UNDERSTANDING THE LIMITS OF LATENT WORLD MODELS

Rodrigo Gonzalez Laiz & Steffen Schneider

Helmholtz Munich
Munich Center for Machine Learning (MCML)

ABSTRACT

World models are widely proposed as core components for enabling agents to reason over long temporal and spatial horizons. Yet current evaluations often emphasize task performance while providing limited insight into whether learned world models acquire predictive latent dynamics that reflect the underlying structure of the environment. Here we introduce the concept of faithful latent world models as an equivariance condition between the environment's transition map and the learned latent dynamics. We propose latent rollout consistency (LRC) as a metric to evaluate action-conditioned latent prediction. Strikingly, we find that across the studied state of the art latent world models, none solves the task by actually faithfully modeling the dynamics of the environment.

</> Code Website

1 INTRODUCTION

Building agents that can reason over extended temporal and spatial scales is a major frontier in AI. It has been argued that general agents need *world models* if they are to accomplish complex, goal-directed tasks[1–3]. To support such behavior, world models must predict how the environment evolves under the agent's actions. Yet, latent world models are still evaluated primarily through task reward. While reward measures whether the overall agent succeeds, it does not reveal whether the learned latent dynamics capture the underlying dynamics of the environment.

This motivates the central question of this work: *Do successful latent world models actually learn representations and dynamics that model their environment?* Recent work has identified failures of physical understanding in related settings. For example, Joseph et al.[4] study passive video encoders such as V-JEPA and VideoMAE through probing and representation geometry but do not evaluate future rollouts. Motamed et al.[5] evaluate long-horizon observational video prediction from visual context using metrics in pixel space. In contrast, we study *action-conditioned dynamics* in agents that interact with and control an environment.

We evaluate DreamerV3, TD-MPC2, and LeWM[6–8] on three environments: Pendulum, CartPole, and Reacher. These environments are *deliberately simple* and serve as minimal test cases for world modeling. Their low-dimensional, well-defined dynamics allow us to isolate whether the learned world models accurately capture the underlying environment dynamics. Failures in these controlled settings therefore provide counterexamples and let us isolate mechanisms.

We call a latent world model *faithful* when its representation preserves the environment state and its latent dynamics evolve consistently with the true dynamics under the same actions. We measure this with latent rollout consistency (LRC) and post-hoc state decoding. Our results make two main points. First, task reward alone does not reveal whether a world model learns accurate latent dynamics or preserves the underlying physical state. LRC and state decoding expose these failures directly. Second, these diagnostics reveal two recurring failure modes under pixel observations: inaccurate velocity estimation and difficulty learning dynamics through nonlinear observation mappings. Together, these results suggest that improving both learned representations and latent dynamics is central to building more faithful world models.

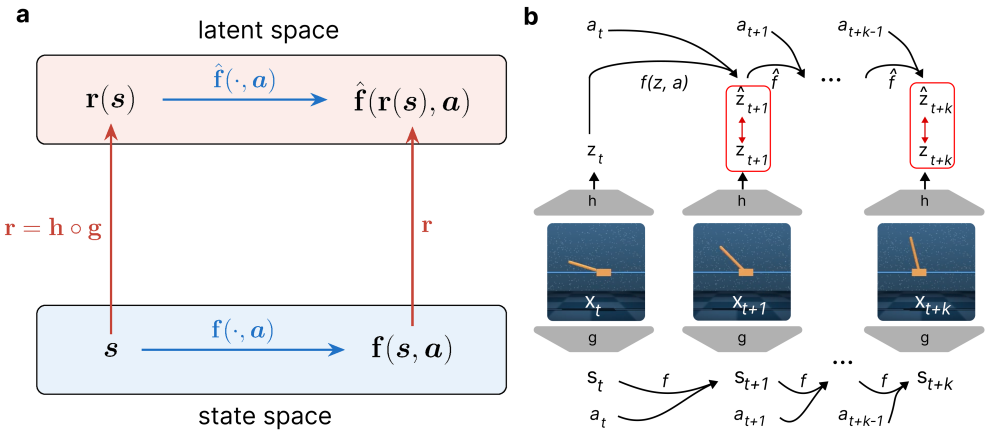


Figure 1: **Faithful latent dynamics and their evaluation.** (a) In a faithful latent world model, applying the true transition $\mathbf{f}$ and then encoding ($\mathbf{r}$) gives the same result as encoding first and applying the learned latent dynamics $\hat{\mathbf{f}}$ under the same action. (b) Latent rollout consistency (LRC) evaluates this by comparing autoregressive latent predictions $\hat{\boldsymbol{z}}_{t+k}$ with representations $\boldsymbol{z}_{t+k}$ inferred from the corresponding realized observations.

2 What does it take to model the world?

We model the environment as a partially observed controlled dynamical system. The considered environments have deterministic dynamics and observations,

$$\begin{aligned} \boldsymbol{s}_{t+1} &= \mathbf{f}(\boldsymbol{s}_t, \boldsymbol{a}_t), \\ \boldsymbol{x}_t &= \mathbf{g}(\boldsymbol{s}_t) \end{aligned} \tag{1}$$

where $\boldsymbol{s}_t$ is the unobserved environment state, $\boldsymbol{a}_t$ the action, and $\boldsymbol{x}_t$ the observation. The observation may be a nonlinear view of the state. Following nonlinear independent component analysis, we call $\mathbf{g}$ the mixing function[9].

A *latent* world model has two coupled roles: (1) infer, or "demix," a latent representation from observations, and (2) model how that representation evolves under actions. This gives the following definition of a latent world model:

$$\boldsymbol{z}_t = \mathbf{h}(\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{<t}) \tag{2}$$

$$\hat{\boldsymbol{z}}_{t+1} = \hat{\mathbf{f}}(\boldsymbol{z}_t, \boldsymbol{a}_t). \tag{3}$$

where $\mathbf{h}$ is the learned encoder and $\hat{\mathbf{f}}$ is the learned dynamics model. We use the term specifically for models that infer the action-conditioned latent dynamics governing an environment. Generative video models[10] and cognitive/spatial maps[11] are often forms of "world models" but are conceptually very different from the world models studied here, and hence, are out of scope for this analysis.

We are interested in the question: *when is a world model* $(\hat{\mathbf{f}}, \mathbf{h})$ *a good model of the world* $(\mathbf{f}, \mathbf{g})$*?* Two conditions follow naturally. First, the model should recover all information about the state from the observations. This condition is task-independent and allows the same world model to be used for different tasks, as long as those tasks are defined within the given state space. Second, predicting the evolution of an encoded state should agree with encoding the state after a true transition. This ensures that latent rollouts follow the dynamics of the environment. We define:

Definition 1: Faithful latent world model

Define the state-to-latent map $\mathbf{r} \coloneqq \mathbf{h} \circ \mathbf{g} : \mathcal{S} \to \mathcal{Z}$. The latent world model $(\mathbf{h}, \hat{\mathbf{f}})$ is *faithful* if:

1. *Dynamics equivariance:* $\hat{\mathbf{f}}(\mathbf{r}(\boldsymbol{s}), \boldsymbol{a}) = \mathbf{r}(\mathbf{f}(\boldsymbol{s}, \boldsymbol{a}))$ for all $\boldsymbol{s} \in \mathcal{S}$, $\boldsymbol{a} \in \mathcal{A}$.
2. *State sufficiency:* $\mathbf{r}$ is injective.

The first condition establishes an equivariance relation between the environment and latent dynamics: encoding after a true transition must agree with predicting in latent space. This is closely related to MDP homomorphisms [12], which preserve transition and reward structure, and to the consistency requirements of causal abstractions [13,14]. Unlike these frameworks, which allow information-losing abstractions, we focus on task-independent dynamics and additionally require injectivity to preserve the full environment state.

Learning the equivariance condition. The equivariance condition can, in principle, be optimized from observed transitions without access to the true state. Since $\boldsymbol{s}_{t+1} = \mathbf{f}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, the encoded next observation provides the target

$$\mathbf{h}(\boldsymbol{x}_{\leq t+1}, \boldsymbol{a}_{<t+1}) = \mathbf{r}(\mathbf{f}(\boldsymbol{s}_t, \boldsymbol{a}_t)). \tag{4}$$

Many approaches, therefore, match the predicted latent state with the encoded next observation,

$$\min_{\mathbf{h},\hat{\mathbf{f}}} \mathbb{E}\left[\left\|\hat{\mathbf{f}}(\mathbf{h}(\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{<t}), \boldsymbol{a}_t) - \mathbf{h}(\boldsymbol{x}_{\leq t+1}, \boldsymbol{a}_{<t+1})\right\|^2\right] + \lambda\Omega(\mathbf{h}, \hat{\mathbf{f}}). \tag{5}$$

Here, Ω broadly denotes additional learning signals used to prevent collapse. These include reconstruction losses [6,15], contrastive learning [16], regularization [17], and, in reward-based models, reward and value prediction [6,7].

Metrics for faithful world modeling. Definition 1 gives two conditions that we evaluate separately. We quantify empirical violations of dynamics equivariance using latent rollout consistency (LRC) and assess violations of injectivity using post-hoc state decoders. At rollout horizon H, we compare the autoregressively predicted latent $\hat{\boldsymbol{z}}_H$ with the latent inferred from the realized trajectory:

$$\mathrm{LRC}(H) = R^2(\mathbf{h}(\boldsymbol{x}_{\leq H}, \boldsymbol{a}_{<H}), \hat{\boldsymbol{z}}_H)\,. \tag{6}$$

A faithful model has $\mathrm{LRC}(H) = 100\%$ at every horizon. High LRC, however, is not sufficient on its own: model collapse can still produce latents that are easy to predict. We therefore use post-hoc state decoding to test how much of the underlying state is found in the representation. Good decoding performance supports state sufficiency but does not guarantee injectivity.

Violations of the faithful condition. From left to right, Figure 2 shows three ways a world model can fail to be faithful, using CartPole swingup with MPC and the true simulator as the planning model (Appendix F).

Column A uses an identity encoding and adds noise to the oracle transitions. Linear and nonlinear decoding stay at $R^2 = 1$, so probes cannot see the failure, while reward falls with LRC. Column B applies the same dynamics noise after an invertible nonlinear mix. Linear decoding is already poor at zero noise, because the coordinates are nonlinear, whereas nonlinear decoding stays at 1. Reward again tracks LRC rather than linear R^2. Column C keeps the transitions exact and corrupts the inverse encoder. Decoding remains high and reward can stay high even as LRC collapses: encoder error is amplified over the rollout before return or probes break.

These settings show that both the representation and the dynamics model can limit task performance. They also motivate evaluating latent prediction, state decoding, and reward together.

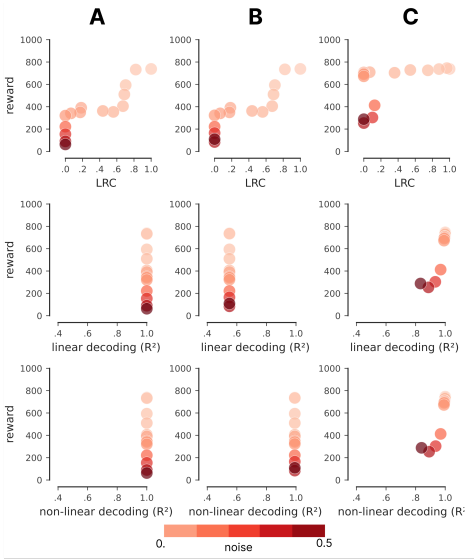


Figure 2: **A.** identity encoding, noisy dynamics; **B.** nonlinear encoding, noisy dynamics; **C.** noisy encoder, exact dynamics. LRC at $H = 30$, linear R^2, nonlinear R^2.

