operator is not differentiable and would therefore not work for FoG. Instead, we chose to omit the $\mathrm{SymExp}(x)$ 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 4, 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 (Schulman et al., 2017), 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 4.1 using the default hyper-parameters and a replay buffer of size 1024.

Under review as a conference paper at ICLR 2025

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 3.3 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 (Hansen et al., 2024; Lee et al., 2023), 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:

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

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 $\mathrm{SymExp}(x)$ but note that the $\mathrm{sign}(x)$ operator is not differentiable and would therefore not work for FoG. Instead, we chose to omit the $\mathrm{SymExp}(x)$ 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 4, 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 (Schulman et al., 2017), 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 4.1 using the default hyper-parameters and a replay buffer of size 1024.

15

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.

D CONTACT-RICH SINGLE TASK EXPERIMENT DETAILS

In Section 4.1, 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 $m = 8$.
3. **Anymal**, a more sophisticated quadruped with $n = 49$ and $m = 12$ modeled after (Hutter et al., 2016).
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:

Under review as a conference paper at ICLR 2025

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.

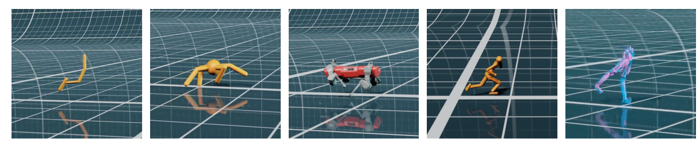


Figure 12: Locomotion environments (left to right): Hopper, Ant, Anymal, Humanoid and SNU Humanoid.

D CONTACT-RICH SINGLE TASK EXPERIMENT DETAILS

In Section 4.1, 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 $m = 8$.
3. **Anymal**, a more sophisticated quadruped with $n = 49$ and $m = 12$ modeled after (Hutter et al., 2016).
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:

16

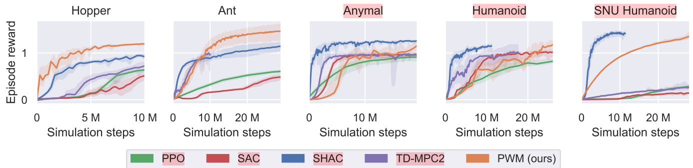


Figure 13: Learning curves for each environment. This figure shows 50% IQM and 95% CI across 5 random seeds for each task in the dflex simulation suite. Rewards are normalized by the maximum reward achieved by PPO (usually $\geq$ 100M steps). While PWM remains competitive with SHAC for most tasks, it 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 4

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.

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

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

$R_{angle} = \|\boldsymbol{q}_{forward} - \boldsymbol{q}_{agent}\|_2^2$ 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:

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.

Under review as a conference paper at ICLR 2025

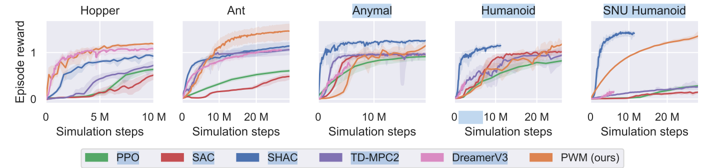


Figure 13: Learning curves for each environment. This figure shows 50% IQM and 95% CI across 5 random seeds for each task in the dflex simulation suite. Rewards are normalized by the maximum reward achieved by PPO (usually $\geq$ 100M steps). While PWM remains competitive with SHAC for most tasks, it does not scale well to the 152 action dimension SNU Humanoid. Note that SHAC uses gradients from the differentiable simulation directly and converges in fewer samples, explaining the truncated curves. DreamerV3 results are incomplete.

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 4

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.

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

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

$R_{angle} = \|\boldsymbol{q}_{forward} - \boldsymbol{q}_{agent}\|_2^2$ 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:

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.

17

	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	1.36 ± 0.56

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.

	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	5767 ± 2394

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.

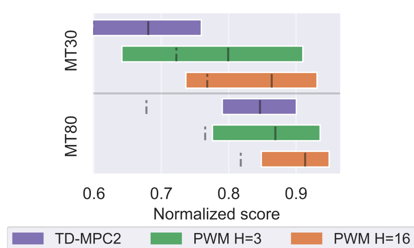


Figure 14: Horizon ablation of the world model

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.

Under review as a conference paper at ICLR 2025

	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
DreamerV3	1.12 ± 0.56	...	...	...	...
TDMPC2	0.85 ± 0.37	1.07 ± 0.44	0.98 ± 0.48	1.05 ± 0.46	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	1.36 ± 0.56

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.

