k-WM: PREDICTING k STEPS AT ONCE FOR LONG-HORIZON PLANNING WITH WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models have become an appealing approach to visual control, enabling goal-conditioned optimization at test time and generalization across tasks with a single model. In practice, however, planning success remains limited to short horizons, and gradient-based planners driven by autoregressive rollouts still underperform sampling-based methods, despite their potential to optimize action sequences efficiently through model gradients. We hypothesize that two properties of next-step prediction contribute to these limitations: autoregressive rollouts compound prediction errors, and planning gradients must traverse a chain of nested model calls. We study a simple alternative: predicting the entire future sequence in a single forward pass. We introduce k-WM, a minimal extension to transformer-based JEPA world models: a multi-horizon training objective and input layout that let the predictor map a variable-length action sequence directly to the corresponding sequence of future latent states in a single forward pass. We apply k-WM to three classes of JEPA world models and evaluate on four environments. Overall, k-WM improves gradient-based planning by 37% and sampling-based planning by 15% over the existing next-step world models it extends. Our analysis shows that the gains cannot be explained by parallel rollout alone: improvements persist under autoregressive inference and coincide with substantial changes in the learned representation. Parallel prediction additionally reduces long-horizon inference cost. Together, k-WM offers a simple, broadly applicable path to more accurate and efficient long-horizon control. Videos are available at `https://anon-k-wm.github.io`.

1 INTRODUCTION

The tremendous success of imitation learning in robotics (Kim et al., 2025; Physical Intelligence et al., 2025) notwithstanding, the stringent expert-data requirement for behavior cloning policies and the lack of real world adaptation after policy training motivate an alternative path towards physical intelligence. Model-based control with a learned world model is emerging as an alternative paradigm, which offers flexible data requirements and the potential for deployment-time adaptation through classical optimization algorithms (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2025; Hansen et al., 2024). Within this paradigm, joint embedding predictive architecture (JEPA) world models (LeCun, 2022; Assran et al., 2023; Bardes et al., 2024) are particularly compelling. The dynamics are learned in a latent space rather than directly in the high-dimensional image space, allowing the model to run efficiently while avoiding explicit pixel reconstruction. A series of recent works (Zhou et al., 2024; Sobal et al., 2025; Maes et al., 2026) has made progress towards a general recipe for training and planning with such models.

Two shortcomings of the current recipe stand out. As seen in Zhou et al. (2024), planning success remains restricted to short horizons, and gradient-based planners, despite having direct access to the gradients of the world model, are substantially outperformed by their slower sampling-based counterparts. We identify two potential causes, both rooted in autoregressive world-model rollout. First, each rollout step feeds the model's own prediction error back as input, causing errors to compound with horizon (Ross et al., 2011; Venkatraman et al., 2015; Lambert et al., 2022). Second, the planning gradient must backpropagate through every rollout step, introducing backpropagation through time (BPTT). At long horizons, this can lead to vanishing, exploding, or poorly conditioned gra-

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.