3 Related Work

World models and model-based RL. The role of explicit internal models in intelligent behaviour has been debated since Brooks [18], with Richens et al. [3] recently formalizing the opposing view that

any sufficiently general goal-directed agent must implicitly learn one. The modern formulation of *world models* as compact latent dynamics models learned by an agent from its own experience was popularized by Ha and Schmidhuber [1]. Contemporary world models span video generators (surveyed in Zhu et al. [19]), JEPA-style latent predictors [20–22] (LeWM [8] belongs to this family), and task-driven model-based RL agents—PlaNet [15], DreamerV3 [6], and TD-MPC2 [7]; diffusion-based variants such as DIAMOND [23] replace the latent dynamics with a generative model. Several recent works question whether such models actually capture environment physics: Motamed et al. [5] show that high-fidelity video models violate basic physical principles; Garrido et al. [24] evaluate V-JEPA with violation-of-expectation tests on IntPhys, GRASP, and InfLevel; and Joseph et al. [4] use layerwise probing, subspace analysis, and targeted ablations to study physical representations in V-JEPA 2 and VideoMAE-v2. Closest in spirit, Ying et al. [25] call for metrics that quantify how well an agent has inferred the latent dynamics beyond task success.

Probing learned representations. We build on the idea of probing classifiers [26]. Probes have revealed emergent latent state in Othello-GPT [27,28] and have been used to reverse-engineer trained RL agents, from DQN [29] and AlphaZero [30] to recent model-free agents and recurrent world models [31–34]. A complementary line probes the physical structure in foundation models [35,36]. We apply this idea to state-of-the-art world models to ask whether reliable physical content is recoverable in the first place.

Identifiable representation learning. Our analysis connects to identifiable representation learning, which studies when the underlying data-generating process can be provably recovered from observations [37,38], with foundations in nonlinear ICA [9,39] and has recently been studied in the context of latent world models [16]. Recently, Reizinger et al. [40] asked for more empirically grounded identifiability theory in self-supervised learning, a position our results reinforce: current world-model objectives fail to invert the data-generating process and learn faithful environment dynamics. This might be due mismatches in the assumptions of theoretically-motivated papers, or in properties of the numerical implementations that do not fully match the theory.

4 Experimental Setup

4.1 Benchmarks and Training

We evaluate three complementary world-model families spanning two objective axes (Figure 3a) on three DeepMind Control Suite [41] environments (Figure 3b). DreamerV3 is reconstruction-based and reward-driven/task-specific, TD-MPC2 is reconstruction-free but reward-driven/task-specific, and LeWM is reconstruction-free and reward-free/task-agnostic, learning through a self-supervised latent-prediction objective and performing model-predictive control (MPC) as a post-hoc step.

We use three low-dimensional continuous-control environments: Pendulum (d_s=3), CartPole (d_s=5), and Reacher (d_s=6). Pendulum and CartPole exhibit nontrivial unforced dynamics, whereas Reacher remains stationary from rest without control; Appendix A provides further details. In all three cases, we have access to the ground-truth simulator state and dynamics, allowing us to evaluate what the models learn rather than relying on task reward alone [42]. These three environments are not meant to form a comprehensive benchmark. They are deliberately simple test cases. Doing well on them does

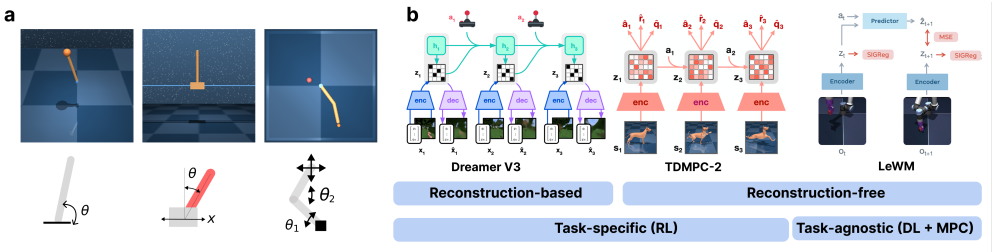


Figure 3: **Experimental setup.** (a) Three world model families: TD-MPC2 and DreamerV3 (task-specific) and LeWM (dynamics learning + MPC), grouped by reconstruction and training objective. (b) Three DeepMind Control Suite environments—Pendulum, CartPole, and Reacher—evaluated with state and pixel observations.

not imply that a model will scale to harder settings, but if it cannot learn accurate dynamics here, it is difficult to expect it to do so in more complex systems.

We train each model family on both raw state vectors and rendered RGB images (64×64 px for TD-MPC2 and DreamerV3; 224×224 px for LeWM). Because the original LeWM implementation does not support state inputs, we implement a state-based variant to enable a direct comparison with the other models and with pixel-based LeWM. We train four seeds per model, modality, and environment. Implementation and training details are provided in Appendix C.

4.2 Evaluation protocol

For each environment, we collect a separate evaluation set of 1,000 100-step episodes using a random Ornstein–Uhlenbeck (OU) policy designed to provide broad state-space coverage. Appendix A describes the data-collection procedure and compares the state, action, and reward distributions of the collected datasets in Figures 8–10.

Latent prediction (LRC). We compute LRC (Eq. 6) from one autoregressive latent rollout initialized at the beginning of each episode. Each model is given the recorded action sequence, and its predicted latents are compared with the latents inferred from the corresponding observed trajectory. At each horizon, we compute R^2 across episodes separately for each latent coordinate and aggregate coordinates using target-variance weighting. Scores are computed separately for each environment and model seed, and each model condition is summarized by its mean across four seeds (Appendix B). As a state-space topline, we train a two-hidden-layer MLP to predict $\boldsymbol{s}_{t+1}$ from $(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and roll it out autoregressively. We also report persistence ($\hat{\boldsymbol{s}}_{t+H} = \boldsymbol{s}_0$) as a baseline. For both state-space baselines, we set $\boldsymbol{z}_t = \boldsymbol{s}_t$ and compute LRC directly between the predicted and ground-truth state trajectories.

State decoding (probing). To measure which physical quantities are encoded in each model's latent representation, we freeze the world model and train a separate nonlinear probe to predict each state coordinate $s_t^{(j)}$ from $\boldsymbol{z}_t$. Each probe is a two-hidden-layer MLP with widths 128 and 64. We split complete episodes into 80% for training and 20%. On the held-out episodes, we report MAE in physical units and as a percentage of the MAE of a shared shuffled-target control. Appendix D provides full details.

Reward evaluation. We report the sum of the `dm_control` rewards over 500-step episodes, using the learned controller for TD-MPC2 and DreamerV3 and post-hoc MPC for LeWM. Appendix C.3 provides the controller and evaluation details.

4.3 Controlled interventions

To isolate specific failure modes, we run two follow-up experiments on the CartPole environment. *Nonlinear mixing*: ground-truth state is passed through 0–3 fixed, invertible mixing layers before encoding; the simulator dynamics are unchanged. *Velocity ablation*: separate pixel models are trained with ground-truth velocities fused into the visual representation ($+v$); as a control, temporal context is reduced from three frames to one for TD-MPC2 and LeWM. Implementation details for both interventions are provided in Appendix E.

5 Results

5.1 Task performance does not imply accurate latent prediction

Task reward is the metric on which current latent world models are most often evaluated. We compare episode reward with latent rollout consistency (LRC, see Eq. 6), which measures how closely an open-loop latent rollout follows the representations inferred from the corresponding observed trajectory.

Reward and LRC capture distinct aspects of model performance: Models can achieve high rewards while producing inaccurate latent rollouts, even just a few steps into the future (Fig. 4a). This dissociation is most apparent for the task-trained models, TD-MPC2 and DreamerV3, whose controllers can

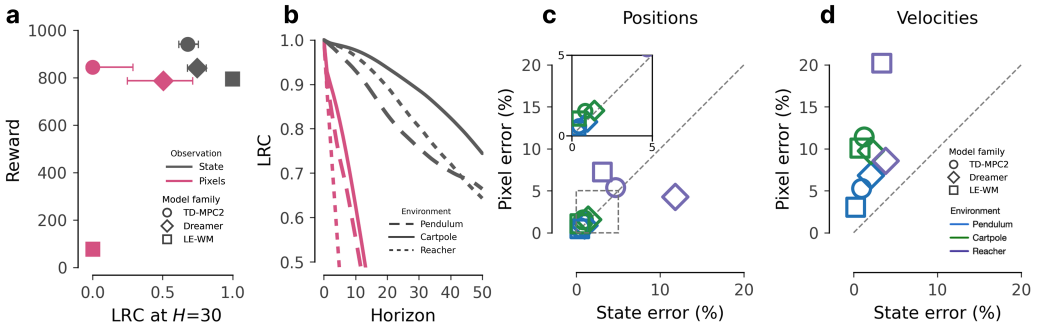


Figure 4: **Latent rollout consistency (LRC) vs. Reward.** (a) LRC vs. reward averaged across environments. The 95% CI is estimated by bootstrap over $n = 4$ model seeds. (b) LRC over rollout horizons. (c) Position-decoding errors for state- and pixel-based models across environments, normalized by a shuffled-target baseline. (d) Velocity-decoding errors, shown as in (c).

solve the task despite having a very low LRC. For each model family, we consider the model variant trained on pixels against a topline that receives raw states, including velocities, as the input. Appendix Figure 12 reports the corresponding results separately for each environment. For task-trained models, this dissociation might arise because their objectives permit task-specific shortcuts. Alternatively, predictive information might be encoded implicitly in policy and value networks, allowing these components to compensate for inaccuracies in the explicit latent dynamics model[43]. In contrast, JEPA-based models with a distinct pretraining phase followed by model-predictive control cannot exhibit this shortcut by design. As a consequence, sufficiently high LRC is required to obtain reward, and on the tested environments the quality of the rollouts is not sufficient to meet this bar. These results show that evaluating latent world models solely through task reward provides an incomplete picture and may obscure whether they learn accurate internal dynamics.

5.2 Pixel observations degrade latent prediction

Having shown that task reward can obscure poor latent prediction, we next ask where these predictive deficiencies come from. We compare models trained from ground-truth simulator states with models trained from pixels. State inputs directly provide the physical variables needed for prediction. Pixel inputs, by contrast, require the model to infer those variables from a high-dimensional nonlinear observation and from temporal context. This comparison helps to separate limitations in learning the dynamics from limitations in constructing a predictive representation.

In Figure 4b, we report LRC over increasing rollout horizons, averaged across models for each environment. We include a supervised state-space MLP as a topline and persistence as a naive baseline (Section 4.2). Although prediction quality degrades in both modalities, pixel-based rollouts deteriorate faster and approach the persistence baseline. The same pixel–state gap is visible within individual model families (Appendix Figure 11).

Figure 5b illustrates this behavior through CartPole rollouts decoded using the nonlinear probes described in Section 4.2. Across all three model families, predictions from pixels depart from the realized trajectory even over the relatively short horizon of 30 control steps, corresponding to only 0.6 seconds of simulated time ($\Delta t = 0.02$ s).

As formalized by the faithful condition in Section 2 (Eq. 1), accurate latent prediction requires both an encoder that retains the predictive state information and a latent dynamics model that evolves that information correctly. Low LRC can therefore arise from a failure of either component, or both.

We first examine what the encoder captures by decoding individual state coordinates from frozen latents using the nonlinear probes described in Section 4.2. Figures 4c and 4d compare normalized decoding errors for positions (linear positions and angles) and velocities (linear and angular velocities), respectively. Positions are decoded with low error (0–7%) in most conditions, with relatively small differences between state and pixel inputs. Velocities are decoded less accurately (3–22%), and state models consistently outperform pixel models by a large margin. Appendix Figures 17 and 18 report the per-environment errors in physical units and compare linear with nonlinear probes, respectively.

