BEYOND MYOPIC WORLD MODELS: LONG-HORIZON END-TO-END TRAINING FOR DIRECT FUTURE PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

World models are expected to support imagination over extended temporal horizons, yet most are still trained through local few-step prediction objectives and deployed by recursively rolling out their own predictions. This creates a fundamental mismatch: few-step losses optimize local transition fidelity, while long-horizon prediction depends on how errors and gradients propagate through the entire trajectory. As a result, transitions with different downstream influence on the endpoint are treated uniformly during training, and small local errors are amplified through recursive inference. We argue that long-horizon accuracy is better achieved by optimizing directly, through an end-to-end endpoint prediction objective. To instantiate this paradigm, we introduce the **Direct Prediction World Model (DPWM)**, a non-recursive architecture that compresses an action sequence of arbitrary length into a single embedding and predicts the endpoint observation in a single forward pass. This design avoids recurrent rollout in both prediction and gradient propagation, making long-horizon end-to-end training practical. Empirically, DPWM substantially improves long-horizon endpoint prediction over recursive world-model baselines on continuous-control and pixel-based benchmarks, with larger gains as the prediction horizon increases. We further show that a recurrent any-step baseline improves substantially at long horizons when retrained with the same long-horizon endpoint objective, supporting our central claim that the training horizon, rather than the backbone alone, is a major driver of long-horizon prediction accuracy. Our results suggest that world models can benefit from being trained and evaluated at the temporal scales where they are ultimately used, shifting the focus from local transition modeling toward long-horizon predictive accuracy.

1 INTRODUCTION

World models (Ha & Schmidhuber, 2018; LeCun, 2022; Sutton, 1991) provide a compact predictive interface between past experience and future decision making. In model-based reinforcement learning and embodied control, they are expected to support imagination over extended temporal horizons: an agent must evaluate the consequence of an action sequence, compare alternative plans, or reason about delayed outcomes many steps into the future (Sutton, 1991; Hafner et al., 2019b;a; 2020; 2025; Henaff et al., 2017; Alonso et al., 2024; Hao et al., 2025). Long-horizon prediction is also a useful test of whether a model has captured the underlying dynamics (Vafa et al., 2025; Liu et al., 2026; Li et al., 2025; Xia et al., 2026; Finn et al., 2016), since errors in transition structure are often only revealed after the model is rolled forward over many steps. However, many widely used world models are trained through local transition objectives. They learn to predict the next observation, or a short segment of future observations, and long-horizon predictions are obtained at inference time by recursively feeding the model's own predictions back into itself (Hafner et al., 2019b;a; Micheli et al., 2023; Alonso et al., 2024).

This creates a mismatch between how world models are trained and how they are used. A one-step objective optimizes local transition fidelity under ground-truth inputs, but long-horizon prediction depends on the entire composed trajectory under model-generated inputs: small errors drift the rollout away from the training distribution and amplify through recursive inference (Bengio et al., 2015;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

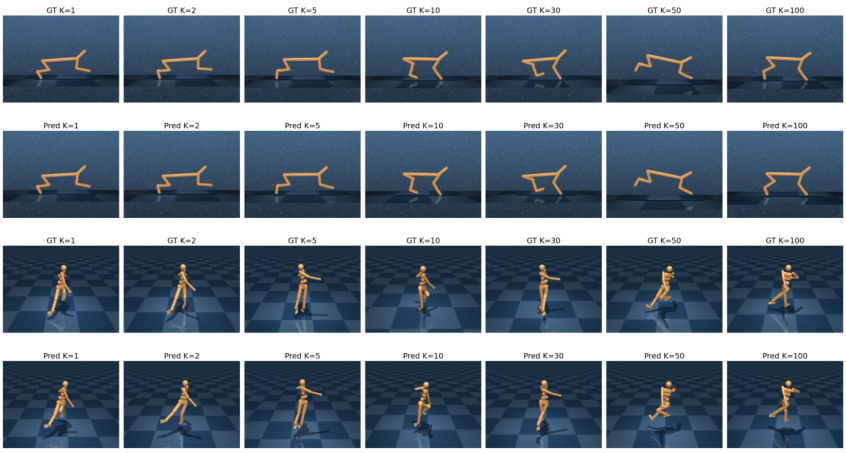


Figure 1: Long-horizon prediction visualizations rendered from states predicted by DPWM$_{K=100}$, trained and evaluated on the policy dataset.

Goyal et al., 2016; Asadi et al., 2019; Talvitie, 2017). More fundamentally, transitions are weighted uniformly by the local loss even though their downstream influence on the endpoint differs substantially with the dynamics and horizon, so improving one-step accuracy does not necessarily optimize the prediction an agent ultimately uses after tens or hundreds of imagined steps (Asadi et al., 2019; Benechehab et al., 2024). A natural alternative is to attach the loss directly to the endpoint induced by an initial observation and an action sequence, so that the optimization signal is tied to the long-horizon prediction target rather than to a collection of local transition errors (Lambert et al., 2021). However, extending this approach to long horizons introduces practical obstacles: training an autoregressive model with an endpoint loss requires unrolling the model across the full prediction horizon, increasing training cost and optimization difficulty as the horizon grows (Pascanu et al., 2012; Goyal et al., 2016). Compared to short-horizon and recursively rolled-out world modeling, long-horizon end-to-end endpoint supervision remains relatively underexplored.

To study this training paradigm directly, we introduce the **Direct Prediction World Model (DPWM)**, a non-recursive world model that predicts the endpoint observation from the initial observation and the entire variable-length action sequence in a single forward pass. DPWM first compresses the action sequence into a single embedding and then conditions a dynamics module on both the initial observation representation and the action embedding to predict the endpoint. By predicting the endpoint directly rather than rolling out intermediate observations, DPWM avoids feeding prediction errors back into later prediction steps, reducing one source of compounding rollout error. The same property also makes long-horizon end-to-end training more computationally practical because the endpoint loss is applied without recursively unrolling the model through the extended horizon.

Empirically, DPWM substantially improves endpoint prediction accuracy on continuous-control tasks in DeepMind Control Suite (Tunyasuvunakool et al., 2020), with the advantage becoming more pronounced as the evaluation horizon increases. Increasing the maximum training horizon further improves stability and generalization at long evaluation horizons, including evaluation beyond the training horizon.

