

Learning to Walk from Three Minutes of Data with Semi-structured Dynamics Models

Anonymous Author(s)

Affiliation

Address

email

Abstract: Traditionally, model-based reinforcement learning (MBRL) methods exploit neural networks as flexible function approximators to represent *a priori* unknown environment dynamics. However, training data is typically scarce in practice, and these black-box models often fail to generalize beyond the training data. Modelling architectures that leverage known physics can substantially reduce the complexity of system-identification, but break down in the face of complex real-world phenomena such as contact. This paper introduces a novel framework for learning predictive models for contact-rich system which seamlessly integrates structured first-principles modeling techniques with black-box autoregressive models. Specifically, we develop an ensemble of probabilistic models to estimate external forces, conditioned on historical observations and actions, and integrate these predictions using known Lagrangian dynamics. This semi-structured approach enables us to make accurate predictions far into the future with substantially fewer training samples than prior methods. We leverage this capability to push the sample-complexity boundary for real-world model-based reinforcement learning. We validate our approach through real-world experiments with a Unitree Go1 quadruped robot, learning dynamics gaits – from scratch – on both hard and soft surfaces with just minutes of data.

Keywords: Model-Based Reinforcement Learning, Physics-Based Models

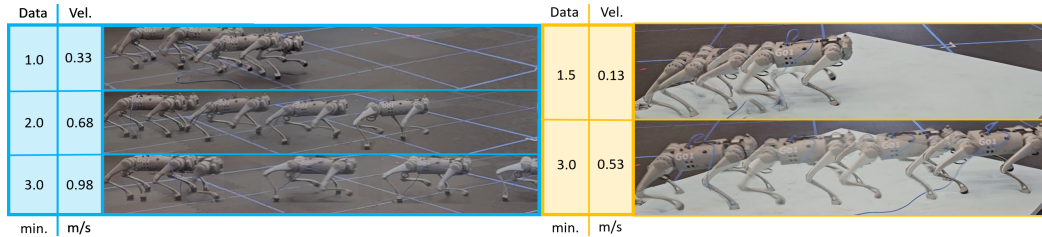


Figure 1: Training a Unitree Go1 quadruped to walk from scratch using our method on hard ground (left) and memory foam (right).

1 Introduction

Effective robotic agents must navigate complex interactions between the robot and its environment, which are difficult to model using first principles. Model-based reinforcement learning is a powerful paradigm for controller synthesis [1], wherein the robot learns a generative dynamics model through repeated interaction with the environment. The model can then be used to hallucinate synthetic rollouts [2], providing a source of data augmentation for policy optimization algorithms [3, 4, 5]. When the model is accurate, it can generate long rollouts which extrapolate beyond the training data and substantially reduce the number of real samples needed to learn an effective control strategy. However, in practice, the black-box neural network models favored in the model-based reinforcement

learning (MBRL) literature struggle to generalize beyond the training data [6, 7, 8], and thus do not outperform modern model-free alternatives [9, 10]. Currently, both paradigms are too inefficient and unreliable to make learning new behaviors in the real world practical for many applications.

An appealing alternative is to leverage knowledge of physics to design model classes which can generalize beyond small real-world training sets. This general approach has been used for efficient controller synthesis across many bodies of work, ranging from classic adaptive control techniques [11, 12, 13, 14] to more recent physics-informed neural architectures [15, 16, 17]. However, while models built upon physical first principles can reliably capture the rough structure of the dynamics, they break down in the face of complex phenomena which are difficult to model such as contact [18]. Moreover, even modelling these interactions typically requires access to privileged information about the environment. State-of-the-art MBRL methods [6, 7, 8] ‘branch’ hallucinated trajectories off states encountered by the robot in the real world, but high-fidelity contact solvers [19, 20] require access to features such as the geometry of the ground under the robot and precise locations where contact is made. Reliably estimating these quantities at runtime is an open and active area of research [21, 22, 23, 24, 25]. Due to these challenges, however, real-world learning of control policies using first-principles models has largely excluded contact-rich systems.

These issues beg the question: can we develop a model-based policy optimization framework that is implementable in the real world and which leverages the structure of physics-based models to accelerate learning? To make this question concrete, this paper focuses on the problem of learning an effective locomotion strategy for a quadrupedal robot **entirely from scratch in the real world**, as depicted in Fig. 1. We consider the case where the robot’s observations include proprioceptive measurements via joint encoders, IMU measurements, and a global velocity estimator.

We can begin unpacking this question by studying the Lagrangian dynamics for the robot:

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = B\tau + J^T F^e, \quad (1)$$

where q and \dot{q} are the generalized coordinates and velocities of the robot, motor torques τ are distributed to the joints by the matrix B , F^e represents the external forces acting on the robot, J is the Jacobian, and M , C , and G are the mass matrix, Coriolis terms, and gravity vector. Here, B , M , C , and G are determined by the geometry and inertial properties of the robot, which are known *a priori*.

Thus, $J^T F^e$ is the only unknown term in the dynamics, but we can fully exploit the remaining structure of the Lagrangian differential when performing system identification. We estimate $J^T F^e$ using an ensemble of probabilistic models [26] which are conditioned on a history of available observations and actions. The history enables the models to infer latent representations of the environment, similar to recent works on simulation-based training [27], that are useful for inferring the external torques that will act on the robot. We fit these models by *a*) propagating the external torque estimate through the Lagrangian dynamics (1) to construct informative multi-step prediction losses, and *b*) using techniques from filtering theory [12] to attenuate the effects of noisy state estimates. By using these techniques, we avoid the need to add random ‘jittering’ exploration noise to the robots actions when learning the dynamics model, which can potentially damage hardware but is required but is required for many less structured approaches. Altogether, our approach (Fig. 2) unifies structured system identification techniques with black-box modeling and learns semi-structured (auto-regressive, history-conditioned) models which demonstrate substantial generalization capabilities beyond typical approaches.

Finally, we put these piece together to push model-based policy optimization algorithms into new regimes of sample efficiency. The sample efficiency of modern policy optimization algorithms are limited by the update-to-data ratio (UTD), which is ratio of policy updates to real world data points. In particular, when too many updates are performed algorithms will over-fit to the data set and become unstable. Because black-box MBRL algorithms can produce synthetic data, they can support relatively high UTD’s. Our approach, however, is able to produce rollouts that generalize far beyond the available training data and leverage a substantially richer synthetic data set. As a result, our approach supports UTDs that are substantially higher than prior model-based method, enabling aggressive policy optimization in low-data regimes.