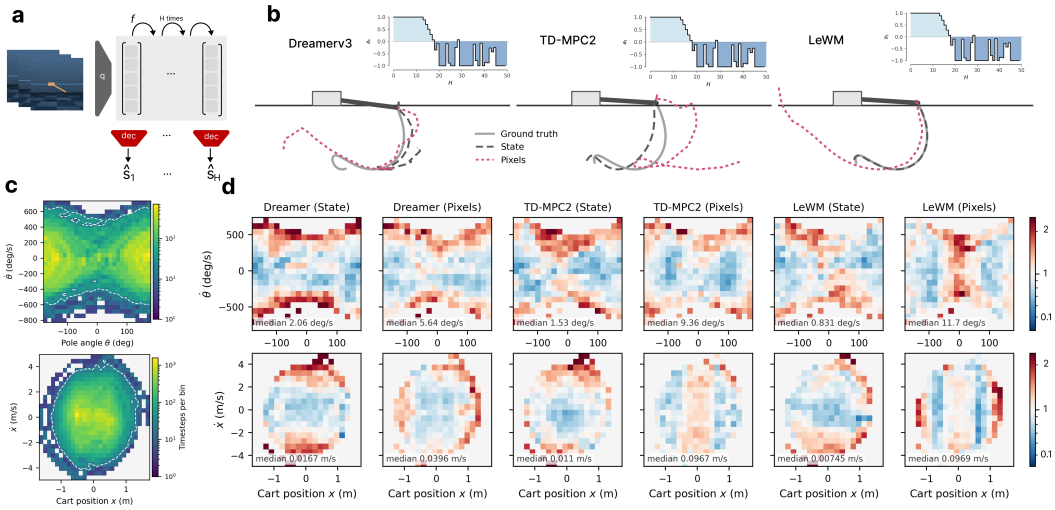


Figure 5: **CartPole rollouts and state-dependent prediction errors.** (a) Predicted latent rollouts are mapped to physical state using the frozen nonlinear probes. (b) Example pole-angle trajectories for DreamerV3, TD-MPC2, and LeWM, with the recorded actions shown above each trajectory. (c) Evaluation-set occupancy over pole angle and angular velocity (top) and cart position and velocity (bottom). (d) Prediction errors over the same state spaces for models trained from states or pixels. Color denotes the error within each bin relative to the median prediction error, and the annotation reports the median error in physical units.

Figure 5d shows angular-velocity errors (top row) and cart-velocity errors (bottom row) across the CartPole state space. These errors are not uniformly distributed: they increase with large velocities, where the system evolves most rapidly and estimation errors can compound over autoregressive rollouts. This pattern appears for both input modalities, but the median error is roughly 2–15$\times$ larger for pixel-based models, depending on the model and velocity variable.

5.3 Missing velocities explain part of the pixel–state gap

This asymmetry likely arises because the underlying physical state is Markovian, whereas a single pixel observation is not: velocities must instead be inferred from temporal context. We test this hypothesis on CartPole by providing pixel-based models with ground-truth cart and pole velocities and comparing LRC, task reward, and velocity decoding error (Figure 6). During evaluation, velocity is provided only at initialization; the latent dynamics must evolve the state thereafter. As a negative control, we also reduce the temporal context of TD-MPC2 and LeWM from three frames to a single frame. Full implementation details are provided in Appendix E.2.

Adding velocity substantially improves TD-MPC2 and LeWM, both with reduced and full temporal context. However, it has no effect on DreamerV3. This could be partly because the velocity is available to its RSSM posterior but not its prior, potentially creating a mismatch.

The decoding results (Table 1, $\epsilon_{\dot{q}}$) confirm that the intervention has the intended effect on the representation. Without velocity injection, reducing the temporal context to one frame raises normalized velocity error to approximately 130% for both TD-MPC2 and LeWM. Providing velocity reduces this error to approximately 1–2% (Table 1).

Regarding task reward, improved velocity representations do not consistently increase reward. TD-MPC2 changes little, LeWM remains far below its state-input performance despite substantially improving LRC, and DreamerV3 performs worse. CartPole swing-up requires coordinated actions over a longer timescale than 30 steps, which corresponds to only 0.3 seconds. LeWM's improved LRC at $H = 30$ is therefore insufficient for planning the complete swing-up motion, consistent with its limited reward improvement.

Overall, while adding velocity explains part of the LRC gap between pixel and state inputs, it does not fully account for the remaining difference in rollout accuracy.

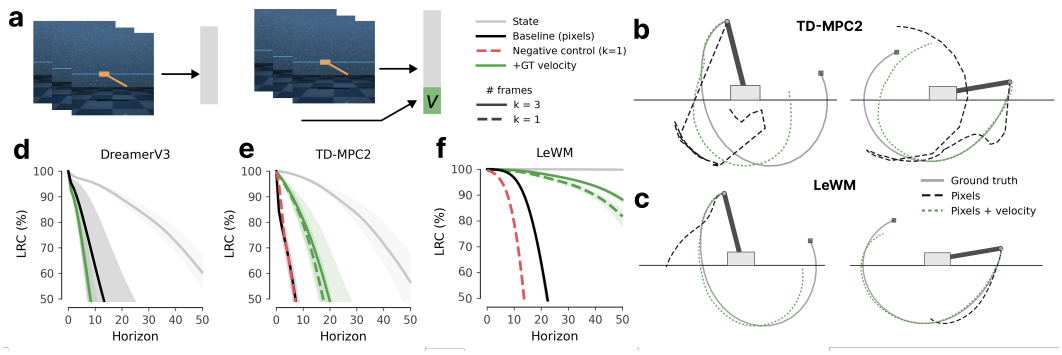


Figure 6: **Velocity observability and world-model prediction.** (a) Ground-truth velocity is fused with the visual representation. (b–c) Example CartPole rollouts with and without velocity for TD-MPC2 and LeWM. (d–f) LRC over the rollout horizon for DreamerV3, TD-MPC2, and LeWM. Full results are reported in Table 1.

(a) DreamerV3

Input	LRC_{10}	LRC_{30}	R	$\epsilon_{\dot{q}}$
state	**95.2**	81.9	**871**	2.28
frames	63.8	8.9	**872**	9.51
+v	37.9	< 0	**774**	8.06

(b) TD-MPC2

Input	LRC_{10}	LRC_{30}	R	$\epsilon_{\dot{q}}$
state	**97.4**	83.4	**878**	1.18
3 frames	29.6	< 0	**832**	11.45
+v	80.8	16.4	**855**	1.90
1 frame	27.7	< 0	206	130.61
+v	79.5	5.0	**865**	1.81

(c) LeWM

Input	LRC_{10}	LRC_{30}	R	$\epsilon_{\dot{q}}$
state	**100.0**	**99.9**	**757**	0.73
3 frames	**96.7**	1.2	151	10.16
+v	**99.6**	96.3	178	1.29
1 frame	79.4	< 0	171	130.39
+v	**99.3**	94.4	195	1.28

Table 1: LRC at $H = 10$ and $H = 30$, reward (R), and normalized velocity error ($\epsilon_{\dot{q}}$); $+v$ add ground-truth velocity. LRC and errors are in percentage units. Bold: $R > 700$ or $\text{LRC}_{10} > 90$.

5.4 Learning latent dynamics from nonlinear observations

Missing velocity explains only part of the pixel–state gap. Another key difference is that *pixels are a nonlinear transformation of the underlying physical state*. On CartPole, we replace rendering with an invertible nonlinear observation map whose complexity we can control. We then ask whether models can still learn faithful latent dynamics when all state information is preserved but represented in increasingly nonlinear coordinates. We present results for LeWM in Figure 7 because its self-supervised objective most directly approximates the faithful condition in Eq. 1. This allows us to study the effect of nonlinear observations with fewer task-specific confounds. Appendix Figure 19 reports the same intervention for TD-MPC2 and DreamerV3 alongside LeWM. For each world model, we pass the ground-truth state vector through a bijective mixing function[39] with 0–3 layers before the encoder (Figure 7a–b). Because the mixing function is invertible, *the mixed state vector contains the same information as the raw state, only in a harder-to-invert coordinate system*. We include the standard pixel condition as a reference point on the same plots.

Task reward and LRC jointly decline as the number of mixing layers increases (Figure 7c), and the controller solves only the zero-layer (raw-state) and one-layer conditions. We next examine representation quality in these conditions. The pixel model has a normalized position-decoding error slightly above 1% and does not solve the task. With one mixing layer, the error is approximately 0.5% and the model solves the task; with two layers, the error exceeds 1.5% and the task fails (Figure 7c–d). Velocity encoding may also contribute to this behavior. The pixel-plus-velocity condition from Figure 6 has a velocity-decoding error comparable to that obtained with two mixing layers (Figure 7e). Qualitative rollouts show that errors arise early and severely degrade rollout quality for models with more than one mixing layer. These failures cannot be attributed to missing information in the model input. In this controlled setting, the evaluated objectives struggle to learn representations from which the physical state is recoverable, even though the observation map is invertible.

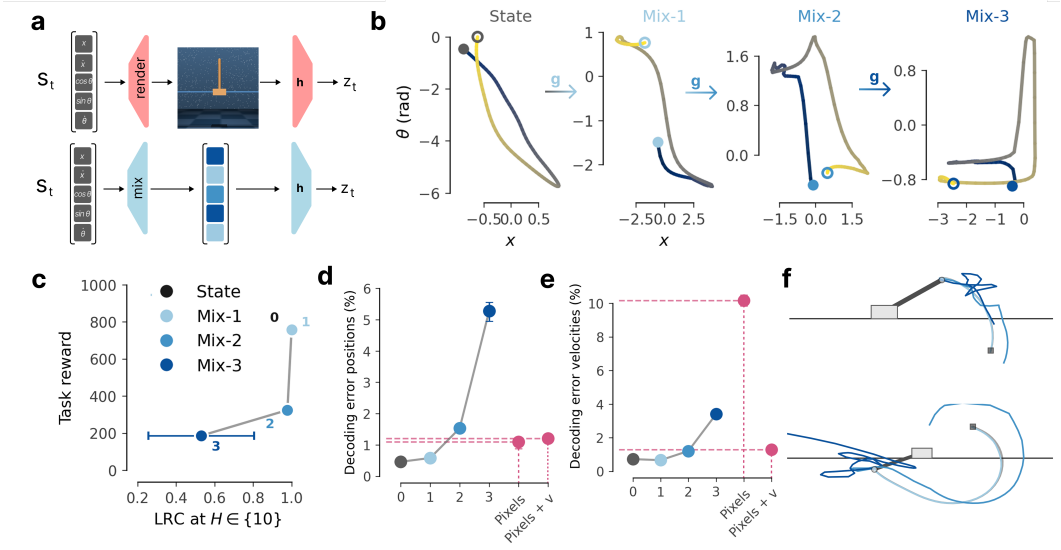


Figure 7: **World modeling under nonlinear observation mixing on CartPole.** (a–b) Pixel rendering is contrasted with approximately invertible mixing of the underlying state. (c–f) LRC at $H = 10$, task reward, normalized position error, and normalized velocity error for LeWM versus mixing depth, with pixels shown as a reference.

6 Discussion and Limitations

Our experiments show that task performance alone provides limited information about the dynamics learned by a latent world model. Using LRC and state decoding, we point to two sources of difficulty in learning faithful dynamics. First, pixel-based models struggle to recover velocity. Providing velocity explicitly improves latent prediction but does not fully close the gap to state-based models. Second, even when the underlying state is fully observed, nonlinear transformations make both the representation and its dynamics substantially harder to learn.