	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
DreamerV3	5321 ± 2664	...	...	...	...
TDMPC2	4027 ± 1768	7080 ± 2885	11787 ± 4702	7634 ± 3317	1121 ± 525
SHAC	4837 ± 142	7662 ± 859	15157 ± 481	8387 ± 292	5924 ± 329
PWM	5680 ± 2303	9672 ± 2012	13927 ± 2882	8661 ± 1792	5767 ± 2394

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.

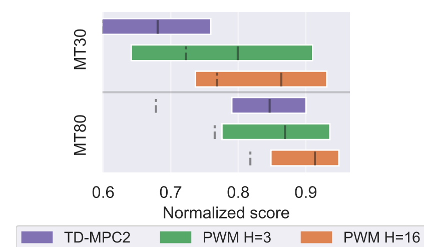


Figure 14: Horizon ablation of the world model

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.

18

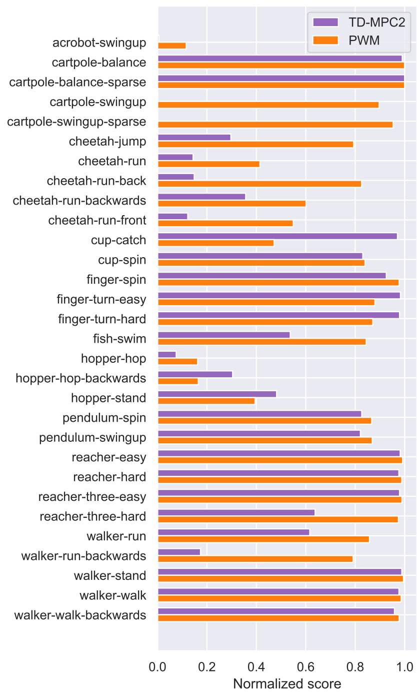


Figure 15: Individual task results for MT30 task set.

Under review as a conference paper at ICLR 2025

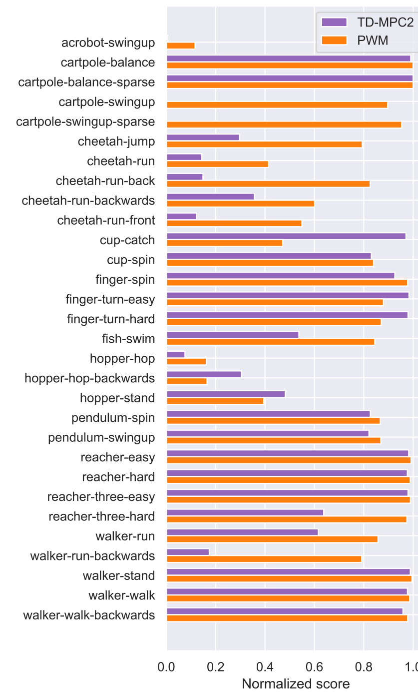


Figure 15: Individual task results for MT30 task set.

19

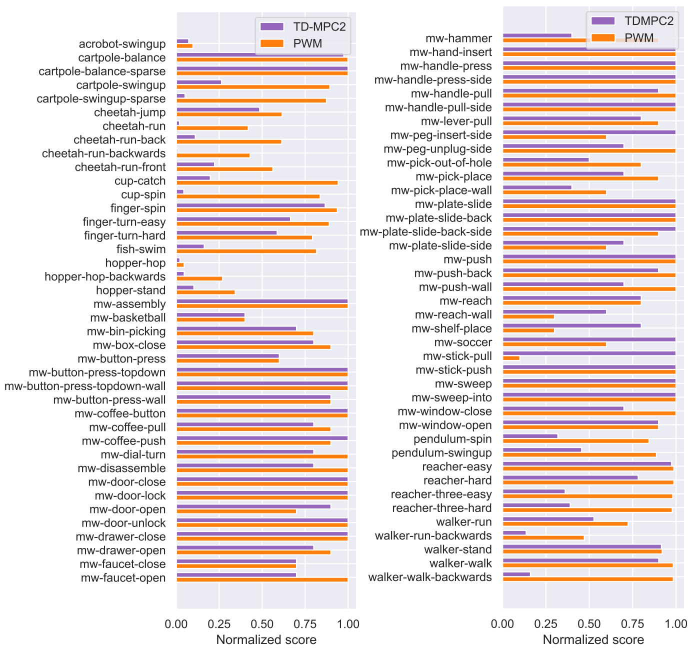


Figure 16: Individual task results for MT80 task set.