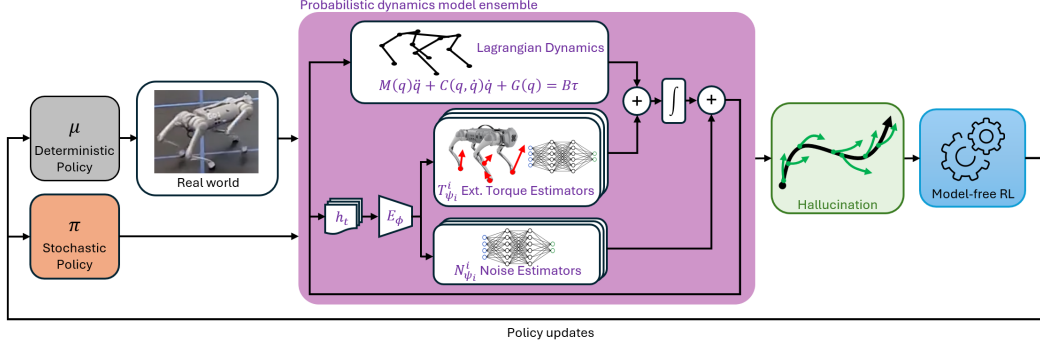


Figure 2: Our approach. A deterministic policy is used to collect data from the real world while a stochastic policy is utilized in conjunction with the learned dynamics model to “hallucinate” short synthetic rollouts which branch from this data. The model incorporates Lagrangian dynamics and encodes previous state predictions, which are fed to external torque and noise estimators to predict future states. The synthetic data is used with a model-free RL algorithm to update the policies.

2 Preliminaries and Problem Formulation

Our goal is to learn control policies from scratch in the real-world which enable the Unitree Go1 quadruped to locomote as rapidly as possible.

Control Architecture: In this work, we use a policy which only has access to histories of proprioceptive measurements from joint encoders and IMU measurements, but not access to global coordinates such as the height of the robot above the ground. We adopt the control architecture depicted in Fig. 3, which is similar to many prior works on RL-based locomotion [27, 28, 29]. We optimize a feed-forward neural network policy which takes in histories of available observations and outputs (i) changes to parameters of a nominal feed-forward repetitive stepping motion that generates desired foot positions and (ii) offsets to these nominal foot positions. The resulting foot positions are sent to an inverse kinematics solver which computes desired joint positions. These desired joint positions are output at 100 Hz to low-level joint PD controllers for conversion to torques.

Notation: While the underlying dynamics of the robot evolve in continuous time according to the differential equation (1), the policy acts in discrete time and we will use subscripts to denote discrete time steps. The state of the robot $s_t \in \mathcal{S} \subseteq \mathbb{R}^{k_1}$ captures the available proprioceptive states within q_t and their velocities \dot{q}_t . We capture the state of the environment (or *extrinsics*) with the variable $e_t \in \mathcal{E}$, which includes non-proprioceptive information (such as the height of the robot above the ground) and the state of the ground underfoot (such as terrain deformation and temperature effects). The actions for the robot $a_t \in \mathcal{A} \subset \mathbb{R}^{k_2}$ are simply the aforementioned outputs of the policy. The overall dynamics for the robot and the environment are defined by:

$$\text{Robot Transitions: } s_{t+1} \sim p_s(\cdot | s_t, a_t, e_t) \quad \text{Environment Transitions: } e_{t+1} \sim p_e(\cdot | s_t, a_t, e_t). \quad (2)$$

To control this joint system, we will denote the policy the agent optimizes via $a_t \sim \pi(\cdot | s_t, h_t)$, where $h_t = (s_{t-1}, \dots, s_{t-h})$ bundles the histories of proprioceptive state measurements.

Reinforcement Learning Problem: We formally frame the learning of a locomotion controller in the real world in terms of a partially observable Markov decision process (POMDP) [30], defined by the tuple $(\mathcal{X}, \mathcal{A}, p, r, \Omega, O, \gamma)$. Here, $\mathcal{X} = \mathcal{S} \times \mathcal{E}$ is the overall state space for the system, and $p(\cdot | s_t, a_t, e_t) = (p_s(\cdot | s_t, a_t, e_t), p_e(\cdot | s_t, a_t, e_t))$ captures the joint robot-environment dynamics. The space of observations Ω consists of the proprioceptive states that can be measured, and the observation distribution $\hat{s}_t \sim O(\cdot | s_t, a_t, e_t)$ provides estimates of the proprioceptive states from onboard sensors. The reward function $r(s_t, a_t, s_{t+1})$ depends only upon the robot states and actions, and is therefore directly measurable in the real world. We define the reward function to maximize the robot’s forward velocity, maintain upright orientation, minimize angular rates, conserve energy, and avoid excessive torques. Finally, we define a termination flag $d_t \in \{0, 1\}$ where $d_t = 1$ when body roll or pitch exceed limits. Exact definitions of the observation space, reward function, and

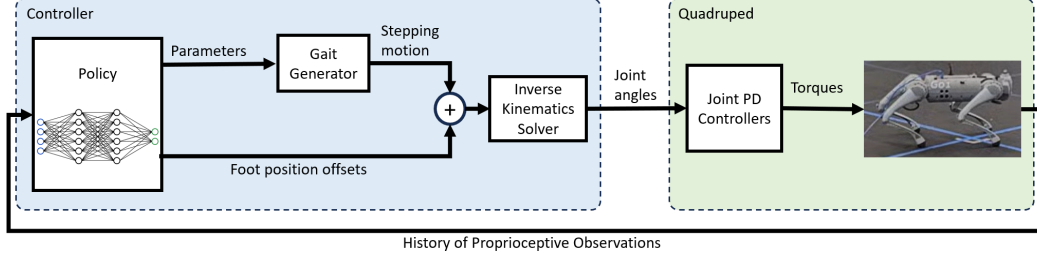


Figure 3: Control architecture. The policy takes in a history of observations and outputs parameters to an open-loop gait generator and offsets to the gait. The resulting foot positions are sent to an inverse kinematics solver which computes desired joint angles for joint level PD controllers.

112 termination condition are found in the supplementary material. Given an episode length $T \in \mathbb{N}$,
 113 discount factor $\gamma \in (0, 1)$, and a distribution x_0 over \mathcal{X} of initial conditions for the system, the goal
 114 is to maximize the expected discounted total reward: $\max_{\pi} \mathbb{E} \sum_{t=0}^T \gamma^t (1 - d_t) \cdot r(s_t, a_t, s_{t+1})$.

115 3 Reinforcement Learning with Semi-structured Dynamics Models

116 We now present our framework for leveraging semi-structured dynamics models for real-world
 117 MBRL. First, we detail our novel method for training proprioceptive contact models, which are
 118 integrated with known robot dynamics to generate accurate multi-step predictions. Next, we outline
 119 the key features of our algorithm that enable efficient real-world training.