Our work deliberately focuses on simple deterministic continuous-control environments. This makes the physical state and transition dynamics observable and allows direct evaluation of faithful latent dynamics, but leaves open how these results extend to stochastic, partially observable, discrete, or substantially higher-dimensional environments. Similarly, we evaluate three representative latent world-model families rather than attempting an exhaustive comparison of existing approaches. *Our results should be interpreted as controlled counterexamples to the assumption that successful control implies an accurate model of the underlying environment dynamics, rather than as evidence that all latent world models exhibit the same failure modes.*

7 Conclusion

We asked whether latent world models that successfully solve control tasks actually learn representations and dynamics that model their environment. To make this question testable, we defined *faithful latent world modeling* through an equivariance condition and introduced latent rollout consistency (LRC) together with state decoding to measure these properties independently of task reward. Across three latent world models and controlled continuous-control environments, high reward did not imply faithful latent dynamics, particularly when learning from pixels.

Our results further show that this gap cannot be attributed to a single source of failure: recovering the dynamical state from observations is difficult, but even with the relevant state information available, nonlinear observations can make the dynamics substantially harder to learn. These results suggest that improving world models requires addressing representation and dynamics learning jointly, and motivate closing the gap between theory of identifiable representation learning and numerically successful implementations of these algorithms.

AI USE STATEMENT

We used generative AI tools to assist with drafting and editing the manuscript, implementing and running experimental code, and creating plotting code for the figures. All AI-assisted text and code were reviewed by the authors. Experimental outputs and figures were checked for correctness, and the authors take full responsibility for the final content of this work.

REFERENCES

[1] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc., 2018. URL https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution. https://worldmodels.github.io.

[2] Yann LeCun. A path towards autonomous machine intelligence. OpenReview, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf. Version 0.9.2, 27 June 2022.

[3] Jonathan Richens, Tom Everitt, and David Abel. General agents need world models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 51659–51687. PMLR, 2025. URL https://proceedings.mlr.press/v267/richens25a.html.

[4] Sonia Joseph, Quentin Garrido, Randall Balestriero, Matthew Kowal, Thomas Fel, Shahab Bakhtiari, Blake Aaron Richards, and Michael Rabbat. Interpreting physics in video world models. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 54687–54724. PMLR, 2026. URL https://proceedings.mlr.press/v306/joseph26a.html.

[5] Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models understand physical principles? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 948–958, 2026.

[6] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

[7] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.

[8] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026. URL https://arxiv.org/abs/2603.19312.

[9] Aapo Hyvärinen, Ilyes Khemakhem, and Ricardo Monti. Identifiability of latent-variable and structural-equation models: from linear to nonlinear. *Annals of the Institute of Statistical Mathematics*, 76(1):1–33, 2024. doi: 10.1007/s10463-023-00884-4. URL https://doi.org/10.1007/s10463-023-00884-4.

[10] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35: 8633–8646, 2022.

[11] John O'keefe and Lynn Nadel. *The hippocampus as a cognitive map*. Oxford university press, 1978.

[12] Balaraman Ravindran and Andrew G. Barto. Model minimization in hierarchical reinforcement learning. In *Abstraction, Reformulation, and Approximation*, volume 2371 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2002. doi: 10.1007/3-540-45622-8_15.

[13] Paul K Rubenstein, Sebastian Weichwald, Stephan Bongers, Joris M Mooij, Dominik Janzing, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Causal consistency of structural equation models. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, pages 151–160. PMLR, 2017.

[14] Sander Beckers, Frederick Eberhardt, and Joseph Y Halpern. Approximate causal abstractions. In *Uncertainty in artificial intelligence*, pages 606–615. PMLR, 2020.

[15] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

[16] Rodrigo González Laiz, Tobias Schmidt, and Steffen Schneider. Self-supervised contrastive learning performs non-linear system identification. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ONfWFluZBI.

[17] Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

[18] Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991.

[19] Zheng Zhu, Xiaofeng Wang, Wangbo Zhao, Chen Min, Bohan Li, Nianchen Deng, Min Dou, Yuqi Wang, Botian Shi, Kai Wang, Chi Zhang, Yang You, Zhaoxiang Zhang, Dawei Zhao, Liang Xiao, Jian Zhao, Jiwen Lu, and Guan Huang. Is Sora a world simulator? a comprehensive survey on general world models and beyond, 2024. URL https://arxiv.org/abs/2405.03520.

[20] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

[21] Gaoyue Zhou, Hengkai Pan, Yann Lecun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 79115–79135. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

[22] Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems*, volume 38, pages 49126–49162, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3e7cf447f21cd11c846463affefce665-Abstract-Conference.html.

[23] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

[24] Quentin Garrido, Nicolas Ballas, Mahmoud Assran, Adrien Bardes, Laurent Najman, Michael Rabbat, Emmanuel Dupoux, and Yann LeCun. Intuitive physics understanding emerges from self-supervised pretraining on natural videos. *arXiv preprint arXiv:2502.11831*, 2025. URL https://arxiv.org/abs/2502.11831.

[25] Lance Ying, Katherine M Collins, Prafull Sharma, Cedric Colas, Kaiya Ivy Zhao, Adrian Weller, Zenna Tavares, Phillip Isola, Samuel J Gershman, Jacob D Andreas, et al. Assessing adaptive world models in machines with novel games. *arXiv preprint arXiv:2507.12821*, 2025.

[26] Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

[27] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *ICLR*, 2023.

[28] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 16–30. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.blackboxnlp-1.2/.

[29] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding dqns. In *International conference on machine learning*, pages 1899–1908. PMLR, 2016.

[30] Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47):e2206625119, 2022.

[31] Thomas Bush, Stephen Chung, Usman Anwar, Adrià Garriga-Alonso, and David Krueger. Interpreting emergent planning in model-free reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/cc4d9cfc45325e460b455a820d5f212c-Abstract-Conference.html.

[32] Mohammad Taufeeque, Aaron David Tucker, Adam Gleave, and Adrià Garriga-Alonso. Path channels and plan extension kernels: a mechanistic description of planning in a sokoban RNN. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=aAshH4kQ1v.

[33] Riley Simmons-Edler, Ryan P Badman, Felix Baastad Berg, Raymond Chua, John J Vastola, Joshua Lunger, William Qian, and Kanaka Rajan. Deep RL needs deep behavior analysis: Exploring implicit planning by model-free agents in open-ended environments. In *Advances in Neural Information Processing Systems*, volume 38, pages 81066–81113, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/748ba247386936692f0f9a05565c926b-Paper-Conference.pdf.

[34] Jan Sobotka, Auke Ijspeert, and Guillaume Bellegarda. Reverse-engineering memory in dreamerv3: From sparse representations to functional circuits. In *Mechanistic Interpretability Workshop at NeurIPS 2025*.

[35] Keyon Vafa, Peter G. Chang, Ashesh Rambachan, and Sendhil Mullainathan. What has a foundation model found? using inductive bias to probe for world models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 60727–60747. PMLR, 2025. URL https://proceedings.mlr.press/v267/vafa25a.html.

[36] Ziming Liu, Sophia Sanborn, Surya Ganguli, and Andreas Tolias. From kepler to newton: Inductive biases guide learned world models in transformers. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/62200.

[37] Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12979–12990. PMLR, 2021. URL https://proceedings.mlr.press/v139/zimmermann21a.html.

[38] Fabian A. Mikulasch and Friedemann Zenke. Understanding self-supervised learning via latent distribution matching. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=schXkGSTus.

[39] Aapo Hyvarinen, Hiroaki Sasaki, and Richard Turner. Nonlinear ica using auxiliary variables and generalized contrastive learning. In *The 22nd international conference on artificial intelligence and statistics*, pages 859–868. PMLR, 2019.

[40] Patrik Reizinger, Randall Balestriero, David Klindt, and Wieland Brendel. Position: An empirically grounded identifiability theory will accelerate self supervised learning research. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 82072–82090. PMLR, 2025. URL https://proceedings.mlr.press/v267/reizinger25a.html.

[41] 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. ISSN 2665-9638. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

[42] Ingmar Posner, Anson Lei, and Bernhard Schölkopf. From observation to insight: Mechanistic world models and the quest for autonomous discovery. *arXiv preprint arXiv:2607.12474*, 2026.

[43] Alistair Letcher, Mattie Fellows, Alexander D Goldie, Jonathan Richens, Jakob N Foerster, and Oliver Richardson. Inverting the bellman equation: From q-values to world models. *arXiv preprint arXiv:2606.21173*, 2026.

A Environments and evaluation datasets

Environments. We evaluate on three DeepMind Control Suite[41] environments: *pendulum*, *cartpole*, and *reacher*. Each exposes a low-dimensional simulator state $\boldsymbol{s}_t$ that we use as the probing target, and image observations rendered at 64×64 and 224×224 for pixel-based world models. Pendulum (pendulum-swingup) has a 1-dimensional torque action and a 3-dimensional state $(\cos\theta, \sin\theta, \dot{\theta})$. CartPole (cartpole-swingup) has a 1-dimensional force action and a 5-dimensional state $(x, \cos\theta, \sin\theta, \dot{x}, \dot{\theta})$ following the standard DM-Control parameterization. Reacher (reacher-hard) has a 2-dimensional torque action and a 6-dimensional state consisting of two joint angles, two joint velocities, and the 2D Cartesian target offset. Table 2 summarizes all three environments.

Table 2: Evaluation environments and the underlying physical state used as probing targets.

Env.	d_s	d_a	**State variables**	**Task**
Pendulum (pendulum-swingup)	3	1	$\cos\theta$, $\sin\theta$, pole angular velocity	Swing the pole upright from below and balance it.
CartPole (cartpole-swingup)	5	1	cart position, cos/sin of pole angle, cart and pole velocities	Swing the pole upright from below and balance it by moving the cart.
Reacher (reacher-hard)	6	2	two joint angles, two joint velocities, 2D target offset	Bring the fingertip of a two-link arm to a small randomized target.

Evaluation data. For each environment, we collect 1,000 episodes of 100 steps each. Each episode stores the simulator state $\boldsymbol{s}_t$, the action $\boldsymbol{a}_t$, the reward, and rendered 64×64 and 224×224 RGB frames.

Actions are drawn from an Ornstein–Uhlenbeck (OU) process with mean reversion $\theta = 0.15$ and base diffusion $\sigma = 0.35$, modulated by intermittent high-energy bursts to encourage broad coverage of joint angles and velocities. The OU schedule is identical across environments except that, for CartPole, a higher fraction of episodes uses the high-energy regime (0.75 vs. 0.4 for Pendulum and Reacher) and the cart position is softly constrained to $[-0.85, 0.85]$ via a position-keeper that becomes active near the rails. Initial states are sampled from broad priors over joint positions and velocities (uniform over angles, near-zero velocities), so that the resulting trajectories visit a wide range of physical states without using a trained controller. Figures 8–10 compare the marginal distributions of every state variable, action, and native `dm_control` reward under the random OU and expert policies.

B Latent rollout consistency computation

For each model seed and environment, we initialize one autoregressive latent rollout at the beginning of every evaluation episode. At rollout horizon H, let $z_{e,H}^{(j)}$ denote coordinate j of the target latent representation obtained by encoding the realized observation history in episode e, and let $\hat{z}_{e,H}^{(j)}$ denote the corresponding coordinate predicted by the latent dynamics model. We pool episodes within the given model-seed and environment condition, but do not pool representations across model seeds or environments.