The central claim of this paper, however, is not that a particular sequence encoder or backbone is inherently superior. Instead, we argue that long-horizon prediction accuracy is strongly shaped by the training objective and prediction paradigm; the architecture of DPWM serves as a practical mechanism for making this paradigm feasible at large horizons. To separate objective from architecture, we retrain autoregressive any-step baselines (Lin et al., 2025; 2026) under the same long-horizon endpoint objective and compare them against their original short-horizon counterparts. We find that these baselines benefit significantly from the same endpoint supervision, supporting the view that the training paradigm, rather than the specific backbone alone, is a major driver of long-horizon prediction accuracy. These results suggest that world models can benefit from being trained and

evaluated at the temporal scales at which they are ultimately used, rather than assuming that accurate short-horizon prediction will translate to accurate long-horizon prediction.

In summary, our main contributions are as follows:

- We formulate long-horizon world modeling as direct endpoint prediction, and analyze how this paradigm and the choice of training horizon shape long-horizon prediction accuracy.
- We propose DPWM, a non-recursive architecture that makes this paradigm practical at extended horizons, and achieves notable gains in endpoint prediction accuracy on continuous-control and pixel-based benchmarks.

2 METHOD

2.1 DIRECT PREDICTION FORMULATION

Markov decision process. We consider a deterministic Markov decision process[1] $\mathcal{M} = (\mathcal{S}, \mathcal{A}, f)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, and $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the deterministic transition function. Given a state s_t and an action sequence $a_{t:t+K-1} := (a_t, \dots, a_{t+K-1})$, the K-step deterministic transition map $\mathcal{S} \times \mathcal{A}^K \to \mathcal{S}$ induced by f is the composition:

$$f^{(K)}(s_t, a_{t:t+K-1}) := f(\cdots f(f(s_t, a_t), a_{t+1}) \cdots, a_{t+K-1}). \tag{1}$$

Although $f^{(K)}$ is by construction a K-fold composition of f, its approximator need not be recursive.

Two paradigms. A standard one-step autoregressive world model learns an approximation $\hat{f}_\theta$ to $f^{(1)} = f$ and produces long-horizon predictions by recursively applying $\hat{f}_\theta$ to its own predicted states. We instead train a model $g_\theta : \mathcal{S} \times \mathcal{A}^K \to \mathcal{S}$ to approximate $f^{(K)}$ directly for variable horizons $K \in \{1, \dots, K_{\max}\}$: rather than predicting the intermediate states $s_{t+1}, \dots, s_{t+K-1}$, the model predicts the endpoint s_{t+K} in a single forward pass.

The two paradigms differ in how prediction errors scale with K. Under standard L-Lipschitz assumptions on f and a one-step approximation error ϵ for $\hat{f}_\theta$, recursive rollout produces an endpoint error bounded by $\epsilon \sum_{i=0}^{K-1} L^i$, which grows exponentially in K whenever $L > 1$ (Asadi et al., 2018). This amplification is not a property of the modeling error itself, but of the recursive structure in which each step re-applies $\hat{f}_\theta$ to a perturbed input. For tasks with locally expansive dynamics (e.g., chaotic systems, contact-rich manipulation), $L > 1$ is the norm rather than the exception. Direct prediction, by approximating $f^{(K)}$ as a single mapping, incurs an endpoint error bounded only by $\delta_K := \sup \|g_\theta - f^{(K)}\|$, with no recursive amplification. App. A gives a complete derivation.

This does not eliminate the difficulty of long-horizon modeling; it relocates it. Recursive rollout pays a horizon-dependent *inference-time* cost, while direct prediction pays a *modeling* cost: as K grows, the target map $f^{(K)}$ becomes harder to approximate and δ_K may grow with K. The substantive question is therefore empirical: how low can δ_K be driven in practice, and is this enough to make direct prediction favorable at long horizons? We answer this experimentally in Sec. 3. The same Lipschitz factor L reappears in our analysis of training-time gradient misalignment in Sec. 2.2.

Instantiation. We instantiate the direct prediction model g_θ defined above as a composition of four learned modules. An observation encoder E maps the raw observation o_t to a latent representation $s_t = E(o_t)$. An action encoder ϕ compresses the action sequence into a fixed-dimensional embedding $z_a = \phi(a_{t:t+K-1})$. A dynamics module g produces the predicted endpoint latent $\hat{s}_{t+K} = g(s_t, z_a)$. A decoder D maps the predicted latent back to observation space, $\hat{o}_{t+K} = D(\hat{s}_{t+K})$. Together,

$$\hat{o}_{t+K} \;=\; D\big(\, g\big(E(o_t),\, \phi(a_{t:t+K-1})\big)\,\big), \tag{2}$$

which we view as the operational form of g_θ in this paper. The next subsection describes how this composition is trained; concrete instantiations of the four modules are deferred to Sec. 2.3.

[1] We present the formulation for deterministic dynamics for clarity; in stochastic environments, the same direct-prediction view applies to modeling the conditional distribution or conditional mean of s_{t+K} given $(s_t, a_{t:t+K-1})$.

2.2 LONG-HORIZON END-TO-END TRAINING

We train the full pipeline (E, ϕ, g, D) end-to-end to predict the ground-truth endpoint observation o_{t+K}. For each training example, we draw a transition tuple $(o_t, a_{t:t+K-1}, o_{t+K})$ from the dataset $\mathcal{D}$, where the horizon K is sampled from a predefined horizon distribution, instantiated as either uniform or log-uniform in our experiments, and minimize

$$\mathcal{L}(\theta) \;=\; \mathbb{E}\Big[\, \big\| D\big(g(E(o_t),\, \phi(a_{t:t+K-1}))\big) \;-\; o_{t+K} \big\|^2 \,\Big]. \tag{3}$$

A single set of weights is therefore trained to predict observations across the full range of horizons $K \in \{1, \ldots, K_{\max}\}$, rather than separately for any fixed K.

This objective stands in contrast to the standard training objective for autoregressive world models, which fits a one-step transition function $\hat{f}_\theta$ against single-step targets:

$$\mathcal{L}_{\text{1-step}}(\theta) \;=\; \mathbb{E}_{(s_k, a_k, s_{k+1}) \sim \mathcal{D}} \Big[\, \big\| \hat{f}_\theta(s_k, a_k) - s_{k+1} \big\|^2 \,\Big]. \tag{4}$$

Long-horizon predictions are then obtained at inference time by recursively applying $\hat{f}_\theta$ to its own outputs. The two paradigms also differ in how the training objective allocates gradient across transitions, which we analyze below.