120 3.1 Training Proprioceptive Contact Models for Dynamics Prediction

121 We construct our approximations to the discrete-time probabilistic robot transition dynamics $s_{t+1} \sim$
 122 $p_s(\cdot | s_t, a_t, e_t, \cdot)$ by building on top of the deterministic Lagrangian differential equation (1). At a
 123 high level, our approach uses an ensemble of neural networks which are conditioned on the history
 124 of observations—including a compressed internal representation for the state of the environment—to
 125 estimate the external torques exerted on the robot by its environment. A numerical integrator is
 126 then used to propagate these predictions through the Lagrangian dynamics to produce deterministic
 127 predictions for the next state. Finally, we fit Gaussian noise models to these deterministic predic-
 128 tions to account for uncertainty in the learned estimators, noise from onboard state estimators, and
 129 inherent stochasticity in the dynamics which are not captured in (1).

130 In detail, we can write the deterministic Lagrangian dynamics as:

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = B\tau + \underbrace{J(s, e)^T F^e(s, e, \tau)}_{\tau^e(s, e, \tau)}, \quad (3)$$

131 where we now explicitly denote the dependence of the contact Jacobian $J(s, e)$ on the configura-
 132 tion of the robot and the environment variables and the dependence of the external contact forces
 133 $F^e(s, e, \tau)$ on these terms as well as the low-level torque output by the robot. As discussed in [Sec-](#)
 134 [tion 1](#), identifying precise locations where contact occurs on the robot can be extremely difficult from
 135 on-board measurements. Thus, instead of inferring J and F^e separately, we directly estimate the
 136 external torques $\tau^e = J^T F^e$ imposed in the joint coordinates by the interactions between the robot
 137 and its environment. Altogether, we leverage the Lagrangian dynamics to produce a deterministic
 138 prediction for the next robot state using the following successive computation:

Latent Encoding of History:	$z_t = E_{\phi}(h_t)$	
Estimated Low-Level Motor Torques:	$\tau_t = G(s_t, a_t)$	
Deterministic External Torque Estimators:	$\bar{\tau}_t^{e,i} = T_{\psi_i}^i(s_t, a_t, z_t)$	(4)
Integrated Deterministic State Predictions:	$\bar{s}_{t+1}^i = I(s_t, \tau_t, \bar{\tau}_t^{e,i})$	
Overall Deterministic State Prediction:	$\bar{s}_{t+1} = S^i(s_t, a_t, h_t),$	

139 where z_t is a compressed latent representation of e_t produced by an encoder E_{ϕ} with parameters
 140 ϕ , τ_t is a zero-order hold estimate for the low-level motor torques applied to the robot over the

sampling interval, and G is known map that captures how the state of the robot and action by the policy are processed by the gait generator and low-level PD controller into motor commands. $\bar{\tau}_t^{e,i}$ is i -th predicted external torque predicted by network $T_{\psi_i}^i$ with parameters ψ_i , I captures how the numerical integrator propagates the current state estimate and zero-order holds τ_t and $\hat{\tau}_t$ through the Lagrangian dynamics over the sampling interval to produce the i -th deterministic prediction for the next state, and S^i neatly encapsulates the overall mapping from s_t, a_t , and h_t to \bar{s}_{t+1}^i . We then apply learned additive Gaussian noise to produce uncertainty-aware predictions for the next state:

$$\begin{aligned} \text{Additive Noise Estimates:} \quad & \Sigma_t^i = N_{\psi_i}^i(s_t, a_t, z_t) \\ \text{Uncertainty-Aware State Predictions:} \quad & \hat{s}_t^i \sim \hat{p}_{\psi_i}^i(\cdot | s_t, a_t, h_t) := \mathcal{N}(\bar{s}_{t+1}^i, \Sigma_t^i), \end{aligned} \quad (5)$$

where Σ_t^i is a diagonal covariance matrix which is inferred from a second head $N_{\psi_i}^i$ of the networks used to estimate the mean contact force, and $\hat{p}_{\psi_i}^i(\cdot | s_t, a_t, h_t) \approx p_s(\cdot | s_t, a_t, e_t)$ is the i -th member in our ensemble of probabilistic dynamics models which aim to infer the next robot state from the history of available measurements.

Auto-Regressive State Predictions: Our ensemble of probabilistic dynamics models is used to hallucinate k -step synthetic rollouts used for policy optimization, summarized in Algorithm 1. Given a state s_t and state history h_t , to generate a synthetic rollout we: (i) an action is sampled from the policy, (ii) a model in the ensemble is randomly chosen, and then (iii) an uncertainty-aware prediction is computed with (5). This prediction is added to the state history and then both are used to sample the next action from the policy. By propagating state predictions and incorporating them back into the state history, we are implicitly propagating our estimate for the latent environment states z_t forwards in time, without needing to explicitly learn a predictive model for the how the environment changes over time. In particular, the multi-step prediction losses we introduce below ensure that this auto-regressive representation for the robot and environments carries enough information to be predictive of the robot’s state far into the future, which is our ultimate goal.

Algorithm 1 Auto-Regressive State Predictions

- 1: **Inputs** hallucination buffer $\mathcal{D}_{\text{model}}$, models $\{\hat{p}_{\psi_i}^i\}$, policy π_θ , start state s_0 , start history h_0
 - 2: **for** $t = 0 \dots k - 1$ **do**
 - 3: Sample action $a_t \sim \pi_\theta(\cdot | s_t, h_t)$
 - 4: Randomly choose model $i \sim \mathcal{U}[1, \dots, P]$ and predict next state \hat{s}_{t+1}^i with (5)
 - 5: Update state history h_{t+1} with \hat{s}_{t+1}^i and $s_{t+1} \leftarrow \hat{s}_{t+1}^i$
 - 6: Compute reward r_t and termination d_t and add transition $(s_t, a_t, r_t, s_{t+1}, d_t)$ to $\mathcal{D}_{\text{model}}$
 - 7: **Return** $\mathcal{D}_{\text{model}}$
-

162

Approximate Maximum Likelihood Estimation: To train our ensemble of probabilistic dynamics models, we maximize the joint-likelihood of state predictions along these synthetic rollouts, resulting in a multi-step loss approach demonstrated to enhance performance in MBRL [31, 32]. Due to the Markov assumption (Section 2), the joint distribution of the next H states, starting at state s_t is $p_s(s_{t+1:t+H} | s_t, a_{t:t+H}, e_{t:t+H}) = \prod_{j=0}^H p_s(s_{t+1+j} | s_{t+j}, a_{t+j}, e_{t+j})$. Approximating the robot transition dynamics with our probabilistic dynamics models and expressing the joint-likelihood as the negative-log-likelihood, we arrive at our loss function for the i -th model:

$$\mathcal{L}(\psi_i) = \frac{1}{HN_e} \sum_{t=h}^{N_e-H} \sum_{j=0}^H [\bar{s}_{t+1+j}^i - s_{t+1+j}]^\top (\Sigma_{t+j}^i)^{-1} [\bar{s}_{t+1+j}^i - s_{t+1+j}] + \log \det \Sigma_{t+j}^i. \quad (6)$$

Here, N_e is the size of a buffer of real-world transitions \mathcal{D}_{env} , mean state predictions are propagated deterministically according to $\bar{s}_{t+1}^i = S^i(\bar{s}_t^i, a_t, h_t)$, and each prediction \bar{s}_t^i is used to update the state history h_t . For each state s_t in the buffer, (6) generates a synthetic rollout H steps long and computes a loss related to how much the propagated states differ from the experienced states. Optimizing the joint-likelihood across multiple steps leads to improved reinforcement learning (RL) performance compared to a single-step loss approach (Section 4.2) because (i) model updates are enriched as upstream predictions influence downstream predictions, and (ii) predictions are smoothed over the synthetic rollout horizon, effectively filtering out noise between transitions.