For each latent coordinate j, we compute

$$R_j^2(H) = 1 - \frac{\sum_{e=1}^{N}\left(z_{e,H}^{(j)} - \hat{z}_{e,H}^{(j)}\right)^2}{\sum_{e=1}^{N}\left(z_{e,H}^{(j)} - \bar{z}_H^{(j)}\right)^2}, \qquad \bar{z}_H^{(j)} = \frac{1}{N}\sum_{e=1}^{N} z_{e,H}^{(j)}, \tag{7}$$

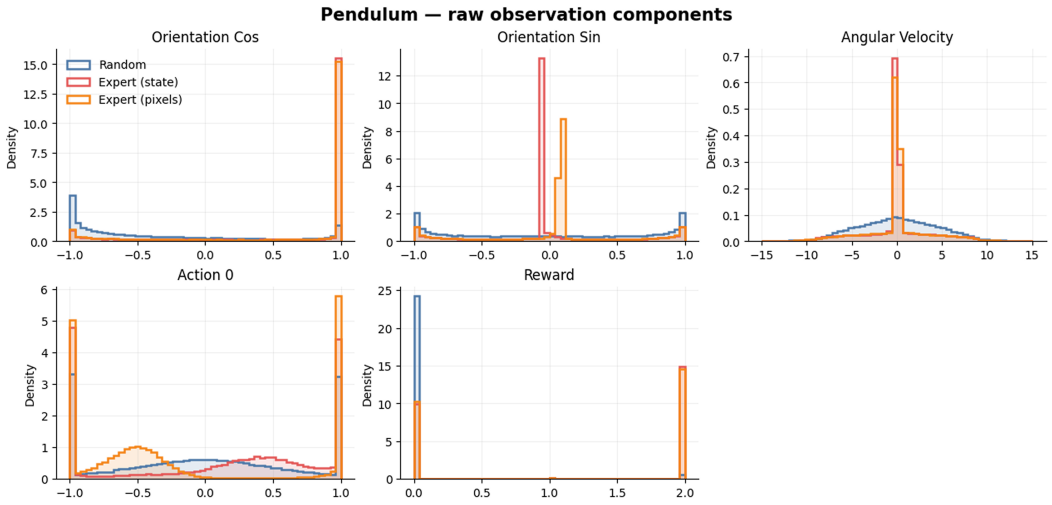


Figure 8: **Pendulum data distributions.** Marginal distributions of the orientation coordinates, angular velocity, action, and native `dm_control` reward in the random OU and expert datasets.

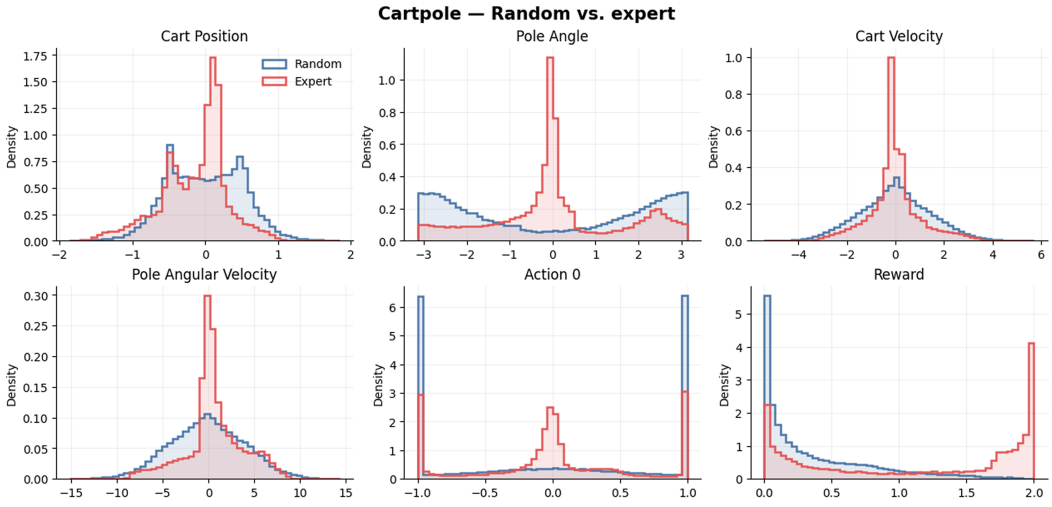


Figure 9: **CartPole data distributions.** Marginal distributions of cart position, pole angle, cart and pole velocities, action, and native `dm_control` reward in the random OU and expert datasets.

where N is the number of evaluation episodes. We then aggregate across the d_z latent coordinates using their target variance:

$$\mathrm{LRC}(H) = \frac{\sum_{j=1}^{d_z} V_j(H) R_j^2(H)}{\sum_{j=1}^{d_z} V_j(H)}, \qquad V_j(H) = \sum_{e=1}^{N} \left(z_{e,H}^{(j)} - \bar{z}_H^{(j)} \right)^2. \tag{8}$$

Coordinates with zero target variance have zero weight and are omitted. Negative coordinate-wise and aggregate R^2 values are retained rather than clipped. Consequently, $\mathrm{LRC}(H) = 1$ indicates exact latent prediction, $\mathrm{LRC}(H) = 0$ matches prediction by the across-episode target mean, and negative values indicate performance worse than that reference.

DreamerV3 stochastic rollouts. Because DreamerV3 uses a stochastic RSSM, we infer the filtered posterior once, copy this same anchor across 500 rollouts, replay the recorded actions, and independently sample the categorical prior state at every future step. We represent each predicted state by the deterministic RSSM state concatenated with the categorical probabilities, average these feature vectors coordinate-wise across the 500 trajectories at every horizon, and use this Monte Carlo mean as $\hat{\boldsymbol{z}}_{e,H}$ above. At each horizon, we therefore average the 500 predicted latent vectors and compute LRC between this mean prediction and the latent representation inferred from the corresponding observed trajectory. Figure 13 compares this estimator with the deterministic RSSM rollout and an oracle that selects the best-matching stochastic rollout from the same samples.

C World-model implementation, training, and evaluation

C.1 Models and implementation sources

We evaluate three complementary world-model families: TD-MPC2[7], DreamerV3[6], and LeWM[8]. We use the authors' implementations and default configurations as the starting point for all three models, with the modifications described below. For every model family, observation modality, and environment, we train four independent seeds. Table 3 summarizes the resulting experimental matrix.

Table 3: World models evaluated in this work. Latent dimension refers to the size of the activation vector used for all downstream analyses.

Model	Obs. type	Seeds	d_z	Notes
TD-MPC2	state	4	512	Single-task, per-environment
	pixels	4	512	Single-task, per-environment
DreamerV3	state	4	1536	RSSM posterior: stoch (1024) + deter (512)
	pixels	4	5120	Same architecture, visual encoder
LeWM	state	4	192	Our state-input implementation
	pixels	4	192	Original LeWM, 224×224 frames

TD-MPC2. We use the official TD-MPC2 implementation (`https://github.com/nicklashansen/tdmpc2`) with a 512-dimensional latent and train a separate model for each environment and observation modality.

DreamerV3. We use the PyTorch implementation of DreamerV3 (`https://github.com/NM512/dreamerv3-torch`) with its standard recurrent state-space model (RSSM). The state-input model has a 1024-dimensional categorical stochastic state and a 512-dimensional deterministic state, making the total latent representation 1536 dimensional.

LeWM. For pixels, we use the official LeWM implementation (`https://github.com/lucas-maes/le-wm`) with 224×224 inputs and a three-frame context. Our reproduced Reacher checkpoints achieve an 81.5% success rate, compared with the 86% reported by the original study (Figure 15). Because the original code does not support state inputs, we implement a state-based variant with a Mish MLP encoder ($d_s \to 512 \to 192$) and an action-conditioned residual MLP predictor ($384 \to 2048 \to 192$). It operates on a single state vector and uses the same one-step objective as

pixel LeWM. For all downstream analyses, we use the TD-MPC2 encoder output, the LeWM encoder output, and the DreamerV3 RSSM posterior feature vector, concatenating its deterministic state with the softmax probabilities of its categorical stochastic state rather than sampled categories.

C.2 Training protocol

TD-MPC2 and DreamerV3. We train both task-driven models for up to one million environment steps using their original optimization settings and without per-task hyperparameter tuning. We retain checkpoints after episode reward has converged to a stable high-return regime (Figure 14, top two rows).

LeWM. For each environment, LeWM is trained on 2,000 collected episodes (200,000 transitions): 1,000 from the random OU policy and 1,000 from the TD-MPC2 expert policy. The same trajectories are used for the state and pixel variants through their aligned state vectors and rendered frames. We train all LeWM variants for 50 epochs and sweep the SigReg regularization coefficient λ[8,17], selecting the values that give the best combination of training-loss convergence. The selected weights are $\lambda = 10^{-5}$ for all state-input models, $\lambda = 0.001$ for pixel CartPole, and $\lambda = 0.1$ for pixel Pendulum and Reacher. Pendulum and CartPole use a frame skip of 1, whereas Reacher uses a frame skip of 5 following the original LeWM setup. Prediction and SigReg losses converge within the training budget (Figure 14, bottom row).

C.3 Task-return evaluation

TD-MPC2 and DreamerV3. For the task-driven models, we evaluate the controller learned jointly with each world model. TD-MPC2 is run in deterministic evaluation mode, whereas DreamerV3 samples from its learned actor distribution. For each model seed, we report native environment return over 100 episodes of 500 environment steps.

LeWM. Because LeWM is trained without a policy or reward, we evaluate it with post-hoc model-predictive control. The original LeWM controller scores an action sequence by the distance between the goal embedding and only the final predicted latent. Figure 16 compares that latent cost with a decoded state cost across planners. The state cost passes each predicted latent through a frozen MLP trained on the aligned simulator states in the LeWM training data, then sums the weighted squared distance to the goal over the horizon. It attains higher return than the latent cost, so we use it for every environment and both observation modalities. Among the planners in Figure 16, we use the one that gives the highest reward. Returns are the sum of the environment's native rewards over 500-step episodes. For each of the four model seeds we evaluate 50 episodes with each of three planner seeds.

The state cost for each environment is:

- **Pendulum (swingup):** The goal is the stationary upright state. The cost weights the sine and cosine of the pole angle by 1 and angular velocity by 0.05.
- **CartPole (swingup):** The goal is the stationary upright pole with the cart centered on the rail. The cost assigns unit weight to cart position and to the sine and cosine of the pole angle, and zero weight to both velocities.
- **Reacher (hard):** Two joint configurations can place the fingertip on a reachable target. We solve the analytic inverse kinematics for the elbow-up and elbow-down goals and roll out toward each. The cost weights the two joint angles by 1, the two-dimensional fingertip-to-target displacement by 100, and joint velocities by 0. The return for an episode is the average native Reacher return across the two branches.

D Probing details

Data and targets. Probing is run on activation datasets extracted from the random-policy and expert-policy episodes described in Appendix A. Each sample is a full episode containing latent activations $\boldsymbol{z}_t$ and aligned ground-truth simulator state $\boldsymbol{s}_t$. For each model we train one scalar probe per state coordinate $s_t^{(j)}$, $j \in \{1, \ldots, d_s\}$.

Probe families. We use three probe configurations:

- **Linear probe**: ordinary least-squares regression from a single latent vector $\boldsymbol{z}_t$.
- **Nonlinear probe**: an MLP regressor with hidden sizes $(128, 64)$, ReLU activations, L_2 penalty $\alpha{=}10^{-4}$, learning rate 10^{-3}, early stopping (patience 20, validation fraction 0.1), and a maximum of 300 epochs.
- **Windowed nonlinear probe**: the same MLP architecture applied to a concatenated trailing window of three consecutive latent vectors $[\boldsymbol{z}_{t-2};\, \boldsymbol{z}_{t-1};\, \boldsymbol{z}_t]$. This probe is used only for LeWM with pixel inputs, whose latent representation corresponds to a short visual context window rather than a single-step state.

Inputs to all probes are standardized feature-wise (zero mean, unit variance). Targets are kept in physical units and are not standardized.