Misaligned gradient weighting under 1-step training. Consider what determines endpoint accuracy under autoregressive inference: $\hat{s}_K$ is the K-fold composition of $\hat{f}_\theta$ along the action sequence, and its sensitivity to the model parameters expands by the chain rule into:

$$\frac{\partial \hat{s}_K}{\partial \theta} \;=\; \sum_{k=0}^{K-1} \underbrace{\left(\prod_{j=k+1}^{K-1} \frac{\partial \hat{f}_\theta(\hat{s}_j, a_j)}{\partial \hat{s}_j} \right)}_{J_{k \to K}} \frac{\partial \hat{f}_\theta(\hat{s}_k, a_k)}{\partial \theta}, \tag{5}$$

where $J_{k \to K}$ is the Jacobian product propagating perturbations (Pascanu et al., 2012; Bengio et al., 1994) from step k to step K. Each transition's contribution to endpoint sensitivity is thus weighted by $J_{k \to K}$, which depends on the horizon offset $K - k$ and the local Lipschitz behavior of the dynamics: early transitions dominate when $L > 1$, late ones when $L < 1$. The one-step gradient $\nabla_\theta \mathcal{L}_{\text{1-step}}$, by contrast, is a uniform average over transitions with no Jacobian product, so it allocates training signal without regard to downstream influence, and this misalignment grows with K.

The endpoint loss in equation 3 avoids this misalignment by construction, since its gradient directly targets the endpoint error whose sensitivity equation 5 describes. This is a property of the *training objective* rather than of any particular architecture: an autoregressive model trained through a K-step unrolled rollout inherits the same Jacobian weighting via the chain rule. We therefore expect the training horizon, not the backbone alone, to be a major driver of long-horizon accuracy, and test this directly in Sec. 3 (Tab. 3); DPWM serves as one instantiation of this objective that remains practical to train at large K.

2.3 MODEL DESIGN

This subsection describes the architecture used to instantiate the four modules of equation 2. The design intentionally uses standard components: the observation encoder/decoder, a Transformer encoder (Vaswani et al., 2023) for the action sequence, and a FiLM-conditioned (Perez et al., 2017) MLP for endpoint dynamics.

Observation encoder/decoder. The observation interface is the only modality-specific component of the pipeline. For state-based environments, the encoder E is an MLP, and the final output head maps the predicted latent directly back to raw state space. For pixel-based environments, E and D are matched convolutional encoder–decoders. Both are trained jointly with the action encoder ϕ and dynamics module g under the endpoint loss in equation 3.

Action sequence encoder. The action encoder ϕ maps the full action sequence $a_{t:t+K-1}$ to a fixed-dimensional embedding $z_a \in \mathbb{R}^{D_a}$. Discrete actions are mapped through a learned embedding lookup, and continuous actions through a linear projection. The resulting tokens are processed

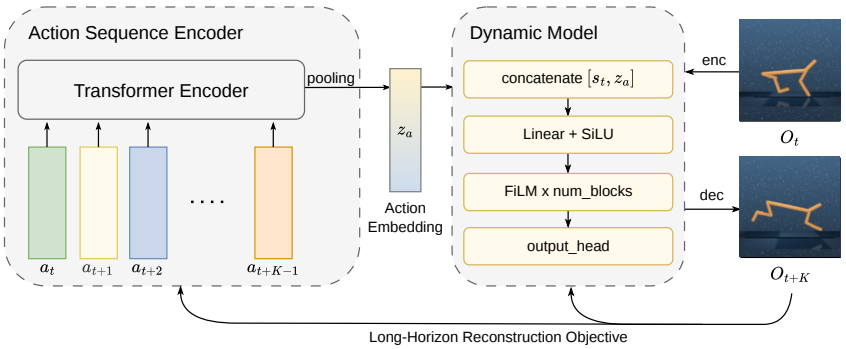


Figure 2: DPWM architecture. **Action Sequence Encoder** compresses the variable-length action sequence into a single embedding, which conditions a **Dynamic Model** that directly predicts the endpoint observation from the initial observation. Modules are trained end-to-end under a long-horizon reconstruction objective on the endpoint, without generating intermediate observations.

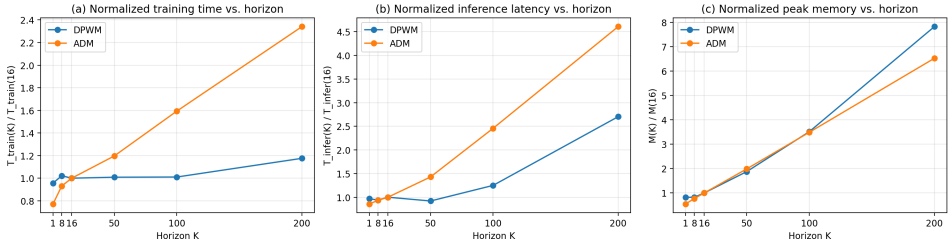


Figure 3: Normalized training time, inference latency, and peak GPU memory for DPWM and ADM across prediction horizons, using each model's value at $K = 16$ as the reference.

by a Transformer encoder with rotary positional encodings (RoPE) (Su et al., 2023). Since the entire action sequence is available at prediction time, we do not use a causal mask. In our default implementation, the Transformer uses non-causal local self-attention with a window size of 20, allowing each action token to attend to nearby past and future action tokens within the window. The final-layer action tokens are aggregated by mean pooling to produce the sequence embedding.

Dynamics module. The dynamics module g predicts the endpoint latent from the encoded initial observation and the action embedding. Let $c = [s_t, z_a]$ denote the concatenated condition. We instantiate g as a stack of FiLM-conditioned residual MLP blocks. An input projection maps c to the hidden dimension; each residual block applies feature-wise affine modulation whose scale and shift parameters are computed from c at every block, re-injecting the conditioning throughout the depth. A final linear head maps the hidden state to the prediction space: raw state space for state-based environments, or latent space before decoding for pixel-based environments.

Importantly, g is non-recursive with respect to the prediction horizon: it predicts the endpoint directly and does not generate any intermediate latent. This keeps the architecture aligned with the endpoint objective in equation 3 and avoids the recursive feedback path.

3 Experiments

3.1 Baselines

We compare DPWM with the following representative world-model baselines.

MoSim (Hao et al., 2025) is a Neural-ODE-based (Chen et al., 2018) motion simulator with physics priors of rigid body dynamics for control tasks. It targets accurate physical state prediction and serves as a strong reference on physics-driven control tasks. Because MoSim requires the prediction

Table 1: Comparison of world-model training and inference paradigms.

Method	**Backbone**	**Training supervision**	**Intermediate state/latent**	**Sequential depth vs. K**
DPWM	Transformer, non-recurrent	Long-horizon endpoint	No[1]	Constant depth[2]
ADM	GRU, recurrent	Short-horizon endpoint	Yes	$O(K)$
MoSim	Neural ODE	Local dynamics	Yes	Horizon-dependent integration
Dreamer	RSSM, recurrent	Local dynamics + auxiliary losses	Yes	$O(K)$

[1] Intermediate states can be obtained by independently querying action prefixes.
[2] Constant sequential depth with respect to horizon K; total computation still depends on sequence length.