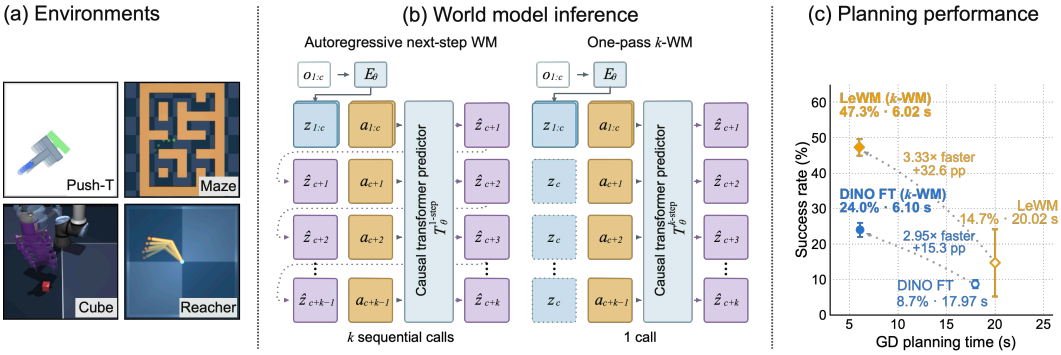

Figure 1: **Overview.** We introduce k-WM, a minimal extension that lets a transformer-based world model predict k future latents from an action sequence in one forward pass instead of rolling out autoregressively. **Left:** we evaluate k-WM on four environments, Push-T, Maze, Cube, and Reacher. **Middle:** comparison of next-step WM and k-WM inference: the former needs k sequential predictor calls, the latter a single call over the padded input sequence. **Right:** long-horizon gradient-based planning on Push-T (H=10): k-WM models (filled) achieve better performance and run faster than their next-step baselines (hollow), across two world model classes.

dients (Bengio et al., 1994; Pascanu et al., 2013), making the planning optimization more difficult. An analogous motivation underlies the shift from recurrent sequence models toward transformer architectures, which remove explicit recurrent computation across sequence positions.

These difficulties motivate separating the sequential structure of the dynamics from the computation used to predict them. Viewing the dynamics as a transformation semigroup (Liu et al., 2026) — where each action induces a state transition and an action sequence composes them — suggests that a world model can learn the composed effect directly, without recursive rollout.

Following this, we propose k-WM, a minimal extension to transformer-based JEPA world models: a multi-horizon training objective and input layout that let the causal transformer predictor map a sequence of actions, together with a short history of context latents, directly to the sequence of future latents, supervised against encoded future observations. The encoder and predictor architecture are left unchanged. k-WM reduces long-horizon prediction error, improves gradient-based planning success by 37% and sampling-based planning success by 15% over the existing next-step world models it extends, averaged across world model classes and environments, and achieves up to 3× faster rollout at long horizons. We further show that k-WM promotes representations whose latent geometry aligns with task-relevant state distances. Concurrently, Gao & Xu (2026) replaces the autoregressive rollout of LeWM with parallel prefix prediction; we compare against them directly in Table 1 and discuss the differences in Section 2.

We summarize our contributions as follows.

- We propose k-WM, a minimal multi-horizon extension to transformer-based world models, consisting of a training objective and input layout, that reduces autoregressive error accumulation and BPTT across world-model rollout steps during gradient-descent planning (Section 3).
- Across three world model classes and four environments, k-WM improves goal-conditioned planning in many settings, particularly with end-to-end trained encoders, while parallel inference substantially reduces long-horizon planning cost (Section 4).
- By independently varying training horizon and inference chunk length, we show that the benefits cannot be attributed to parallel rollout alone; longer-horizon supervision changes the learned latent representation and can improve planning even under autoregressive execution. (Sections 4.7 and 4.8).

2 Related Work

Latent world models for planning. World models (Nguyen & Widrow, 1989; Ha & Schmidhuber, 2018; LeCun, 2022) have proven to be a powerful tool for learning and control (Hafner et al., 2025; Hansen et al., 2024; Bruce et al., 2024; Agarwal et al., 2025). Recently, latent world models (Le-

Cun, 2022) have shown strong generalization and enable planning towards unseen goals (Zhou et al., 2024; Sobal et al., 2025; Maes et al., 2026). They have been applied to navigation (Bar et al., 2025), manipulation (Gautam Goswami et al., 2025), and aerial robotics (Rao et al., 2026). Specifically, DINO-WM (Zhou et al., 2024) plans over frozen DINOv2 (Oquab et al., 2024) patch features, PLDM (Sobal et al., 2025) supervises a sequence of recursively predicted latents, and LeWM (Maes et al., 2026) trains the encoder and predictor end-to-end with the SIGReg regularizer from LeJEPA (Balestriero & LeCun, 2025). LpWM (Kuang et al., 2026) keeps this recipe but replaces the isotropic Gaussian target with a rectified generalized Gaussian, yielding sparse latents, and Temporal Straightening (Wang et al., 2026) regularizes latent trajectories to be straight. Semigroup-JEPA (Liu et al., 2026) finds that multi-step training improves the encoder, and Sobal et al. (2022) show that JEPA objectives can favor slow features. Whether trained with teacher forcing or autoregressively, all of these models roll out the dynamics recursively at test time. In this work, we replace the recursive rollout with a single forward pass that maps an action sequence directly to future latents.