Shuffled-target control. For each environment and physical-state variable, we train a shared control probe that takes the ground-truth state as input and predicts target labels whose correspondence to the inputs has been randomly shuffled. The control uses the same nonlinear architecture and episode-level split as the primary probes. We evaluate it against the original, unshuffled targets in the held-out test episodes. This control is trained independently of any world model, and the same control error is used for every model family, observation modality, and model seed within the corresponding environment and state variable.

Train/test protocol. We split at the *episode level*: 80% of episodes are used for training and 20% for testing. This prevents leakage across temporally adjacent samples from the same rollout.

Reported metrics. We evaluate probes using the mean absolute error (MAE) in physically interpretable units: degrees for angular variables, centimeters for positions, and native units (m/s or deg/s) for velocities. For angular variables, separate probes predict $\cos\theta$ and $\sin\theta$. We reconstruct the angle as $\hat{\theta} = \mathrm{atan2}(\widehat{\sin\theta}, \widehat{\cos\theta})$ and compute its error using the shortest circular distance to the target angle. The shuffled-target control undergoes the same reconstruction and evaluation.

For model m and physical-state variable j, we define error relative to the shared shuffled-target control as

$$\epsilon_{m,j} = 100\,\frac{\mathrm{MAE}_{m,j}}{\mathrm{MAE}_{j}^{\mathrm{control}}}. \tag{9}$$

Thus, $\epsilon_{m,j} = 100\%$ denotes the error of the shuffled-target control, values below 100% indicate that the model's latent representation supports better decoding than the control, and values above 100% indicate worse performance. All metrics are computed on the held-out 20% of episodes. We report means and bootstrap confidence intervals across seeds.

E Controlled-intervention details

E.1 Nonlinear state mixing

We perform this intervention on CartPole, applying a fixed mixing network to its five-dimensional state before the encoder of each world model. Depth zero is the identity; depths one to three compose bias-free $5{\times}5$ linear maps with LeakyReLU activations (negative slope 0.2) between consecutive maps. Following the construction of Hyvarinen et al.[39], each weight matrix is sampled with column-normalized entries and retained only if its condition number lies below the 25th percentile of 10,000 random matrices. The mixer uses seed 42, remains frozen, and is shared across model seeds at a given depth. We train a separate TD-MPC2, DreamerV3, and LeWM model from scratch for every depth using the corresponding primary training protocol; the standard pixel models are included only as a reference.

E.2 Velocity observability

We perform the velocity intervention on pixel-based CartPole models using the ground-truth cart and pole velocities $(\dot{x}, \dot{\theta})$ without normalization. For each model, a small MLP maps the two velocities

to a 16-dimensional feature, which is concatenated with the visual embedding and projected back to the model's standard representation size. The resulting velocity-fused encoder, latent dynamics, and controller (for TD-MPC2 and DreamerV3) are trained jointly from scratch; velocity is supplied only while encoding observations and is not provided during subsequent open-loop latent prediction. TD-MPC2 and LeWM are trained under all four combinations of one or three input frames and with or without velocity. The DreamerV3 encoder processes one 64×64 image at each timestep, while temporal context is maintained by the recurrent RSSM. In the $+v$ condition, the ground truth two-dimensional `velocity` observation is mapped through a two-layer MLP to a 16-dimensional feature, concatenated with the CNN image embedding, and projected back to the original embedding size before entering the RSSM. We use four seeds for TD-MPC2 and DreamerV3 and three seeds for LeWM, and otherwise retain the primary training and evaluation protocols.

Figure 20 shows the same comparison for the CartPole velocity and mixing checkpoints. Both use the state cost and the highest-reward planner from Appendix C.3.

F SYNTHETIC PERTURBATIONS OF THE FAITHFUL CONDITION

Figure 2 uses an oracle CartPole planner. We corrupt either the dynamics or the encoder and measure how reward, LRC, and state decoding move. There is no learned world model.

Task and controller. We use DeepMind Control Suite CartPole swing-up[41]. The planner is random-shooting MPC: at each decision it samples $N = 50$ action sequences of length $H = 10$ over $\{-1, 0, 1\}$, simulates each sequence for $k = 5$ transition steps per horizon step, selects the lowest-cost sequence, and executes its first action in the true environment. Episodes last 500 environment steps. The planning cost is weighted mean squared error to the upright goal in the canonical state $[x, \cos\theta, \sin\theta, \dot{x}, \dot{\theta}]$, with unit weight on the position and angle coordinates and zero weight on both velocities. The internal model is the ground-truth CartPole transition T, not a learned $\hat{\mathbf{f}}$. Reported reward is the sum of the environment's native rewards over the episode.

Conditions. The three columns of Figure 2 are:

- **A.** Observations are the canonical state $\boldsymbol{s}$. The encoder is the identity. The planner uses the noisy oracle $T + \varepsilon$.
- **B.** Observations are a fixed invertible mix $\boldsymbol{y} = \mathbf{g}(\boldsymbol{s})$. The planner acts in $\boldsymbol{y}$ with conjugated dynamics $\boldsymbol{y}' = \mathbf{g}\big(T(\mathbf{g}^{-1}(\boldsymbol{y}), \boldsymbol{a}) + \varepsilon\big)$.
- **C.** Observations are the same $\boldsymbol{y}$. The encoder is a noisy inverse $\mathbf{g}^{-1} + \varepsilon$. The planner uses the clean oracle T.

Encoding. Column A uses the identity on the five-dimensional canonical state. Columns B and C use the depth-3 mixer of Appendix E.1, frozen at seed 42. The mixer is not varied in this sweep.

Noise. Both dynamics and encoder corruptions use relative Gaussian noise: we add $\varepsilon\,\sigma \odot \xi$ in canonical state units, where $\xi \sim \mathcal{N}(0, I)$ and σ is a fixed per-dimension scale. After dynamics noise we renormalize $(\cos\theta, \sin\theta)$ to the unit circle. The scale is $\varepsilon \in \{0, 0.01, \ldots, 0.10, 0.20, 0.30, 0.40, 0.50\}$ (15 values). Columns A and B inject noise in the oracle transition and leave the inverse map clean. Column C injects noise on $\mathbf{g}^{-1}$, perturbing all state components, and sets dynamics noise to zero. Only one noise channel is active in each condition.

Replication. We run three independent MPC rollouts with seeds $\{0, 1, 2\}$ per condition and ε, and average the metrics. The mixer seed remains 42. Decoder diagnostics use deterministic sample streams derived from each MPC seed for the train/eval split.

Metrics. The y-axis of Figure 2 is the cumulative environment reward of the 500-step rollout; point color is ε. The three x-axes are:

- **Linear R^2.** Ridge regression from the planning representation to the canonical state on 10,000 held-out states. Column A uses the identity, so this value is set to 1.

- **Nonlinear R^2.** The same targets, decoded with an MLP with hidden sizes $(64, 64)$.
- **LRC.** The same metric as in the main text (Eq. 6), evaluated at horizon $H = 30$.

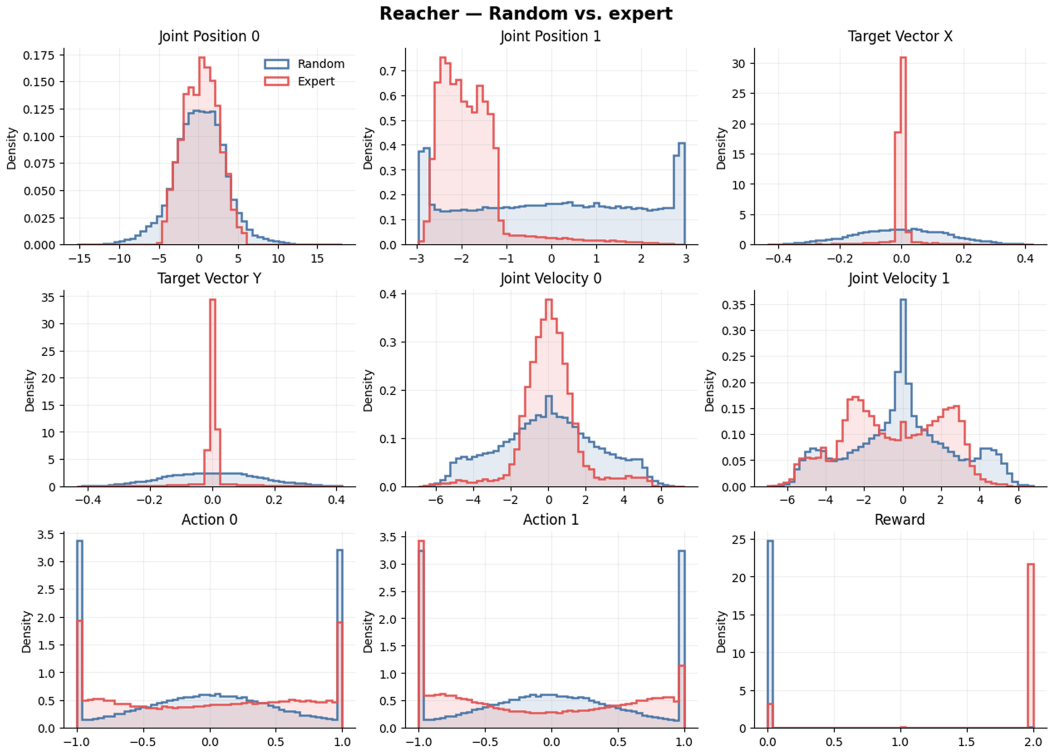


Figure 10: **Reacher data distributions.** Marginal distributions of the joint positions, target-offset coordinates, joint velocities, two action coordinates, and native `dm_control` reward in the random OU and expert datasets.

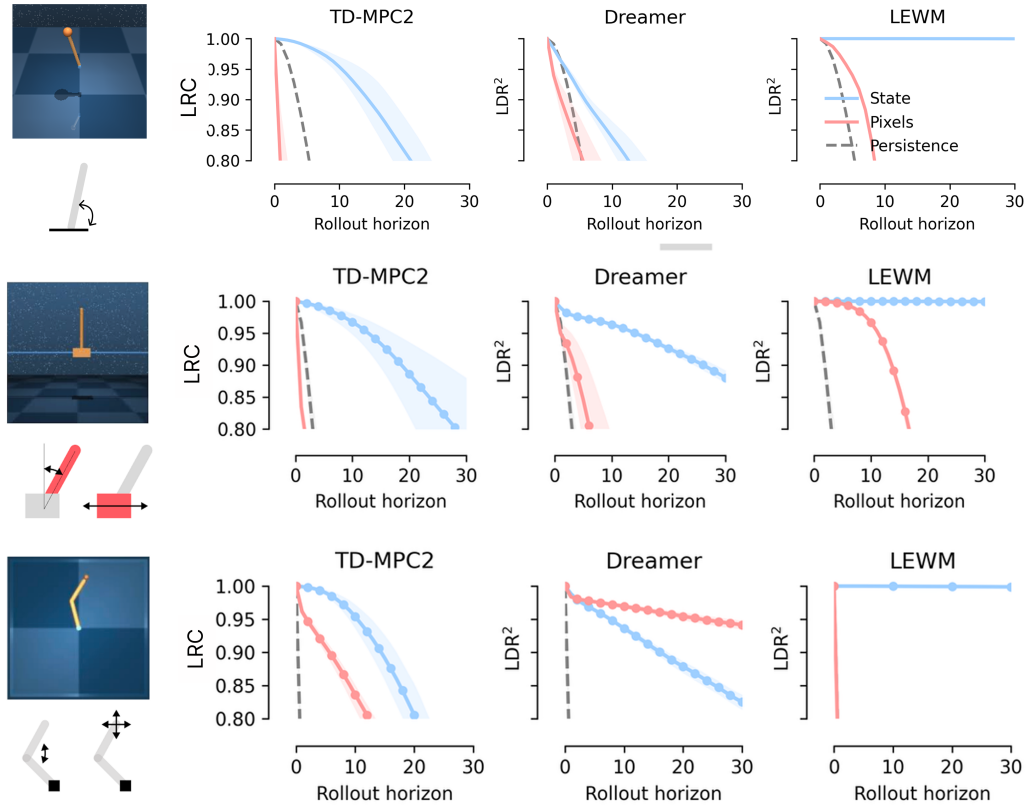


Figure 11: **Latent rollout consistency across environments and model families.** LRC over the rollout horizon for TD-MPC2, DreamerV3, and LeWM (columns) on Pendulum, CartPole, and Reacher (rows). Curves compare models trained from ground-truth states and pixels with the persistence baseline; shaded regions show variation across model seeds.