3.2 Policy Optimization

Our approach (Algorithm 2) to policy optimization is based on the well-known MBPO [6] algorithm, but with two key differences: (i) the robot acts deterministically in the real world, and (ii) multiple steps are performed in the real world before updating the policy. As we demonstrate in Section 4, these changes lead to substantial improvements in sample-efficiency and ultimate performance.

Deterministic policy. Steps in the real environment are taken using deterministic policy μ_θ which simply outputs the mean action from the stochastic policy. Using a deterministic policy in the real world helps to prevent damage to the robot and its actuators; however, poses an immediate question regarding exploration. We employ a stochastic policy during hallucination per Algorithm 1 in order to ensure adequate exploration during the learning process; the stochasticity of our ensemble of predictive models also aids in exploration, but they also embed sufficient structure to accurately extrapolate during random exploration. Policy parameters θ are trained with soft-actor critic (SAC) [33], using a mixture of transitions from real-world and hallucination buffers \mathcal{D}_{env} and $\mathcal{D}_{\text{model}}$.

Enabling real-world training. In the original model-based policy optimization (MBPO) implementation [6], hallucination and policy updates are performed after every environment step, which is not computationally feasible for real-world training. Crucially, we add Line 5 to Algorithm 2 which allows rollouts of length N_E to be collected in the real world before hallucinating and updating the policy. This addition also creates a loop where hallucination is performed, the policy is updated, and hallucination is repeated with the updated policy. This iterative process results in multiple policies being used to generate hallucinations, leading to more diverse hallucinated experiences, and boosts the update-to-data (UTD) ratio in this scenario where longer rollouts are collected in the real world.

Algorithm 2 Policy Optimization with Semi-structured Dynamics Models

```

1: Initialize models  $\hat{p}_{\psi_i}$ , policy  $\pi_\theta$ , critics  $Q_{\phi_i}$ 
2: for  $N_{\text{epochs}}$  epochs do
3:   Train models  $\hat{p}_{\psi_i}$  on  $\mathcal{D}_{\text{env}}$  using loss (6)
4:   Take  $N_E$  steps in the environment deterministically with  $\mu_\theta$  and add transitions to  $\mathcal{D}_{\text{env}}$ 
5:   for  $K$  hallucination updates do
6:     for  $M$  model rollouts do
7:       Sample state  $s_0$  uniformly from  $\mathcal{D}_{\text{env}}$  and hallucinate with Algorithm 1
8:     Perform  $G$  updates of policy  $\pi_\theta$  using mixture of  $\mathcal{D}_{\text{env}}$  and  $\mathcal{D}_{\text{model}}$  at ratio  $r_D$ 

```

4 Experimental Results

4.1 Real-world Results

We demonstrate our approach through two real-world experiments where a Unitree Go1 quadruped is trained from scratch to achieve maximum speed on both hard ground and memory foam.

Experimental setup. Training is performed from scratch in the real-world according to Algorithm 2 over $N_{\text{epochs}} = 18$ epochs with $N_E = 1000$ environment steps per epoch, resulting in 18,000 environment steps or only 3.0 min of interaction with the real-world. We perform up to $K = 1,000$ hallucination updates per epoch and $G = 40$ gradient updates per hallucination update—resulting in a UTD of 40—and 20,000 actor and critic updates between rollouts in the real world. We use an observation history length of $h = 5$, a multi-step loss horizon of $H = 4$ to train the model, and hallucinate synthetic rollouts up to $k = 20$ steps long; the full listing of hyperparameters is found in the supplementary material. We reset the model, actor, and critic at 10,000 steps to improve plasticity [34]. Joint positions and velocities are measured from joint-level encoders, body orientation and angular velocity are obtained from the onboard IMU, and body linear velocity is acquired through a Vicon motion capture system. Neural networks are trained using JAX [35] and low-level joint angle commands are sent to the Go1 via Unitree’s ROS interface [36]. We compute the mass matrix, Coriolis terms, and gravity vector in (3) with the differentiable simulator Brax

[37]. Training the probabilistic dynamics models with (6) requires taking gradients through these computations, which is facilitated by the simulator’s differentiability.

Results. Fig. 1 shows a time-lapse of rollouts generated by intermediate policies as training progresses. As shown in Fig. 4 (right), after only 3.0 min of real-world training data, the quadruped achieves an average velocity of 0.98 m s^{-1} on hard ground and 0.53 m s^{-1} on memory foam. Figure 4 (left) plots the reward accumulated in each episode up to the first termination (i.e., at which $d_t = 1$). Early in training, the quadruped is prone to falling over and this leads to low reward, but after roughly 1.5 min of training the learned policy is robust enough to avoid falling, and subsequent rewards increase steadily. When walking on memory foam, the robot’s feet sink deeply, which makes training more difficult. Nonetheless, forward velocity steadily improves in both scenarios despite their significantly different contact dynamics, demonstrating the adaptability of our approach.

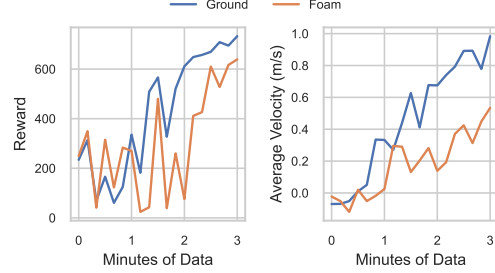


Figure 4: Real-world results. With our approach, the quadruped steadily learns to walk faster.

4.2 Simulated Experiments

In addition to the results presented here, we provide extensive ablations on standard RL benchmarks in the supplementary. Here, we investigate the following hypotheses:

Hypothesis 1. *Utilizing semi-structured dynamics models improves performance and sample efficiency for MBRL in contact-rich environments.*

Hypothesis 2. *Training semi-structured dynamics models with a multi-step loss enhances performance compared to using a single-step loss.*

Hypothesis 3. *Uncertainty-aware predictions from semi-structured dynamics models are robust against errors in a priori knowledge of the robot’s inertial properties.*

Experimental setup. The setup for experiments performed in simulation is similar to the the real-world setup (Section 4.1), except environment rollouts are simulated in Brax [37]; exact parameters used during training may be found in the supplementary material. To study the importance of using semi-structured dynamics models per Hypothesis 1, we compare our semi-structured state predictions (4) to black-box predictions of the form: $\bar{s}_{t+1}^i = s_t + B_{\psi_i}^i(s_t, a_t, z_t)$, where $\{B_{\psi_i}^i\}$ are an ensemble of networks. Learned additive Gaussian noise is also applied to the black-box model, resulting in the same probabilistic ensemble model found in MBPO [6], except this model receives the latent encoding and does not predict rewards since the reward function is provided. We perform runs with both a single-step loss ($H = 1$) and multi-step loss ($H = 4$) to test Hypothesis 2. We also benchmark against SAC [33], allowing the agent to act stochastically in the environment for this algorithm only. Finally, to simulate the modeling errors of Hypothesis 3, we randomly vary each link’s mass by $\pm 25\%$ and each joint’s damping by $\pm 50\%$ for the Go1 environment used for simulated data collection. All runs are repeated for 4 random seeds.