Fast LeWM (Gao & Xu, 2026) parallelizes LeWM rollout, improving rollout quality, speed, and representation. We vary training horizon and inference chunk length separately across encoder classes, separating the effect of multi-horizon supervision from that of parallel execution, and evaluate both CEM and gradient-based planning.

Parallel prediction of sequential dynamics. The sequential nature of a dynamical system does not preclude predicting its trajectory in parallel. In sequence modeling, non-autoregressive decoding (Gu et al., 2017; Ghazvininejad et al., 2019) and multi-token prediction (Gloeckle et al., 2024) trade sequential generation for parallel outputs, and multi-token targets can also improve what the model learns (Gloeckle et al., 2024). On the theory side, shallow transformers learn parallel shortcuts to sequential automata (Liu et al., 2023) and can represent massively parallel computation (Sanford et al., 2024). In control, sequence prediction has mostly been applied to actions: action-chunking policies (Zhao et al., 2023; Chi et al., 2025; Lee et al., 2024; Zhou et al., 2026) and trajectory diffusion (Janner et al., 2022) generate a whole sequence in one shot. Multi-step dynamics models that map an action sequence directly to future states (Mishra et al., 2017; Asadi et al., 2019) avoid compounding error, but have been studied in low-dimensional state spaces. We ask whether a transformer predictor can learn such a parallel approximation of visual dynamics in latent space, and whether doing so benefits goal-conditioned planning.

3 k-WM

Here we describe the JEPA world model setting and how k-WM modifies it: the encoder, which k-WM leaves unchanged; the predictor input layout; the training objective; and inference. The concrete world models we build on are given in Section 4.1. The overall architecture and the distinction between next-step and parallel multi-horizon prediction are illustrated in Figure 2. We distinguish the training horizon k, the number of future targets supervised in each training sequence, from the inference chunk length d, the number of future states produced before feeding model predictions back as context. The planning horizon H is the total number of model steps optimized by the planner.

3.1 Encoder

With the JEPA formulation, instead of directly learning the environment dynamics $o_{t+1} = \mathcal{T}(o_t, a_t)$, we first use an encoder E_θ to map each observation to a latent representation $z_t = E_\theta(o_t)$ and learn the dynamics in this latent space. The image observation is encoded by a visual backbone, while proprioceptive information is encoded by a simple MLP; the two encodings are concatenated to form z_t. k-WM is encoder agnostic — the encoder may be a frozen pretrained model or trained end-to-end with the predictor. We discuss the anti-collapse mechanism used when training the encoder in Section 3.3.

3.2 Predictor

A conventional world model predicts one step at a time and recursively applies the transition model to reach longer horizons. In the spirit of modeling the dynamics using parallel computation, we