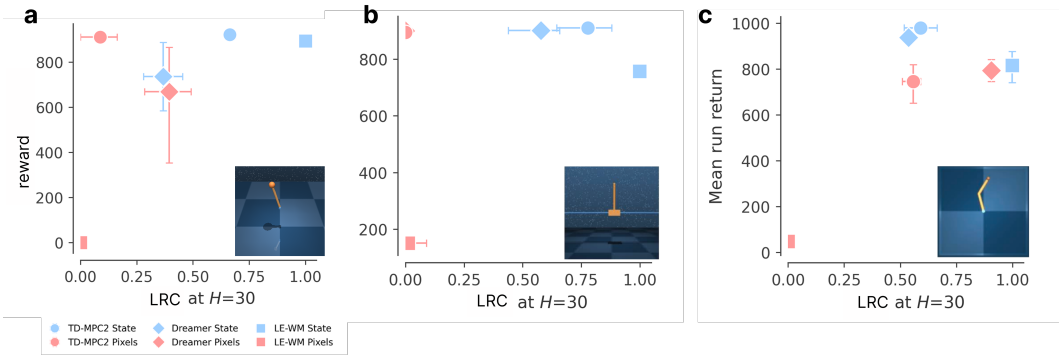


Figure 12: **Task reward and latent rollout consistency by environment.** Episode reward versus LRC at horizon $H = 30$ for (a) Pendulum, (b) CartPole, and (c) Reacher. Marker shape identifies the model family and color identifies whether the model receives ground-truth states or pixels; error bars show variation across model seeds.

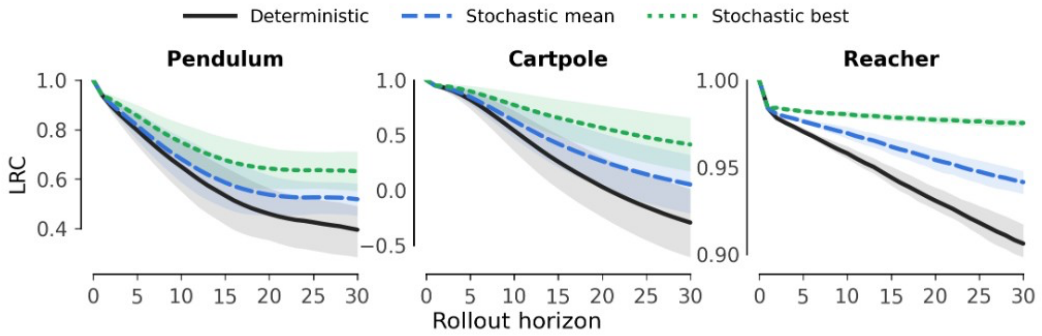


Figure 13: **Marginalizing DreamerV3's stochastic transitions.** LRC over the rollout horizon on Pendulum, CartPole, and Reacher using the deterministic RSSM rollout, the coordinate-wise mean of 500 stochastic rollouts generated from a shared posterior anchor, and the best-matching sampled rollout. The latter uses the realized trajectory to select a sample and therefore represents an oracle upper bound rather than a deployable estimator. Shaded regions show variation across model seeds.

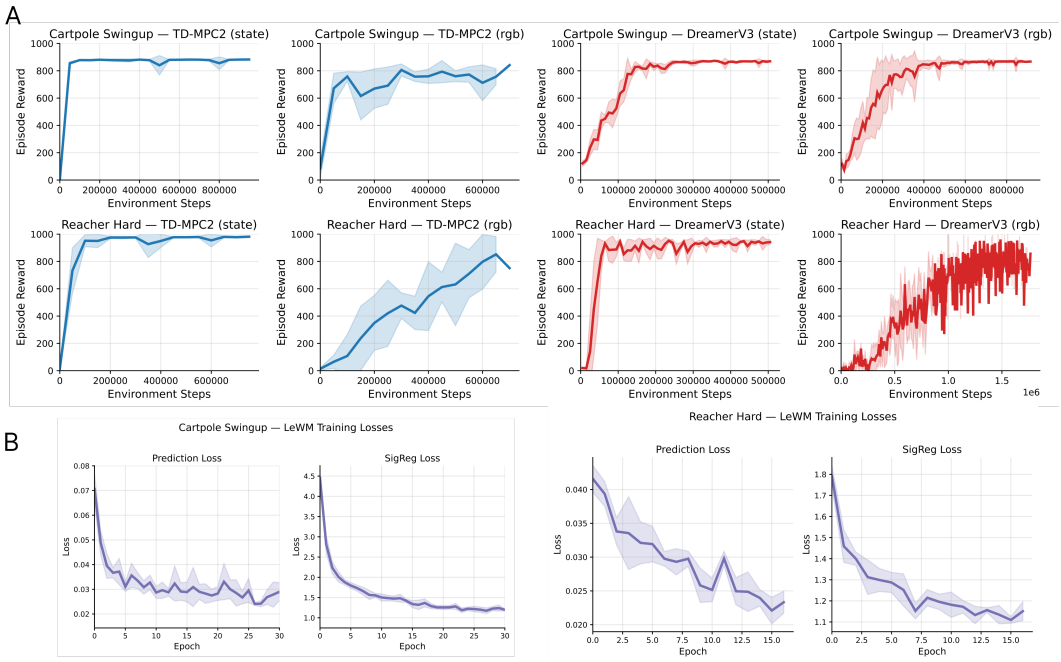


Figure 14: **Training dynamics of the evaluated world models.** The top two rows show episode reward over environment steps for TD-MPC2 and DreamerV3 trained from state and RGB observations on CartPole and Reacher. The bottom row shows LeWM's prediction and SigReg training losses over epochs on the same environments. Curves show means across model seeds and shaded regions show variation across seeds.

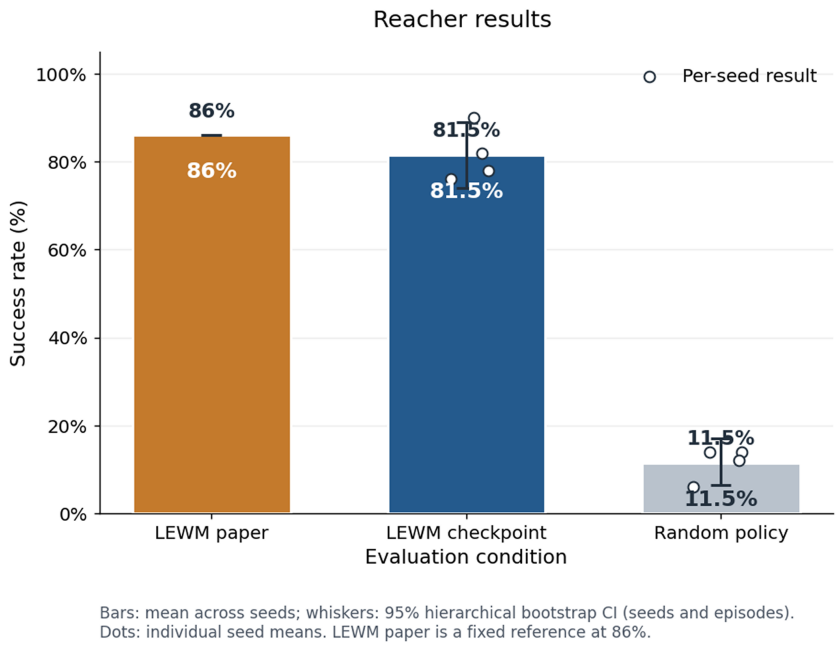


Figure 15: **Reproduction of LeWM performance on Reacher.** The original study reports an 86% success rate for pixel-based LeWM. Our reproduced checkpoints achieve 81.5%, compared with 11.5% for a random policy. Bars show means across four seeds, dots show individual seed means, and whiskers show 95% hierarchical bootstrap confidence intervals over seeds and episodes. The published result is included as a fixed reference without uncertainty.

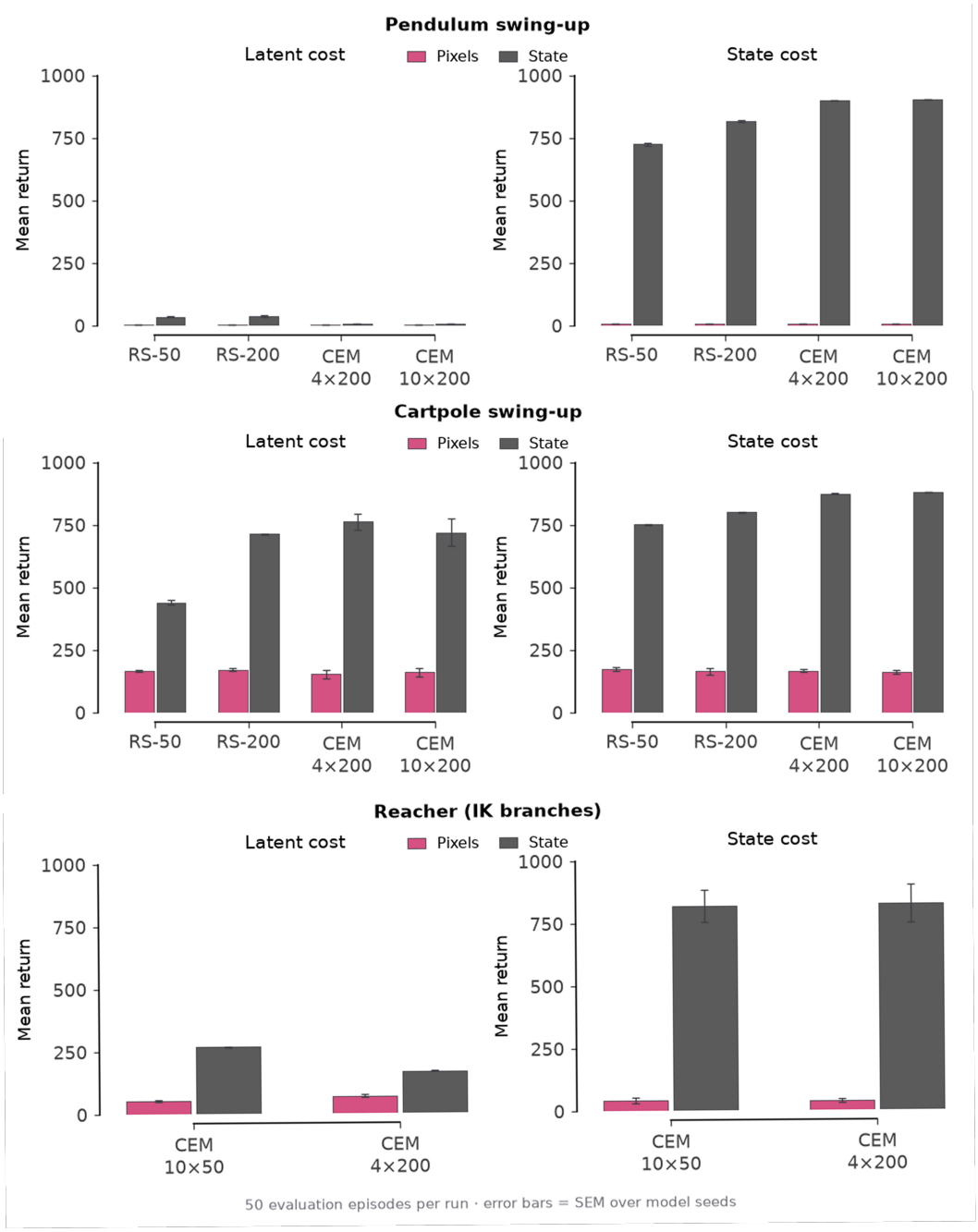


Figure 16: **LeWM cost and planner comparison.** Terminal latent cost and decoded state cost for pixel-input (pink) and state-input (gray) LeWM. Planners are random shooting with 50 or 200 candidates and CEM with 4 or 10 iterations of 200 candidates. Reacher compares two CEM settings. Reported LeWM returns use the state cost and the planner with the highest reward. Bars are means over model seeds; error bars are the standard error. Each bar uses 50 evaluation episodes.