Results. The results of our simulated experiments are presented in Fig. 5. In Fig. 5 (left), we observe that our semi-structured dynamics models lead to significantly improved performance when compared with black-box models, supporting Hypothesis 1. By incorporating physics-based knowledge, our models produce hallucinated rollouts that generalize beyond the available training data, providing richer synthetic data. We also observe in Fig. 5 (left) that, while black-box models show similar performance for both single- and multi-step losses, our semi-structured dynamics models exhibit significantly improved performance when using a multi-step loss, confirming Hypothesis 2. Finally, in Fig. 5 (right), training is similar in performance when there are errors in the *a priori* knowledge of the robot’s inertial properties, supporting Hypothesis 3.

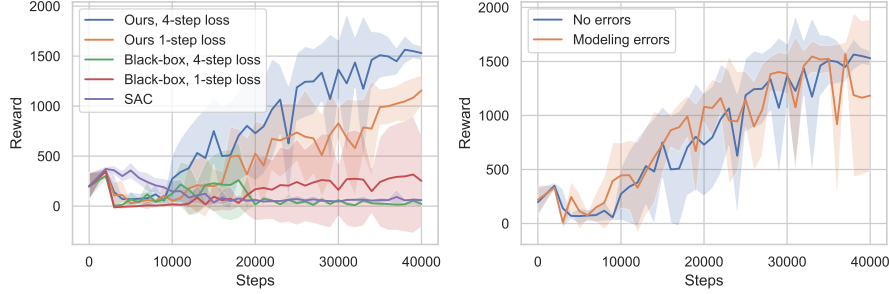


Figure 5: Simulated results. Left—using our semi-structured dynamics models and multi-step loss for training results in better performance. Right—our approach is robust to errors in *a priori* knowledge of the robot’s inertial properties. Plots show the mean and standard deviation for episodic rewards.

5 Related Work

Due to space constraints, we provide an abridged discussion of related work here and leave a fuller discussion to the supplementary material.

Model-Based Reinforcement Learning: Our work builds on a wealth of prior works that use general function approximators and probabilistic modelling to account for uncertainty when identifying dynamics that are impractical model by hand [38, 6, 26]. Model-based reinforcement learning algorithms either learn a model that is used for online planning [39, 26, 40] or Dyna-style algorithms which hallucinate imagined rollouts for direct policy optimization [6, 3, 4]. While we leverage the latter for the purposes of this work, we believe that our general modelling strategies can be use gainfully with online planning algorithms. We build on the insights of these works by integrating their insights with structured system identification techniques.

Learning External Contact Forces: As we have emphasize throughout the this work, inferring the information needed to fully localize and estimate contact forces acting on a robot may be impractical in the real-world, given limitations of on-board sensing modalities. Prior work [41, 15] leverages complementary formulations to predict contact forces, but relies on the availability of a signed-distance function to represent constraints, which may be impractical to construct and evaluate in the real-world. In [42] contacts are inferred from proprioception, but the method assumes the availability of onboard contact force sensor measurements.

Learning Locomotion Strategies in the Real World: Learning locomotion behaviors from scratch directly in the real-world has primarily been studied in the context of model-free reinforcement learning [43, 44, 45, 46], with a few works using black-box models in the context of model-based reinforcement learning [47, 48]. Several other works investigate fine-tuning locomotion controllers trained in simulation to reduce the burden on real world data [49, 50]. Compared to these works, our semi-structured modelling approach enable the robot to achieve dynamic locomotion strategies than these previous approaches, with just a fraction of the real-world samples.

6 Limitations

This paper presents a novel framework for model-based reinforcement learning, which leverages physics-informed, semi-structured dynamics models to enable highly sample-efficient policy learning in the real world. However there are several key limitations. First, our method requires observability of enough proprioceptive states to propagate the Lagrangian dynamics of the robot. Additionally, relying solely on proprioception restricts the model’s ability to predict changes to the environment such as the appearance of an obstacle or transitions between different ground surfaces. In the future we plan to extend the current framework to include additional perceptual modalities which can infer more about the state of the environment around the robot.

References

- [1] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.
- [2] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.
- [3] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.
- [4] R. S. Sutton, C. Szepesvári, A. Geramifard, and M. P. Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. *arXiv preprint arXiv:1206.3285*, 2012.
- [5] H. Yao, S. Bhatnagar, D. Diao, R. S. Sutton, and C. Szepesvári. Multi-step dyna planning for policy evaluation and control. *Advances in neural information processing systems*, 22, 2009.
- [6] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.
- [7] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33: 14129–14142, 2020.
- [8] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.
- [9] X. Chen, C. Wang, Z. Zhou, and K. Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.
- [10] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *arXiv preprint arXiv:2110.02034*, 2021.
- [11] J.-J. E. Slotine and W. Li. On the adaptive control of robot manipulators. *The international journal of robotics research*, 6(3):49–59, 1987.
- [12] S. Sastry and M. Bodson. *Adaptive control: stability, convergence and robustness*. Courier Corporation, 2011.
- [13] T. Westenbroek, D. Fridovich-Keil, E. Mazumdar, S. Arora, V. Prabhu, S. S. Sastry, and C. J. Tomlin. Feedback linearization for uncertain systems via reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1364–1371. IEEE, 2020.
- [14] G. Tao. *Adaptive control design and analysis*, volume 37. John Wiley & Sons, 2003.
- [15] F. Djeumou, C. Neary, E. Goubault, S. Putot, and U. Topcu. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Learning for Dynamics and Control Conference*, pages 263–277. PMLR, 2022.
- [16] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.
- [17] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=BklHpjCqKm>.
- [18] B. Acosta, W. Yang, and M. Posa. Validating robotics simulators on real-world impacts. *IEEE Robotics and Automation Letters*, 7(3):6471–6478, 2022.