F ADDITIONAL ABLATION RESULTS

F.1 WORLD MODEL REGULARIZATION ABLATION

We extend our world model regularization ablation from Section 4.3 to 3 additional dflex locomotion tasks - Hopper, Anymal and Humanoid. The complete results are presented in Figure 17 and follow the same format as the original results in Figure 8a - 50% IQM with 95% CI across 3 seeds. From the results we can observe that progressively adding more regularization, results in higher policy rewards. This follows the ablation results in our paper and the one of the core contribution of work - more accurate world models, do not results in better rewards. Instead of building world models for accuracy, we should build them to enable better policy optimization and thus higher rewards.

F.2 TRAINING A MULTI-TASK POLICY

One of the contributions of our work is the multi-task learning framework of PWM. Instead of training a single multi-task policy, we propose training a single multi-task world model with a supervised objective and then extracting RL policies from it very efficiently. In this section we justify this empirically by reproducing the MT30 experiment of Section 4.2 with two additional baselines - TD-MPC2 without planning and PWM with a single multi-task policy. This strips down both algorithms to be very similar, the major difference being that TD-MPC2 features an off-policy SAC-like policy while PWM features an on-policy SHAC-like policy. Both are fed the same one-hot

Under review as a conference paper at ICLR 2025

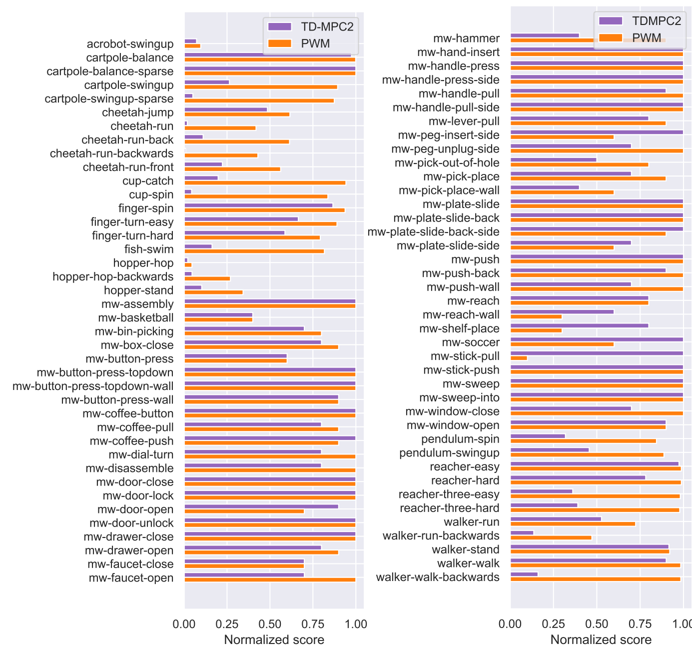


Figure 16: Individual task results for MT80 task set.

F ADDITIONAL ABLATION RESULTS

F.1 WORLD MODEL REGULARIZATION ABLATION

We extend our world model regularization ablation from Section 4.3 to 3 additional dflex locomotion tasks - Hopper, Anymal and Humanoid. The complete results are presented in Figure 17 and follow the same format as the original results in Figure 8a - 50% IQM with 95% CI across 3 seeds. From the results we can observe that progressively adding more regularization, results in higher policy rewards. This follows the ablation results in our paper and the one of the core contribution of work - more accurate world models, do not results in better rewards. Instead of building world models for accuracy, we should build them to enable better policy optimization and thus higher rewards.

F.2 TRAINING A MULTI-TASK POLICY

One of the contributions of our work is the multi-task learning framework of PWM. Instead of training a single multi-task policy, we propose training a single multi-task world model with a supervised objective and then extracting RL policies from it very efficiently. In this section we justify this empirically by reproducing the MT30 experiment of Section 4.2 with two additional baselines - TD-MPC2 without planning and PWM with a single multi-task policy. This strips down both algorithms to be very similar, the major difference being that TD-MPC2 features an off-policy SAC-like policy while PWM features an on-policy SHAC-like policy. Both are fed the same one-hot

20

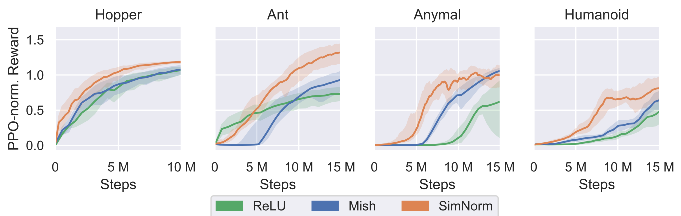


Figure 17: World model regularization ablation. This figure extends the ablation results of Figure 8a with additional tasks. The figure shows 50 % IQM with 95% CI over 5 seeds per task.