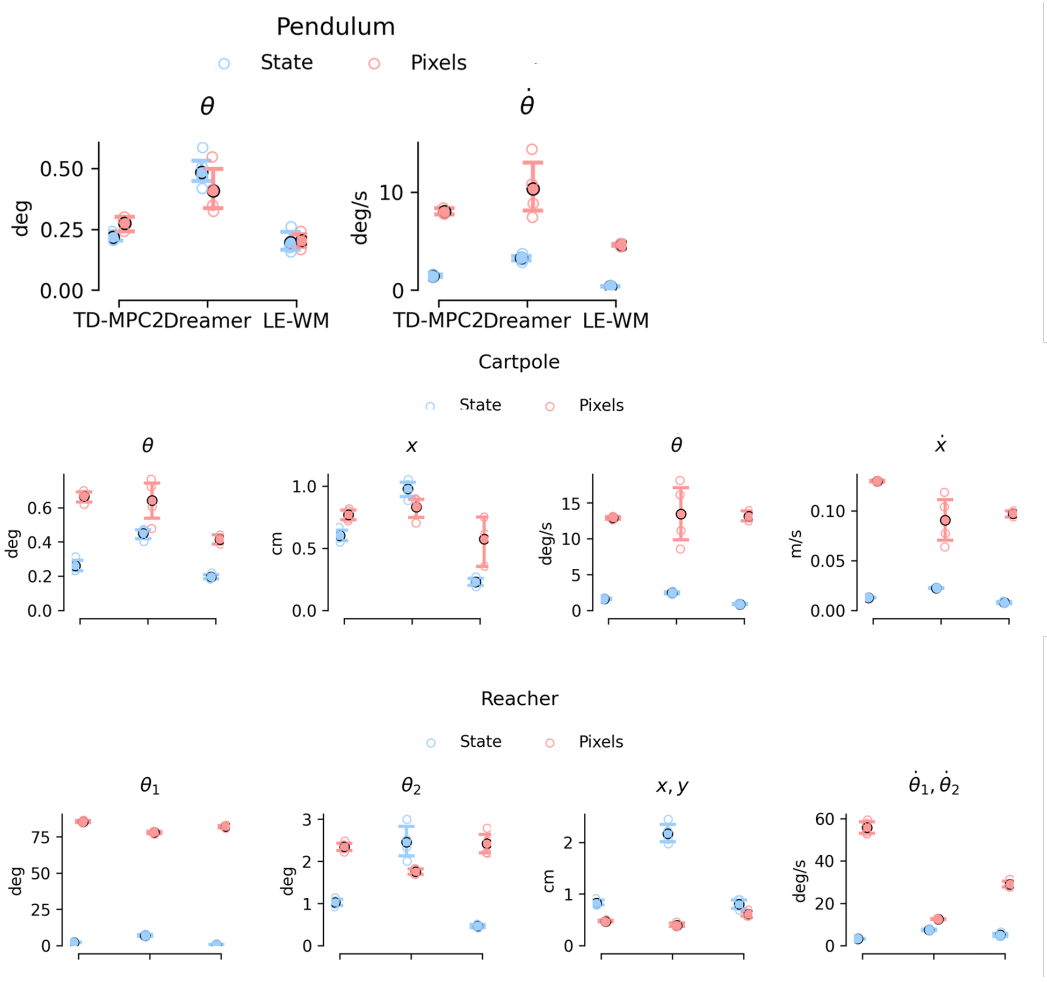


Figure 17: **State-decoding errors in physical units.** Held-out nonlinear-probe errors for models trained from ground-truth states or pixels, reported separately for Pendulum, CartPole, and Reacher. Panels separate the positional and velocity coordinates of each environment and report errors in degrees, centimeters, meters per second, or degrees per second as appropriate. Colored markers show individual model seeds; black markers and error bars summarize performance across seeds.

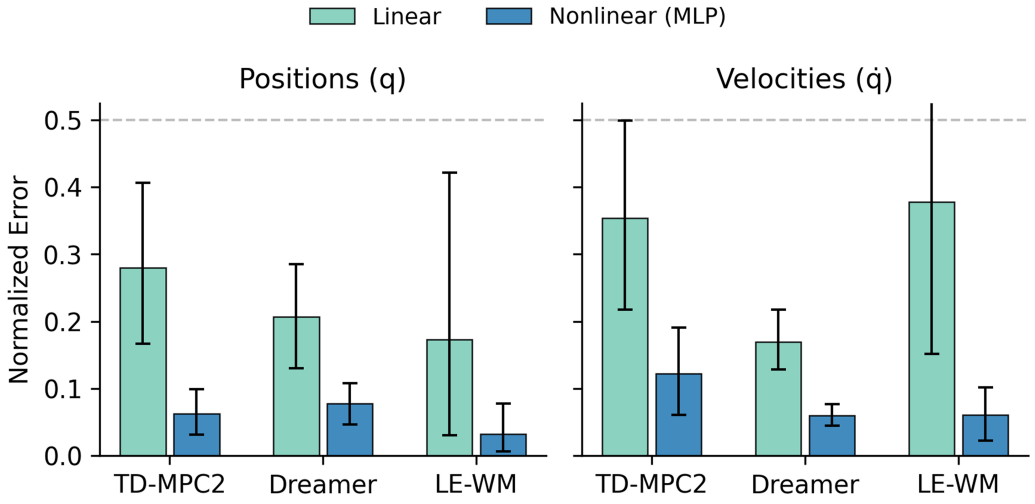


Figure 18: **Linear and nonlinear state decoding.** Normalized decoding error for positional variables (left) and velocities (right), averaged across environments for each model family. Bars compare linear probes with nonlinear MLP probes; error bars show variation across environments. The dashed line marks an error equal to 50% of the shuffled-target control.

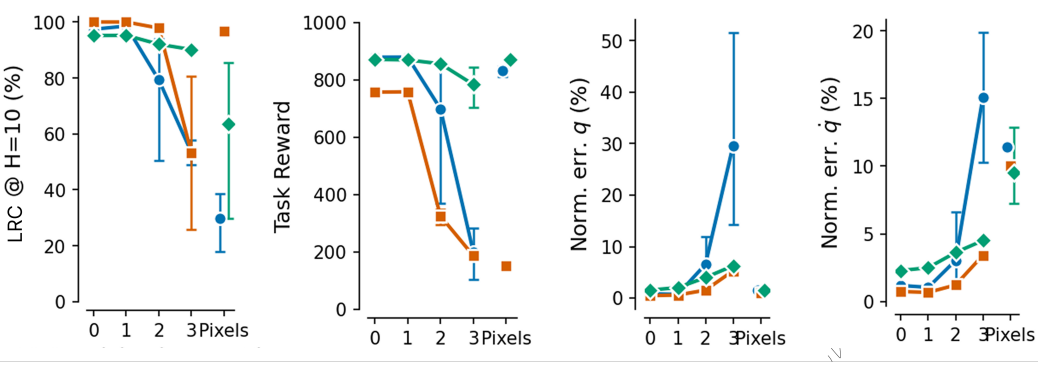


Figure 19: **Nonlinear state mixing across world-model families.** LRC at $H = 10$, task reward, normalized position error, and normalized velocity error on CartPole as the mixing depth increases from zero to three layers. Results are shown for TD-MPC2 (blue circles), LeWM (orange squares), and DreamerV3 (green diamonds); *Pixels* denotes each model's standard pixel-input condition. Error bars show variation across model seeds.

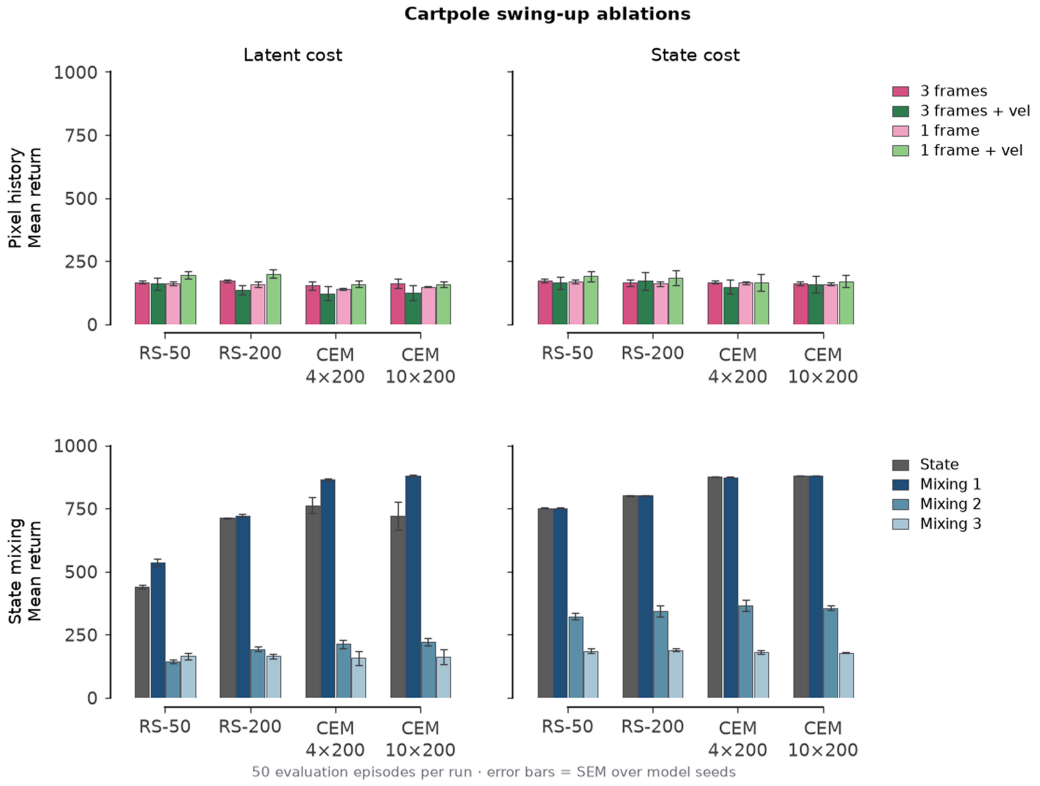


Figure 20: **LeWM controllers on the CartPole interventions.** Top: pixel models with one or three frames, with and without ground-truth velocity. Bottom: state-input models after zero to three nonlinear mixing layers. Columns compare the terminal latent cost with the decoded state cost across the planners in Figure 16. Bars are means over model seeds; error bars are the standard error. Each bar uses 50 evaluation episodes.