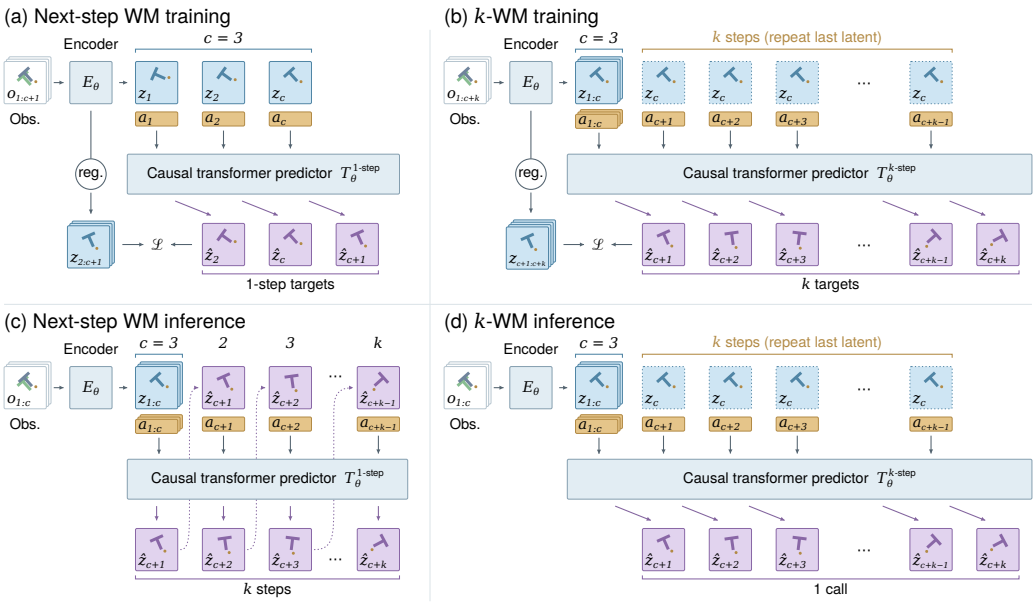

Figure 2: **Architecture.** Left: standard next-step WM training with teacher forcing and autoregressive inference. Right: k-WM, which predicts multiple future latents in a single forward pass during both training and inference. Actions are concatenated with the latent tokens, and the last context latent is repeated to fill the future input positions. The inference process visualized here provides only one ground-truth frame, so the remaining history context must be bootstrapped from model predictions.

instead promote the transition function to predict up to k steps in a single forward pass:

$$(\hat{z}_{t+1}, \ldots, \hat{z}_{t+n}) = T_\theta^{k\text{-step}}(z_t, a_{t:t+n}), \qquad n \leq k, \tag{1}$$

where $a_{t:t+n} = (a_t, \ldots, a_{t+n-1})$. Setting $k = 1$ recovers the conventional next-step predictor.

We implement $T_\theta^{k\text{-step}}$ as a causal transformer over the temporal axis. As is standard, its input begins with a short window of history latents ending in the current latent z_t. Unlike a teacher-forced next-step predictor, whose later input positions contain encoded ground-truth latents, we fill future input positions with copies of the last context latent z_t. Each action is embedded by an MLP and concatenated with the latent at its corresponding position; the future positions therefore share the same latent content but differ in their action and temporal-position information. A causal temporal mask ensures that $\hat{z}_{t+n}$ depends only on the context and on $a_{t:t+n}$, and a single forward pass returns the entire predicted trajectory $\hat{z}_{t+1:t+k}$.

3.3 Training Objectives

To predict over multiple horizons, we also promote the loss to a k-step world-model loss, computed using the k-step transition in Equation (1):

$$\mathcal{L}_{\text{wm}}^{k\text{-step}} = \frac{1}{\sum_{m=1}^{k} f(m)} \sum_{n=1}^{k} f(n) \left\| \hat{z}_{t+n} - z_{t+n} \right\|. \tag{2}$$

Setting $k = 1$ recovers conventional next-step prediction. We study the effect of tuning k in Section 4.6. Here, $f(n)$ is a loss-weighting profile, which may also vary over training to implement a curriculum (see Section A). In our experiments, we take $f(n) = 1$, weighting every prediction horizon uniformly.

To prevent the JEPA objective from collapsing to a trivial constant representation, we optimize $\mathcal{L}_{\text{wm}}^{k\text{-step}} + \lambda \mathcal{L}_{\text{reg}}$, where the anti-collapse regularizer $\mathcal{L}_{\text{reg}}$ and its weight λ depend on the model.

3.4 Inference and Planning

At inference, action sequences longer than k are processed in chunks of at most k, with predictions from one chunk providing context for the next. Autoregressive rollout is chunk $d = 1$.