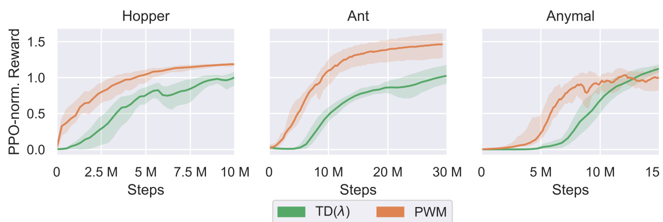


Figure 19: TD(λ) ablation. This figure compares vanilla PWM with a version that uses a TD(λ) actor objective. Results shown are 50% IQM with 95% CI over 3 seeds.

task embeddings. The results in 18 show that both fail at solving the MT30 benchmark. In the case of TD-MPC2, this reveals its reliance on planning and in the case of PWM, this reveals the need to train policies per-task. We believe this is due to the unstable optimization objective of the RL problem.

F.3 TD(λ) ABLATION

Here we replace the actor objective of PWM with TD(λ) similar to Dreamer (Hafner et al., 2019). We evaluate this on 3 single-task experiments - Hopper, Ant and Anymal and show the results in 19. 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.

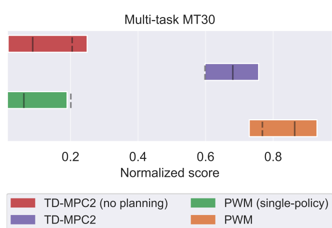


Figure 18: Framework ablation ablation. Extends the results of Figure 6 with two additional baselines: TD-MPC2 without planning and PWM with a single multi-task policy learned over all embeddings. The figure shows 50% IQM results with solid lines, mean with dashed lines and 95 % CI.

Under review as a conference paper at ICLR 2025

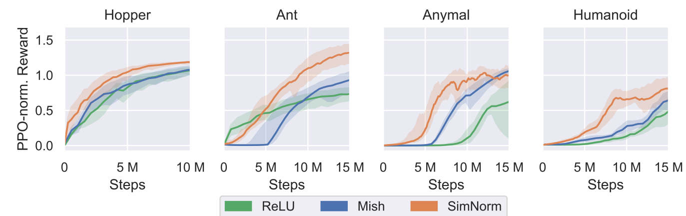


Figure 17: World model regularization ablation. This figure extends the ablation results of Figure 8a with additional tasks. The figure shows 50 % IQM with 95% CI over 5 seeds per task.

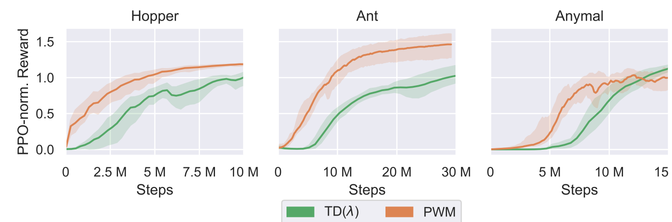


Figure 19: TD(λ) ablation. This figure compares vanilla PWM with a version that uses a TD(λ) actor objective. Results shown are 50% IQM with 95% CI over 3 seeds.

task embeddings. The results in 18 show that both fail at solving the MT30 benchmark. In the case of TD-MPC2, this reveals its reliance on planning and in the case of PWM, this reveals the need to train policies per-task. We believe this is due to the unstable optimization objective of the RL problem.

F.3 TD(λ) ABLATION

Here we replace the actor objective of PWM with TD(λ) similar to Dreamer (Hafner et al., 2019). We evaluate this on 3 single-task experiments - Hopper, Ant and Anymal and show the results in 19. 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.

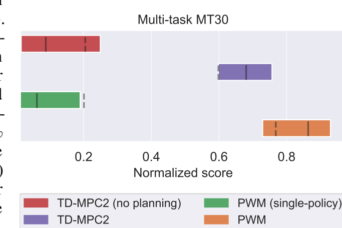


Figure 18: Framework ablation ablation. Extends the results of Figure 6 with two additional baselines: TD-MPC2 without planning and PWM with a single multi-task policy learned over all embeddings. The figure shows 50% IQM results with solid lines, mean with dashed lines and 95 % CI.

21