Table 2: Raw state-space MSE of endpoint observation prediction.

		rand train, rand eval				policy train, policy eval			
Env	Horizon	$\text{DPWM}_{K=100}$	$\text{ADM}_{K=3}$	MoSim	Dreamer	$\text{DPWM}_{K=100}$	$\text{ADM}_{K=3}$	MoSim	Dreamer
Cheetah	1	**0.0277**	0.0345	0.0519	0.8361	**0.1888**	0.2624	0.2801	1.1948
	16	**0.0920**	1.8550	0.2331	1.3888	**0.2918**	17.3993	1.8238	2.7079
	100	**0.1854**	1.7240	0.3215	1.8753	**0.4452**	25.6086	11.0004	35.9435
	200	0.5110	1.7875	**0.3190**	1.8229	**1.3840**	34.0766	N/A	39.5732
Humanoid	1	**0.2449**	0.2724	0.3378	0.8921	**0.2812**	0.3910	0.6887	2.3792
	16	**0.9172**	10.8206	3.0627	11.5281	**1.9022**	28.5179	9.2534	29.7098
	100	**2.6677**	17.9088	4.5153	10.7924	**5.7811**	49.4643	17.4877	26.1435
	200	**3.2823**	18.2562	5.2478	12.3591	**7.7810**	51.0466	20.4348	30.9312
Hopper	1	0.3727	0.4960	**0.0105**	0.2807	0.0491	**0.0253**	0.0452	0.5767
	16	0.4050	0.6727	**0.1887**	0.5386	**0.0769**	5.4239	0.5634	2.2413
	100	**0.4081**	1.0853	0.4158	0.9590	**0.2348**	14.9331	30.2598	8.8345
	200	**0.5975**	1.1014	0.5982	1.0284	**2.7208**	15.3578	N/A	9.4642
Walker	1	**0.0622**	0.0780	0.0686	0.5862	0.1907	**0.1180**	0.3475	2.2887
	16	**1.7475**	2.4299	1.7650	2.3866	**2.0393**	14.9711	11.9352	8.9580
	100	**1.1604**	3.8802	15.3791	2.8103	**3.7131**	55.7769	33.7685	33.6204
	200	**1.6793**	3.8563	N/A	3.0328	**18.9293**	44.1142	N/A	31.3703

interval to coincide with the simulator timestep, for the results in Tab. 2 we set `action_repeat=1` for all methods, rather than the TD-MPC2 defaults. N/A indicates that the MoSim rollout became numerically unstable and did not produce a finite endpoint prediction.

ADM (Lin et al., 2025; 2026) is the closest any-step dynamics baseline, since it also predicts the endpoint state at a variable horizon. However, it differs from our approach in two key respects. First, it recurrently processes the action sequence through latent intermediate states. Second, in its published setting, it is trained with very short prediction horizons, e.g., $K = 2$ or 3 on the Cheetah and Humanoid tasks. In contrast, our focus is end-to-end supervision at much longer horizons. To disentangle these two factors, we evaluate ADM in two settings. Tab. 2 reproduces its published short-horizon training; Sec. 3.2 additionally reports ADM retrained under our long-horizon objective at $K_{\max} = 100$, providing a controlled comparison between the training paradigm and architecture.

Dreamer (Hafner et al., 2025) is a model-based RL method built on a recurrent state-space world model. Since our study focuses on prediction accuracy, we train only its world-model component on our datasets, without the actor–critic pipeline. Long-horizon predictions are obtained by recursively rolling the learned dynamics forward under the given action sequence.

3.2 State-based Continuous Control

Setup. We evaluate on four tasks from the DeepMind Control Suite (Tunyasuvunakool et al., 2020): `cheetah-run`, `humanoid-walk`, `hopper-hop`, and `walker-run`. For each task, the raw observation consists of the physics state $(q_{\text{pos}}, q_{\text{vel}})$. The corresponding dimensions of $(q_{\text{pos}}, q_{\text{vel}})$ are $(9, 9)$, $(28, 27)$, $(7, 7)$, and $(9, 9)$, respectively, and the action spaces are $\mathbb{R}^6$, $\mathbb{R}^{21}$, $\mathbb{R}^4$, and $\mathbb{R}^6$.

For each environment we construct training and evaluation datasets distinguished by the action distribution used to roll out trajectories. The *random* dataset draws actions uniformly from the action space. The *policy* dataset uses a publicly released TD-MPC2 (Hansen et al., 2024) expert checkpoint. A pure expert, however, produces highly consistent trajectories that converge to a narrow distribution of states and actions, particularly over long horizons. We mitigate this with an *intermix* protocol that injects short random segments into expert rollouts, enriching the diversity of trajectories seen during training.

We evaluate prediction accuracy under two matched training and evaluation action distributions: random-policy training and evaluation, and intermix-policy training and evaluation. We report endpoint MSE rather than averaging across the rollout because compounding errors in autoregressive inference (Sec. 2.1) are most pronounced at the horizon endpoint; averaging across intermediate steps would dilute this signal with easier short-horizon predictions. We evaluate at horizons $K \in \{1, 16, 100, 200\}$.

DPWM is trained with our long-horizon end-to-end endpoint objective using $K_{\max} = 100$. ADM is trained using its published short-horizon setting with $K_{\max} = 3$. Autoregressive baselines are trained with local one-step prediction losses, corresponding to $K_{\max} = 1$, and are evaluated by rolling out their predictions to the target horizon.

Evaluation results. DPWM achieves substantially lower endpoint MSE as the prediction horizon increases in most settings (Tab. 2), with the gap widening as K grows, while ADM and MoSim are competitive mainly at short horizons. This trend reflects a fundamental difference in how long-range prediction is learned. ADM composes short transitions recursively, so small prediction errors alter subsequent inputs and are progressively amplified over the rollout. MoSim introduces an explicit physics prior that can improve local prediction, but systematic model mismatch in the learned physical dynamics can still accumulate over long horizons and eventually cause divergence. In contrast, DPWM directly learns the effect of an entire action sequence under multi-horizon endpoint supervision, aligning its training objective with long-horizon evaluation while avoiding repeated self-conditioning on predicted states. These results suggest that accurate long-horizon prediction benefits from explicitly learning the long-range transition, rather than relying solely on short-step accuracy or physics-informed priors.