- [19] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.
- [20] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.
- [21] P. Fankhauser, M. Bloesch, and M. Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4):3019–3026, 2018.
- [22] R. Yang, G. Yang, and X. Wang. Neural volumetric memory for visual locomotion control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1430–1440, 2023.
- [23] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis. Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *IEEE Transactions on Robotics*, 38(5):2908–2927, 2022.
- [24] L. Manuelli and R. Tedrake. Localizing external contact using proprioceptive sensors: The contact particle filter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5062–5069. IEEE, 2016.
- [25] S. Haddadin, A. De Luca, and A. Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6):1292–1312, 2017.
- [26] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.
- [27] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
- [28] G. Bellegarda and A. Ijspeert. Cpg-rl: Learning central pattern generators for quadruped locomotion. *IEEE Robotics and Automation Letters*, 7(4):12547–12554, 2022.
- [29] T.-Y. Yang, T. Zhang, L. Luu, S. Ha, J. Tan, and W. Yu. Safe reinforcement learning for legged locomotion. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2454–2461. IEEE, 2022.
- [30] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.
- [31] M. Lutter, L. Hasenclever, A. Byravan, G. Dulac-Arnold, P. Trochim, N. Heess, J. Merel, and Y. Tassa. Learning dynamics models for model predictive agents. *arXiv preprint arXiv:2109.14311*, 2021.
- [32] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.
- [33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
- [34] E. Nikishin, M. Schwarzer, P. D’Oro, P.-L. Bacon, and A. Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pages 16828–16847. PMLR, 2022.

- [35] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/google/jax>.
- [36] U. Robotics. Unitree ros to real. https://github.com/unitreerobotics/unitree_ros_to_real, 2021.
- [37] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax—a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.
- [38] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.
- [39] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.
- [40] S. Racanière, T. Weber, D. Reichert, L. Buesing, A. Guez, D. Jimenez Rezende, A. Puigdomènech Badia, O. Vinyals, N. Heess, Y. Li, et al. Imagination-augmented agents for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.
- [41] S. Pfrommer, M. Halm, and M. Posa. Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations. In *Conference on Robot Learning*, pages 2279–2291. PMLR, 2021.
- [42] D. Lim, M.-J. Kim, J. Cha, D. Kim, and J. Park. Proprioceptive external torque learning for floating base robot and its applications to humanoid locomotion. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8510–8517. IEEE, 2023.
- [43] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 3, pages 2619–2624. IEEE, 2004.
- [44] R. Tedrake, T. W. Zhang, and H. S. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2849–2854. IEEE, 2004.
- [45] L. Smith, I. Kostrikov, and S. Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.
- [46] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng. Learning cpg sensory feedback with policy gradient for biped locomotion for a full-body humanoid. In *AAAI*, pages 1267–1273, 2005.
- [47] S. Choi and J. Kim. Trajectory-based probabilistic policy gradient for learning locomotion behaviors. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2019.
- [48] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR, 2020.
- [49] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1593–1599. IEEE, 2022.
- [50] T. Westenbroek, F. Castaneda, A. Agrawal, S. Sastry, and K. Sreenath. Lyapunov design for robust and efficient robotic reinforcement learning. *arXiv preprint arXiv:2208.06721*, 2022.

- 429 [51] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies.
430 *Journal of Machine Learning Research*, 17(39):1–40, 2016.
- 431 [52] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A
432 general safety framework for learning-based control in uncertain robotic systems. *IEEE Trans-*
433 *actions on Automatic Control*, 64(7):2737–2752, 2018.
- 434 [53] R. Calandra, S. Ivaldi, M. P. Deisenroth, E. Rueckert, and J. Peters. Learning inverse dynamics
435 models with contacts. In *2015 IEEE International Conference on Robotics and Automation*
436 *(ICRA)*, pages 3186–3191. IEEE, 2015.
- 437 [54] J. Hwangbo, C. D. Bellicoso, P. Fankhauser, and M. Hutter. Probabilistic foot contact esti-
438 mation by fusing information from dynamics and differential/forward kinematics. In *2016*
439 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3872–
440 3878. IEEE, 2016.
- 441 [55] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots.
442 *arXiv preprint arXiv:2107.04034*, 2021.
- 443 [56] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust per-
444 ceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822,
445 2022.
- 446 [57] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforce-
447 ment learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE*
448 *International Conference on Robotics and Automation (ICRA)*, pages 2811–2817. IEEE, 2021.

A Implementation Details

In this appendix, we provide details of our implementation for the Unitree Go1 Quadruped, including the observation and action spaces, the reward function, the termination condition, and the control architecture.

A.1 Observation and Action Spaces

The observation space $\Omega \subset \mathbb{R}^{36}$ consists of the elements in Table 1. The x -axis of the base is the forward direction, the y -axis is the leftward direction, and the z -axis is the upward direction. The phase variable $\phi \in [0, 2\pi)$ represents progression along the gait cycle and is defined as $\phi_t = 2\pi t/T_\phi \bmod (2\pi)$ where $T_\phi = 0.5$ sec is the gait cycle period.

Observation	Symbol	Dimension
Quaternion orientation of the base	φ	4
Joint angles	q^j	12
Base linear velocity (local frame)	(v^x, v^y, v^z)	3
Base angular velocity (local frame)	$(\omega^x, \omega^y, \omega^z)$	3
Joint speeds	\dot{q}^j	12
Cosine of phase	$\cos \phi$	1
Sine of phase	$\sin \phi$	1

Table 1: Observation space.

The action space $\mathcal{A} \subset \mathbb{R}^9$ outputs the change in nominal height for the gait generator and offsets to nominal foot positions from the gait generator, as defined in Table 2.

Action	Symbol	Dimension	Min.	Max.
x -foot position changes	Δp^x	4	−0.15 m	0.15 m
y -foot position changes	Δp^y	4	−0.075 m	0.075 m
Change in gait generator nominal height	Δh^{gait}	1	−0.1 m	0.0 m

Table 2: Action space.

A.2 Reward Function and Termination Condition

Reward Function. The reward function is a weighted sum of the terms in Table 3. We set the weights and use exponentials in most of the terms to normalize the reward such that a forward velocity of 1.0 m s^{-1} with maximal values for all other terms will result in a reward of approximately 1.0 for a single time step. The roll φ^x , pitch φ^y , and yaw φ^z of the base are obtained from the base quaternion φ . We define $a \odot b$ as the element-wise multiplication of vectors a and b . Actual torques output from the joint-level PD controllers are not available; we estimate the torque applied at the joint with (12). We define the following LinearLimit function which linearly penalizes the torque applied at the j -th joint τ^j when exceeding torque limits; within torque limits, the function is a decaying exponential:

$$\text{LinearLimit}(\tau^j, \tau_{\min}^j, \tau_{\max}^j) = \begin{cases} \tau^j - \tau_{\min}^j - 1 & \text{if } \tau < \tau_{\min}^j \\ -\exp\left[-\tau^j + \tau_{\min}^j\right] & \text{if } \tau_{\min}^j \leq \tau^j < \tau_{\max}^j \\ -\exp\left[\tau^j - \tau_{\max}^j\right] & \text{if } 0 \leq \tau^j < \tau_{\max}^j \\ -\tau^j + \tau_{\max}^j - 1 & \text{if } \tau^j \geq \tau_{\max}^j. \end{cases} \quad (7)$$