Selection of training parameter $K_{\max}$. We compare DPWM trained with $K_{\max} \in \{50, 100, 400\}$ on intermix-policy data, evaluated across $K \in [1, 200]$ under three settings: in-distribution evaluation on `humanoid_walk` and `cheetah_run`, and out-of-distribution evaluation on `cheetah_run` with random-action trajectories (Fig. 4). Models trained with $K_{\max} = 50$ perform well only within their training horizon and exhibit sharp error growth beyond it, suggesting that long-horizon prediction requires long-horizon training. Models trained with $K_{\max} = 100$ remain stable across the full evaluation range, including the $K = 200$ extrapolation regime. Models trained with $K_{\max} = 400$ are comparable to $K_{\max} = 100$ at most in-distribution horizons but exhibit elevated error at very short horizons; under out-of-distribution evaluation on random-action trajectories, however, $K_{\max} = 400$ yields consistently lower error than $K_{\max} = 100$, providing additional evidence that longer training horizons yield stronger generalization to unseen action distributions. We adopt $K_{\max} = 100$ as the default for all main experiments based on its balanced performance across horizons.

Training horizon is a major driver of long-horizon accuracy. To isolate the contribution of our long-horizon training paradigm from the architectural choice (Transformer vs. RNN), we compare three models on `humanoid_walk` in Tab. 3: DPWM$_{K=100}$, ADM$_{K=100}$ that aligns with DPWM setting, and ADM$_{K=3}$ that aligns with the published setting. Retraining ADM under our long-horizon paradigm substantially closes its long-range prediction gap to DPWM at evaluation horizons $K \geq 16$. ADM$_{K=3}$ and ADM$_{K=100}$ share the same encoder and inference procedure, differing only in the training horizon. Short-horizon supervision propagates gradient through only a few dynamics steps and offers little training pressure on structure that emerges over longer horizons. Long-horizon end-to-end supervision, by attaching the loss directly to a K-step prediction, forces the encoder to represent features whose evolution remains predictable at horizons beyond the reach of local losses.

Computational scaling. To test whether the non-recurrent factorization provides a practical advantage at long horizons, we benchmark DPWM against ADM while varying only the prediction horizon. All measurements use the same dataset, batch size, and GPU. Training time measures a

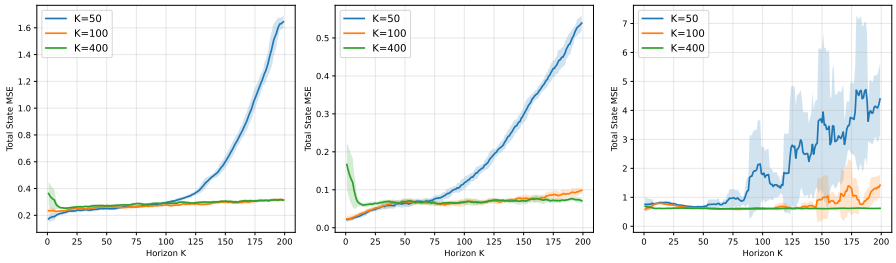


Figure 4: Endpoint prediction MSE computed in *normalized* state space across evaluation horizons for models trained with $K_{\max} \in \{50, 100, 400\}$ on intermix-policy data. Results are shown for `humanoid_walk` evaluated on intermix-policy (left), `cheetah_run` on intermix-policy (middle), and `cheetah_run` on random-policy (right).

Table 3: Endpoint prediction MSE on `humanoid_walk`, computed in *normalized* state space. The `action_repeat` follows the TD-MPC2 default setting. The suffix indicates the training horizon $K_{\max}$, with $\text{ADM}_{K=3}$ following the published ADM setting.

		Policy train, policy eval			Policy train, rand eval		
Env	Horizon	$\text{DPWM}_{K=100}$	$\text{ADM}_{K=100}$	$\text{ADM}_{K=3}$	$\text{DPWM}_{K=100}$	$\text{ADM}_{K=100}$	$\text{ADM}_{K=3}$
Humanoid	1	0.2700	0.3448	**0.1385**	0.4560	0.5639	**0.2637**
	16	**0.2394**	0.2577	0.4318	**0.4340**	0.4732	0.5763
	100	**0.2658**	0.2999	0.7756	**0.6058**	0.6318	0.8011
	200	**0.3234**	0.3483	0.8882	**0.6305**	0.6622	0.9701

complete optimization step, whereas inference latency measures only the forward pass. We additionally record peak allocated GPU memory. Each quantity is normalized by the model's own value at $K = 16$, which focuses the comparison on horizon-dependent scaling instead of absolute size. As shown in Fig. 3, DPWM exhibits substantially milder growth in training time and inference latency as the horizon increases, consistent with its non-recurrent action-sequence processing, while memory usage still grows with K and is not uniformly lower than ADM.

3.3 Pixel-Based Discrete Control

Setup. We evaluate on `Pong` from the Atari Learning Environment (Bellemare et al., 2013). Observations are single-frame 160×160 RGB images cropped from the native 210×160 to remove the static scoreboard region; we apply an action repeat of 4 but no frame stacking and no auxiliary state information, so each prediction is conditioned on a single observed frame. The action space contains 4 discrete actions.

For training data, we use a deterministic chase-ball heuristic policy: the green paddle tracks the ball's vertical position, except when the ball lies within a narrow band, where the paddle stays still. This

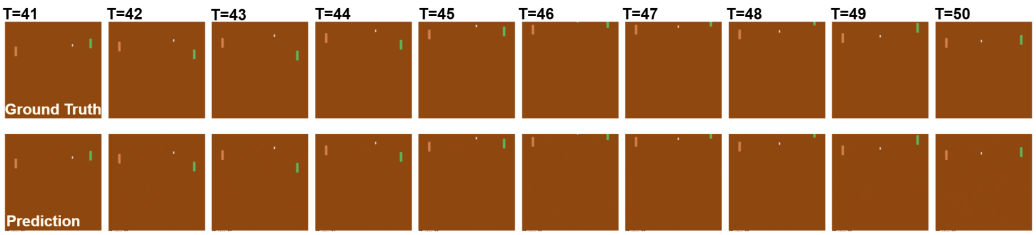


Figure 5: DPWM trajectory generation on Pong. Top: ground-truth observations. Bottom: per-step predictions $\hat{o}_t$, each independently generated from the same initial observation o_0 and the action prefix $a_{0:t-1}$. Frames $T = 41$ to $T = 50$ from a 70-step rollout; full rollout in App. B.

Table 4: Endpoint pixel MSE ($\times 10^{-4}$) on Atari Pong.

Env	Method	1	16	100	200	400
Pong	DPWM$_{K=100}$	1.1759	**5.5467**	**5.9387**	**5.5852**	**5.9844**
	ADM$_{K=100}$	**1.1252**	5.6842	6.8619	6.9098	6.7410

deliberate miss zone produces approximately tied scores against the built-in opponent, balancing the distribution of game states in the training data. DPWM is trained with $K_{\max} = 100$ on this policy data and evaluated on held-out trajectories from the same distribution.