Termination Condition. The termination flag d_t stops the accumulation of reward after the quadruped falls and is defined by:

$$d_t = \begin{cases} 1 & \text{if } |\varphi_t^x| > \pi/4 \text{ or } |\varphi_t^y| > \pi/4 \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

Reward Term	Expression	Weight
Maximize forward velocity	v_{t+1}^x	0.42
Limit base yaw rate	$\exp[-(\omega_{t+1}^z)^2/0.2]$	0.11
Limit base roll	$\exp[-(\varphi_{t+1}^x)^2/0.25]$	0.05
Limit base pitch	$\exp[-(\varphi_{t+1}^y)^2/0.25]$	0.05
Limit base yaw	$\exp[-(\varphi_{t+1}^z)^2/0.07]$	0.11
Limit base side velocity	$\exp[-(v_{t+1}^y)^2/0.01]$	0.11
Limit vertical acceleration	$\exp[-(v_{t+1}^z - v_t^z)^2/0.02]$	0.03
Limit base roll rate	$\exp[-(\varphi_{t+1}^x - \varphi_t^x)^2/0.001]$	0.03
Limit base pitch rate	$\exp[-(\varphi_{t+1}^y - \varphi_t^y)^2/0.005]$	0.03
Limit energy	$\exp[-\ \dot{q}_{t+1}^j \odot \tau_{t+1}\ _1^2/450]$	0.05
Penalize excessive torques	$\sum_j \text{LinearLimit}(\tau_t^j, \tau_{\min}^j, \tau_{\max}^j)/12$	0.02

Table 3: Reward function terms. The reward at each time step is a weighted sum of these terms.

472 A.3 Control Architecture

473 Here, we give detailed specification of the control architecture introduced in Section 2. Referring
474 to the action space definition Table 2, the policy takes in the current observation and a history
475 of observations and outputs offsets to foot positions and a nominal height for the gait generator:
476 $(\Delta p_t^x, \Delta p_t^y, \Delta h_t^{\text{Gait}}) \sim \pi_\theta(\cdot | s_t, h_t)$. The gait generator $\text{Gait} : [0, 1) \times \mathbb{R} \rightarrow \mathbb{R}^3$ is open-loop and
477 generates for each leg, walking-in-place foot positions for the quadruped by computing vertical foot
478 position offsets from nominal standing foot positions:

$$\text{Gait}(\bar{\phi}_t^l; \Delta h_t^{\text{Gait}}) = \begin{cases} p_{\text{stand}}^l + [0, 0, h^{\text{Swing}} \left(1 - \cos \left[2\pi \frac{\bar{\phi}_t^l - r^{\text{Gait}}}{1 - r^{\text{Gait}}}\right)\right] - \Delta h_t^{\text{Gait}} & \text{if } \bar{\phi}^l \geq r^{\text{Gait}} \\ p_{\text{stand}}^l + [0, 0, \Delta h_t^{\text{Gait}}] & \text{otherwise} \end{cases} \quad (9)$$

479 where $p_{\text{stand}}^l \in \mathbb{R}^3$ is the nominal standing foot position of the l -th leg, expressed in the local base
480 frame, $h^{\text{Swing}} = 0.09$ m is the gait peak swing height, and $r^{\text{Gait}} = 0.5$ is that fraction of the time feet
481 should remain in contact with the ground. The normalized phase $\bar{\phi}^l \in [0, 1)$ specifies the progress
482 of the l -th leg along its gait cycle and is calculated with:

$$\bar{\phi}_t^l = \left(\frac{\phi_t}{2\pi} + 0.5 + b^l \right) \bmod 1, \quad (10)$$

483 where b^l is the phase bias for the l -th leg; we use a value of 0 for the front-right and rear-left legs,
484 and a value of 0.5 for the front-left and rear-right legs. The desired positions of the l -th foot in the
485 local base frame are given by:

$$p_t(\Delta p_t^{x,l}, \Delta p_t^{y,l}, \Delta h_t^{\text{Gait}}, \bar{\phi}_t^l) = [\Delta p_t^{x,l}, \Delta p_t^{y,l}, 0] + \text{Gait}(\bar{\phi}_t^l; \Delta h_t^{\text{Gait}}), \quad (11)$$

486 where $\Delta p_t^{x,l}$ and $\Delta p_t^{y,l}$ are the x - and y -foot positions offsets for the l -th foot from the policy. For
487 each foot, the desired foot positions (11) are computed and sent to an inverse kinematics solver to
488 produce desired joint angles $q^{\text{des}} \in \mathbb{R}^{12}$. The desired joint angles are sent to the joint level PD
489 controllers, where the desired torque outputs are:

$$\tau_t = K_p(q^{\text{des}} - q^j) - K_p \dot{q}^j, \quad (12)$$

490 and we use proportional gain $K_p = 112$ N m rad⁻¹ and derivative gain $K_p = 3.5$ N m s rad⁻¹.

491 B Simulated Benchmark Experiments

492 To demonstrate the versatility of our approach, we perform additional simulated experiments using
493 standard, contact-rich, benchmark environments [20] commonly used to evaluate RL algorithms.

Experimental setup. We use the standard MuJoCo [20] environments Hopper, Walker2d, and Ant, which have been implemented as part of Brax [37]. Similar to the quadruped, each of these environments feature a floating-base robot with articulated limbs which make and break contact with the ground to produce motion. However, unlike the Go1 environment, these environments lack structured controllers. Instead, the outputs from the policy are only scaled linearly before being directly applied as torques on the joints. To test [Hypothesis 1](#) and [Hypothesis 2](#), we compare our semi-structured approach trained with a multi-step loss ($H = 4$) to the black-box approach from [Section 4.2](#) trained with the single-step loss ($H = 1$). In both of these cases, the agent acts deterministically within the environment per [Algorithm 2](#). We also benchmark against SAC [33], allowing the agent to act stochastically in the environment for this algorithm only. The hyperparameters used for training are found in [Appendix C](#) and all runs are repeated for 4 random seeds.

Results. The results of these experiments are presented in [Fig. 6](#). We observe a significant performance improvement when utilizing our semi-structured models trained with a multi-step loss, compared to the black-box approach trained with a single-step loss, confirming [Hypothesis 1](#) and [Hypothesis 2](#). These results demonstrate that our approach works not only with the Go1 environment, but also with other contact-rich environments with unstructured controllers.