Results. Fig. 5 shows a 10-frame slice of a 70-step trajectory generated by DPWM$_{K=100}$. Although trained only on endpoint targets, DPWM produces a per-step trajectory by querying successive action prefixes $a_{0:t-1}$ from the same initial observation o_0, with each frame obtained by an independent forward pass rather than autoregressive rollout. The predicted ball closely tracks the ground-truth position and moves smoothly across consecutive frames, forming a coherent trajectory. Quantitatively, DPWM maintains low pixel error across horizons and outperforms ADM$_{K=100}$ as K grows (Tab. 4).

4 Related Work

World Models. A world model is an action-conditioned generative model of environment dynamics: given an initial state and a sequence of actions, it predicts the resulting future observations (Ha & Schmidhuber, 2018; Sutton, 1991; LeCun, 2022; Bruce et al., 2024). Such models serve as learned surrogates for the environment, supporting downstream uses including planning (Hansen et al., 2022; 2024) and model-based reinforcement learning (Hafner et al., 2019a; 2020; 2025; Sutton, 1991; Henaff et al., 2017; Micheli et al., 2023; Alonso et al., 2024). Recent work targets multiple dimensions of model quality including physical fidelity, prediction accuracy, temporal coherence over rollouts, and generalization to new states and action distributions (Kang et al., 2025; Liu et al., 2026; Li et al., 2025; Xia et al., 2026; Hao et al., 2025). Across these settings, however, many widely used systems still rely on one-step or short-horizon transition prediction, with long-horizon behavior obtained at inference time via recursive rollout.

Multi-Step Prediction and Action Sequence Modeling. A line of work (Eysenbach et al., 2018; Machado et al., 2023) compresses consecutive low-level actions into higher-level units such as learned skills, and plans or predicts at this level rather than at the primitive-action level. Another direction processes raw action sequences to reduce the number of autoregressive prediction steps without explicit semantic abstraction: some (Zhang et al., 2023; Zhao et al., 2023) predict states at a fixed horizon offset, while variable-length self-segmenting models (Gumbsch et al., 2024; Jayaraman et al., 2018) learn to find the next endpoint in an unsupervised manner. Hierarchical latent world models (Zhang et al., 2026) also encode variable-length action sequences with a Transformer encoder, but they operate at shorter horizons and focus on downstream task performance. These works either focus on finding good segmentation points or operate at short horizons; DPWM instead predicts over variable-length action sequences without requiring a temporal segmentation prior, training at extended horizons and processing the entire sequence non-recursively in a single forward pass.

5 Conclusion

We examined long-horizon endpoint supervision as an alternative to local transition training for world models. To make this paradigm practical, we proposed DPWM, a non-recursive model that directly predicts the endpoint observation from an initial observation and an action sequence. Across continuous-control and pixel-based benchmarks, DPWM improves endpoint accuracy at extended horizons. Our ablations further show that the benefit is not tied to the DPWM backbone alone: other model designs, including recurrent ones, also improve under the same end-to-end endpoint objective. These results point to supervision horizon as an important design choice for world models used in long-range prediction.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experimental design, assist with software implementation and debugging, and help interpret experimental results. We also used generative AI tools to assist with manuscript editing and revision, improve clarity and readability, and support literature search and summarization.

All AI-assisted outputs were reviewed by the authors. Experimental results, methodological decisions, mathematical arguments, code, citations, and claims included in the final manuscript were independently checked and verified by the authors. The authors take responsibility for the final content of this work, including any text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. DIAMOND: Diffusion for world modeling: Visual details matter in atari. In *Advances in Neural Information Processing Systems*, 2024.

Kavosh Asadi, Dipendra Misra, and Michael L. Littman. Lipschitz continuity in model-based reinforcement learning, 2018. URL https://arxiv.org/abs/1804.07193.

Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L. Littman. Combating the compounding-error problem with a multi-step model, 2019. URL https://arxiv.org/abs/1905.13320.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL http://dx.doi.org/10.1613/jair.3912.

Abdelhakim Benechehab, Giuseppe Paolo, Albert Thomas, Maurizio Filippone, and Balázs Kégl. Multi-timestep models for model-based reinforcement learning, 2024. URL https://openreview.net/forum?id=Rh4DmXaf8R.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks, 2015. URL https://arxiv.org/abs/1506.03099.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments, 2024. URL https://arxiv.org/abs/2402.15391.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function, 2018. URL https://arxiv.org/abs/1802.06070.

Chelsea Finn, Ian J. Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *ArXiv*, abs/1605.07157, 2016. URL https://api.semanticscholar.org/CorpusID:2659157.

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing: a new algorithm for training recurrent networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 4608–4616, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Christian Gumbsch, Noor Sajid, Georg Martius, and Martin V. Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TjCDNssXKU.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

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

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*, pp. 2555–2565, 2019b.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pp. 1–7, 2025.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control, 2022. URL https://arxiv.org/abs/2203.04955.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024. URL https://arxiv.org/abs/2310.16828.

Chenjie Hao, Weyl Lu, Yifan Xu, and Yubei Chen. Neural motion simulator pushing the limit of world models in reinforcement learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 27608–27617, June 2025.

Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.

Dinesh Jayaraman, Frederik Ebert, Alexei A. Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames, 2018. URL https://arxiv.org/abs/1808.07784.

Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How far is video generation from world model: A physical law perspective, 2025. URL https://arxiv.org/abs/2411.02385.

Nathan O. Lambert, Albert Wilcox, Howard Zhang, Kristofer S. J. Pister, and Roberto Calandra. Learning accurate long-term dynamics for model-based reinforcement learning, 2021. URL https://arxiv.org/abs/2012.09156.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Xinyi Li, Zaishuo Xia, Weyl Lu, Chenjie Hao, and Yubei Chen. Smallworlds: Assessing dynamics understanding of world models in isolated environments, 2025. URL https://arxiv.org/abs/2511.23465.

Haoxin Lin, Yu-Yan Xu, Yihao Sun, Zhilong Zhang, Yi-Chen Li, Chengxing Jia, Junyin Ye, Jiaji Zhang, and Yang Yu. Any-step dynamics model improves future predictions for online and offline reinforcement learning. In *The 13th International Conference on Learning Representations (ICLR'25)*, Singapore, 2025.

Haoxin Lin, Siyuan Xiao, Yi-Chen Li, Zhilong Zhang, Yihao Sun, Chengxing Jia, and Yang Yu. Adm-v2: Pursuing full-horizon roll-out in dynamics models for offline policy learning and evaluation. In *The 14th International Conference on Learning Representations (ICLR'26)*, Rio de Janeiro, Brazil, 2026.

Ziming Liu, Sophia Sanborn, Surya Ganguli, and Andreas Tolias. From kepler to newton: Inductive biases guide learned world models in transformers. *ArXiv*, abs/2602.06923, 2026. URL https://api.semanticscholar.org/CorpusID:285401944.

Marlos C. Machado, Andre Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation, 2023. URL https://arxiv.org/abs/2110.05740.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, 2012. URL https://api.semanticscholar.org/CorpusID:14650762.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Erik Talvitie. Self-correcting models for model-based reinforcement learning, 2017. URL https://arxiv.org/abs/1612.06018.

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. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Keyon Vafa, Peter G. Chang, Ashesh Rambachan, and Sendhil Mullainathan. What has a foundation model found? using inductive bias to probe for world models. *ArXiv*, abs/2507.06952, 2025. URL https://api.semanticscholar.org/CorpusID:280150828.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Zaishuo Xia, Yukuan Lu, Xinyi Li, Yifan Xu, and Yubei Chen. Cloning deterministic worlds: The critical role of latent geometry in long-horizon world models, 2026. URL https://arxiv.org/abs/2510.26782.

Jingwei Zhang, Jost Tobias Springenberg, Arunkumar Byravan, Leonard Hasenclever, Abbas Abdolmaleki, Dushyant Rao, Nicolas Heess, and Martin Riedmiller. Leveraging jumpy models for planning and fast learning in robotic domains, 2023. URL https://arxiv.org/abs/2302.12617.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, and Nicolas Ballas. Hierarchical planning with latent world models, 2026. URL https://arxiv.org/abs/2604.03208.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

A Lipschitz Proof of Error Bound

We now compare how prediction errors scale with the horizon K under the two modeling paradigms. Throughout this analysis, we assume the dynamics f are L-Lipschitz in the state argument: for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$,

$$\|f(s,a) - f(s',a)\| \le L\,\|s - s'\|. \tag{6}$$

Recursive rollout. Let $\hat{f}$ denote a learned one-step model with bounded approximation error $\|\hat{f}(s,a) - f(s,a)\| \le \epsilon$ for all (s,a). Define the k-step rollout error as $e_k := \|\hat{s}_{t+k} - s_{t+k}\|$, where $\hat{s}_{t+k}$ is obtained by recursively applying $\hat{f}$ to its own predictions. A standard triangle inequality yields

$$\begin{aligned} e_{k+1} &= \|\hat{f}(\hat{s}_{t+k}, a_{t+k}) - f(s_{t+k}, a_{t+k})\| \\ &\le \underbrace{\|\hat{f}(\hat{s}_{t+k}, a_{t+k}) - f(\hat{s}_{t+k}, a_{t+k})\|}_{\le \epsilon} + \underbrace{\|f(\hat{s}_{t+k}, a_{t+k}) - f(s_{t+k}, a_{t+k})\|}_{\le L\, e_k} \le \epsilon + L\, e_k. \end{aligned} \tag{7}$$

Unrolling the recursion from $e_0 = 0$ gives

$$e_K \;\le\; \epsilon \sum_{i=0}^{K-1} L^i \;=\; \begin{cases} \epsilon\,\dfrac{L^K - 1}{L - 1}, & L \ne 1, \\ K\,\epsilon, & L = 1. \end{cases} \tag{8}$$

This compounding-error bound has been observed in various forms across the model-based RL literature, which is linear in K when $L = 1$, and grows exponentially in K when $L > 1$. Crucially, this amplification arises not from the modeling error ϵ itself, but from the recursive structure of rollout: each step re-applies f to a perturbed input, and the Lipschitz constant L propagates the perturbation forward. For control tasks with locally expansive dynamics (e.g., chaotic systems, contact-rich manipulation), $L > 1$ is the norm rather than the exception, making this error amplification a *structural* obstacle to long-horizon prediction.

Direct prediction. Let $g_\theta : \mathcal{S} \times \mathcal{A}^K \to \mathcal{S}$ denote our direct prediction model, which approximates $f^{(K)}$ as a single mapping. Denote its endpoint approximation error by

$$\delta_K \;:=\; \sup_{s_t,\, a_{t:t+K-1}} \big\| g_\theta(s_t, a_{t:t+K-1}) - f^{(K)}(s_t, a_{t:t+K-1}) \big\|. \tag{9}$$

The endpoint error of direct prediction is then bounded by δ_K alone, with no recursive amplification term. Comparing with equation 8, direct prediction yields a tighter long-horizon bound whenever

$$\delta_K \;<\; \epsilon \sum_{i=0}^{K-1} L^i. \tag{10}$$

We emphasize that direct prediction does not eliminate the difficulty of long-horizon modeling; it relocates it. Recursive rollout incurs a horizon-dependent *inference-time* amplification through the recursive factor $\sum_{i=0}^{K-1} L^i$ in equation 8. Direct prediction avoids this recursive amplification, but pays a *modeling* cost: as K grows, the target map $f^{(K)}$ can become harder to approximate, and the endpoint error δ_K may grow with the prediction horizon. The substantive question is therefore empirical: *how fast does δ_K grow?* When $L > 1$, if δ_K grows more slowly than the rollout bound $\epsilon \sum_{i=0}^{K-1} L^i$, direct prediction can become favorable at long horizons. When $L = 1$, the corresponding comparison is against the linear bound $K\epsilon$. For contractive dynamics with $L < 1$, the rollout bound is less severe, and the advantage of direct prediction must be established empirically.

B DPWM Prediction Visualization

Fig. 6 shows the full 70-step Pong trajectory corresponding to the 10-frame slice in Fig. 5. DPWM is trained on trajectories from a simple chase-ball policy and evaluated here on a held-out trajectory from the same policy. The sequence is obtained by querying the model at successive horizons, where the prediction at time t is conditioned on o_0 and $a_{0:t-1}$ rather than on previously predicted frames.