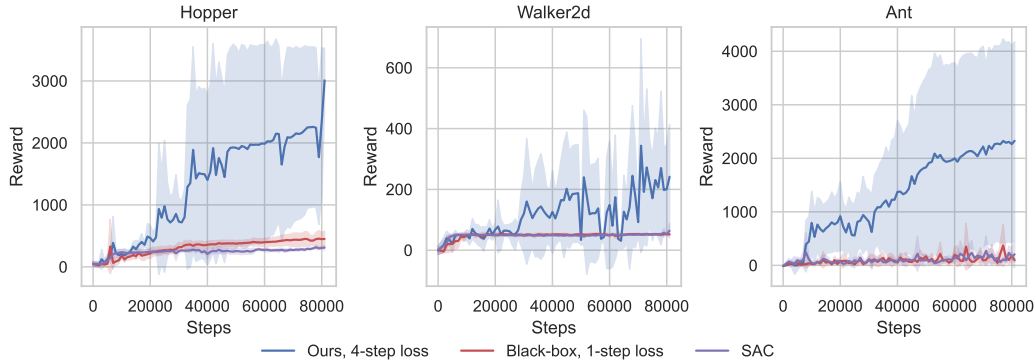


Figure 6: Simulated benchmark results. Better performance is achieved when using our semi-structured dynamics models and a multi-step loss. Plots show the mean and standard deviation for episodic rewards.

C Experiment Hyperparameters

[Table 4](#) contains the hyperparameters used with our approach; these hyperparameters were also used with the approach that incorporated black-box models. [Table 5](#) contains the SAC hyperparameters used for our approach, the black-box approach, and standard SAC.

D Expanded Related Work

Here we provide an extended related work beyond, expanding on the context provided in the main submission.

Reinforcement Learning: Reinforcement learning algorithms are attractive because they enable robots to learn general control policies through repeated interactions with the world [49, 51, 50]. However, when learning to control a system from scratch, the sample complexity, safety, and reliability [52] of these algorithms remains a significant concern.

Our work builds on a wealth of prior works that use general function approximators and probabilistic modelling to account for uncertainty when identifying dynamics that are impractical to model by hand [38, 6, 26]. Model-based reinforcement learning algorithms either learn a model that is used for online planning [39, 26, 40] or Dyna-style algorithms which hallucinate imagined rollouts for direct policy optimization [6, 3, 4]. While we leverage the latter for the purposes of this work, we

Hyperparameter	Go1 (real world)	Go1 (simulated)	Benchmarks
Epochs, N_{epochs}	18	40	80
Environment steps per epoch, N_E	1000		
Hallucination updates per epoch, K	10 \rightarrow 1,000 over epochs 0 \rightarrow 4		
Model rollouts per hallucination update, M	400		
Synthetic rollout length, k	1 \rightarrow 20 over epochs 0 \rightarrow 10		1 \rightarrow 45 over epochs 0 \rightarrow 15
Real to synthetic data ratio, $r_{\mathcal{D}}$	0.06		
Gradient updates per hallucination update, G	40	60	20
State history length, h	5		1
Multi-step loss horizon, H	4	1 or 4	
Model learning rate	1×10^{-3}		
Model training batch size	200		

Table 4: Hyperparameters for our approach and the baseline approach with black-box models. $x \rightarrow y$ over epochs $a \rightarrow b$ denotes a clipped linear function, i.e. at epoch i , $f(i) = \text{clip}(x + \frac{i-a}{b-a}(y-x), x, y)$.

Hyperparameter	Go1 (real world)	Go1 (simulated)	Benchmarks
Learning rate	2×10^{-3}		3×10^{-3}
Discount factor, γ	0.99		
Batch size	256		
Target smoothing coefficient, τ	1×10^{-3}		5×10^{-3}
Actor network (MLP) width \times depth	512×2		256×2
Critic network (MLP) width \times depth	512×2		256×2

Table 5: SAC hyperparameters used for our approach, the black-box approach, and standard SAC.

526 believe that our general modelling strategies can be use gainfully with online planning algorithms.
527 We build on the insights of these works by integrating their insights with structured system identifica-
528 tion techniques, substantially accelerating our ability to learn in the real-world for our quadrupedal
529 case-study.

530 While our work seeks to make model-based data-augmentations strategies more accurately reflect
531 the true dynamics of the system, a parallel line of work [9, 10, 49] aims to make off-policy model-
532 free algorithms (which form the back-bone for our policy optimization strategy) more stable and ef-
533 ficient in low-data regimes. These approaches introduce regularization techniques which enable the
534 use of higher update-to-data ratios without overfitting to the available data, matching the efficiency
535 of model-free methods such as the MBPO [6] algorithm that we build upon. These algorithmic ad-
536 vances are generally orthogonal to our contribution, and thus in the future we plan to incorporate
537 them into our framework to further accelerate real-world learning.

538 **Learning External Contact Forces:** As we have emphasized throughout the this work, inferring
539 the information needed to fully localize and estimate contact forces acting on a robot may be im-
540 practical in the real-world, given limitations of on-board sensing modalities. Prior work [41, 15]
541 leverages complementary formulations to predict contact forces, but computing these forces relies
542 on the availability of a signed-distance representation of surfaces the robot is making contact with,
543 which may be impractical to construct and evaluate in the real-world with available on-board sen-
544 sors. In [42] contacts are inferred from proprioception, but the method assumes the availability of
545 onboard contact force sensor measurements. In [53], contact forces are learned directly from avail-
546 able measurements, but these models are not history-conditioned and attempt to reconstruct multiple
547 independent contacts which may be occurring at different locations on the robot. Altogether, while
548 we build upon perspectives from many prior works, we introduce an semi-structured auto-regressive
549 formulation for inferring contact which is compatible with standard MBRL algorithms, lightweight,
550 and capable of learning from on-board observations in the real-world.

551 **Learning Locomotion Strategies in the Real World:** Learning locomotion behaviors from scratch
552 directly in the real-world has primarily been studied in the context of model-free reinforcement
553 learning [43, 44, 45, 46], with a few works using black-box models in the context of model-based
554 reinforcement learning [47, 48]. Compared to these works, our semi-structured modelling approach
555 enable the robot to achieve more dynamic locomotion strategies than these previous approaches,
556 with just a fraction of the real-world samples. Specifically, our approach either achieves a signifi-
557 cantly higher walking speed than each of these approaches, or improves on their sample complexity
558 by approximately an order of magnitude. Several other works investigate fine-tuning locomotion
559 controllers trained in simulation to reduce the burden on real world data [49, 50] – as we discuss
560 below, we hope to investigate this direction in the near future.

561 **Direct Transfer From Simulation:** There has also been recent and rapid progress directly transfer-
562 ring locomotion controllers from simulation zero-shot [54, 55, 27, 56, 57], using techniques such as
563 domain adaptation and domain randomization. In this paper we have focused on learning locomo-
564 tion controllers from scratch, in an effort to demonstrate the ability of our framework to substantially
565 adapt the behavior of the robot with small amounts of real-world data. However, in the future we
566 plan to fine-tune policies that have been trained using extensive simulated experience, improving the
567 performance of these policies in cases where they fail [49] but leveraging a better initialization for
568 the policy for real-world learning.