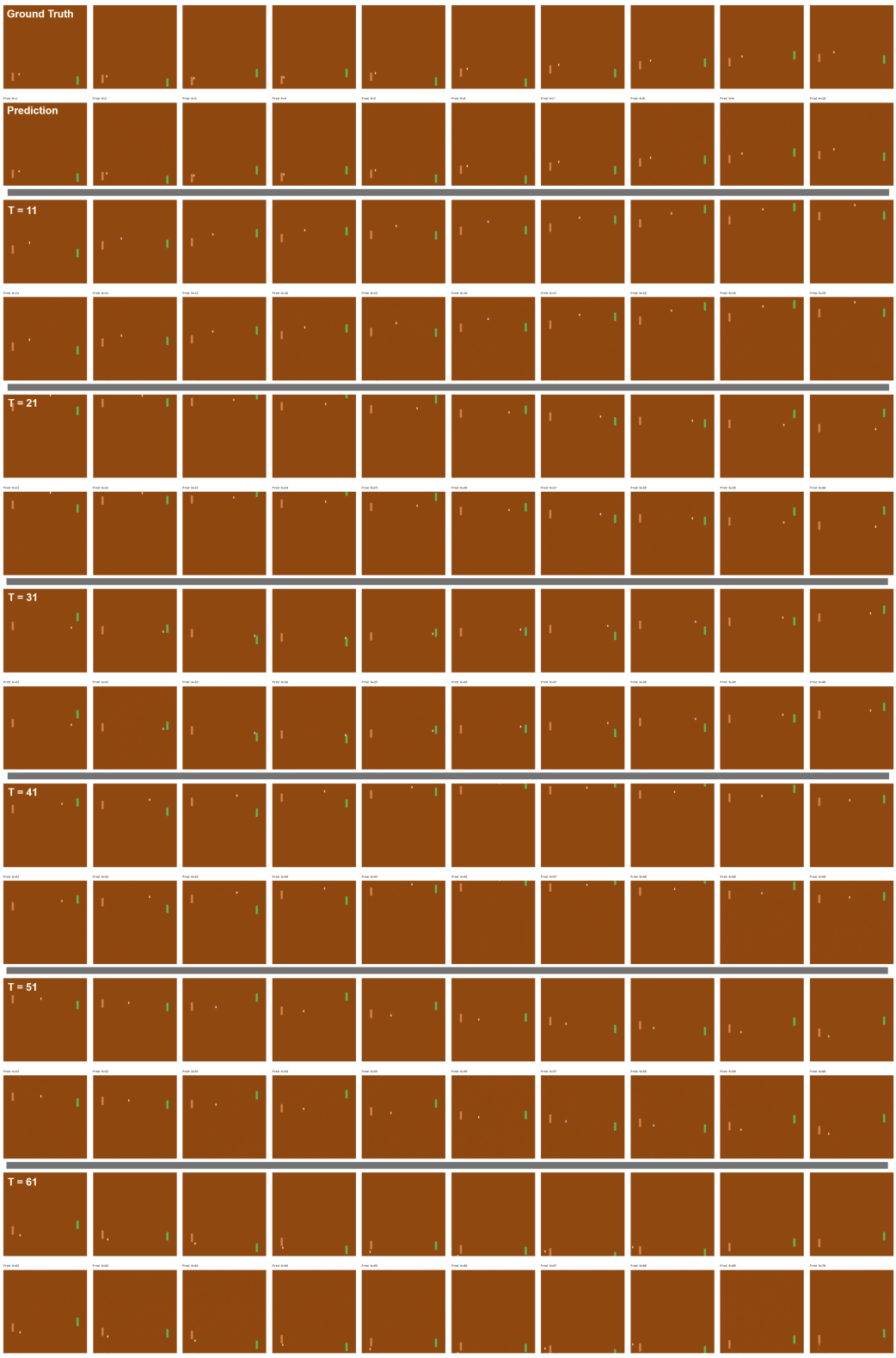


Figure 6: Full DPWM 70-step trajectory generation on Pong. Top: ground-truth observations. Bottom: per-step predictions $\hat{o}_t$, each independently generated from the same initial observation o_0 and the action prefix $a_{0:t-1}$.

Table 5: Default DPWM hyperparameters and implementation settings.

Component	Hyperparameter	Value
State observation	Cheetah state/action dim	18 / 6
	Humanoid state/action dim	55 / 21
	Hopper state/action dim	14 / 4
	Walker state/action dim	18 / 6
	Encoder	4-layer MLP, width 256
Pixel observation	Resolution (crop)	160×160 (from 210×160)
	Action repeat / discrete acts	4 / 4
	Conv/deconv blocks	5 / 5
	Channel widths	$32, 64, 128, 256, 256$
	Latent dim	256
Transformer Action encoder	Token dim D_a	256
	Layers	5
	Heads	4
	Head dim	64
	FFN expansion / dropout	$4\times$ / 0.1
	Attention window size	20
Dynamics module	Condition	$[s_t, z_a]$
	Hidden width	512
	Hidden blocks	10
Optimization	Optimizer	Adam
	Learning rate	3×10^{-4}
	LR schedule	cosine to 10^{-10}
	State batch size	256
	Pixel batch size	128
	Gradient clipping	1.0
	Epochs	3600
	Max updates per epoch	2000 state / 500 pixel

C Hyperparameter and Implementation Details

Table 5 summarizes the architecture and training configuration of DPWM. All modules are trained jointly with the endpoint reconstruction loss mentioned in Sec. 2.2.

Observation encoder and decoder. For state-based DeepMind Control experiments, the observation encoder is a 4-layer MLP with width 256 and ReLU activations, mapping the raw physics state o_t to a latent representation $s_t \in \mathbb{R}^{256}$. For these experiments, the dynamics output head maps directly back to raw state space, where the endpoint MSE is computed.

For pixel-based Pong experiments, we use a matched convolutional encoder–decoder. The observation is a single RGB frame cropped from the native 210×160 resolution to 160×160 to remove the static scoreboard region, with an action repeat of 4 and no frame stacking. Pixel values are normalized to $[0, 1]$, and the reconstruction loss is computed in this normalized space (the decoder ends in a sigmoid). The encoder applies five stride-2 convolutional blocks with kernel size 4, padding 1, GroupNorm, and SiLU activations, using channel widths $32, 64, 128, 256, 256$. The resulting 5×5 feature map is flattened and projected to a 256-dimensional latent.

Action sequence encoder. The default action encoder is a RoPE Transformer. Continuous actions are projected linearly to action tokens, while discrete actions are mapped through a learned embedding lookup. The token dimension is $D_a = 256$. The Transformer uses 5 layers, 4 attention heads, head dimension 64, a $4\times$ feed-forward expansion with GELU activations, and dropout 0.1. We use rotary positional encodings (RoPE) and bidirectional (non-causal) local self-attention with window

size 20. The final action embedding $z_a \in \mathbb{R}^{256}$ is obtained by mean-pooling the final-layer action tokens.

Dynamics module. The dynamics module is a FiLM-conditioned residual MLP. Given the encoded observation s_t and the action embedding z_a, we form the condition $c = [s_t, z_a]$. The condition is projected to the hidden width with a linear layer and SiLU activation, then processed by a stack of FiLM residual blocks. Each block applies LayerNorm, a linear layer, SiLU activation, feature-wise affine modulation, and a residual connection. We use hidden width 512 and 10 blocks.

Training. We train with Adam using learning rate 3×10^{-4} and gradient clipping with maximum norm 1.0. The learning rate follows cosine annealing to 10^{-10} over 3600 epochs. State-based tasks use batch size 256, and pixel-based tasks use 128.