For planning, given a goal observation o_{goal} and an initial observation o_0, we optimize H model steps of actions. We extract the latents z_{goal} and z_0, roll out z_0 to $\hat{z}_H$, and minimize a simple cost function $\mathcal{C}$ with respect to the action sequence:

$$\mathcal{C}(a_{0:H}) = \|z_{\text{goal}} - \hat{z}_H\|, \qquad \hat{a}_{0:H} = \operatorname*{argmin}_{a_{0:H}} \mathcal{C}(a_{0:H}). \tag{3}$$

We plan open-loop: the model receives only the initial observation (image and proprioception) and the goal, and bootstraps the required history context from its own predictions; the optimized actions are then executed without replanning.

We adopt two planning algorithms for minimizing this cost: the Cross-Entropy Method (CEM) and gradient descent (GD) with Adam. The optimizer settings are given in Section E.2.

4 Experiments

We design our experiments aiming to answer the following questions:

- How does multi-horizon training affect planning success, at short and long horizons?
- How does it affect open-loop rollout quality?
- How does the inference cost of parallel prediction compare to autoregressive rollout?
- How should the training horizon k be chosen?
- What matters in k-WM: the parallel rollout, or the representation the objective induces?

We first describe the world models we build on and the baselines (Section 4.1) and the environments (Section 4.2).

4.1 Instantiations and Baselines

We instantiate k-WM on top of three world model classes and compare each against its next-step counterpart ($k = 1$) with the same encoder, predictor, and training recipe:

a) **DINO-WM** (Zhou et al., 2024): frozen DINOv2 ViT-S patch features, $z_t \in \mathbb{R}^{16\times16\times384}$, with no anti-collapse term. The predictor uses a block-causal mask over frames of patch tokens.
b) **DINO FT**: DINOv2 ViT-S fine-tuned end-to-end, with the CLS feature $z_t \in \mathbb{R}^{1\times384}$ as the latent and a stop-gradient on the training target with an EMA teacher to prevent collapse.
c) **LeWM** (Maes et al., 2026): ViT-Tiny encoder, $z_t \in \mathbb{R}^{1\times192}$, trained end-to-end with SIGReg. Whereas the original LeWM conditions on actions through AdaLN, we concatenate actions with the input latents as in the other instantiations, for both the next-step and the multi-horizon model.

For all instantiations the predictor follows the DINO-WM and LeWM settings (hyperparameters in Section E). We utilize a frameskip of 5 frames across all environments and models, baselines included, and three history context frames. We train with $k = 8$: from a single initial frame, after bootstrapping the history context, one forward pass predicts 10 model steps, or 50 environment steps. We ablate k in Section 4.6.

We additionally compare against current SOTA world models: Temporal Straightening (TS) (Wang et al., 2026), which plans over straightened DINOv2 patch features; and Fast LeWM (Gao & Xu, 2026), which composes actions on top of the LeWM encoder. All baselines use the same proprioception as our models.

4.2 Environments

We evaluate on four environments: a) Push-T, a 2D manipulation task in which an agent pushes a T-shaped block; b) Maze, a 2D navigation task in which the agent navigates to a goal inside a maze; c) Reacher, a 2D task in which a two-joint arm reaches a target joint configuration; and d) Cube, a 3D manipulation task in which a robot arm picks and places a cube. Proprioception and dataset details are given in Section F. The environments are shown in Figure 1.

Method	Objective	CEM				GD				Avg. Δ	
		Push-T	Maze	Cube	Reacher	Push-T	Maze	Cube	Reacher	CEM	GD
k-WM vs. next-step											
DINO-WM	next-step	**83.3** ±1.2	**92.7** ±4.2	51.3 ±7.0	31.3 ±4.6	66.0 ±3.5	40.0 ±6.0	**58.7** ±6.4	13.3 ±3.1		
	k-WM	82.0 ±6.9	86.7 ±5.0	**63.3** ±1.2	**32.0** ±5.3	**70.7** ±8.1	**66.7** ±13.3	56.7 ±7.0	**18.7** ±3.1	+1.3	+8.7
DINO FT	next-step	78.0 ±6.0	62.0 ±7.2	50.7 ±3.1	42.7 ±6.4	63.3 ±4.2	14.7 ±4.2	**48.7** ±7.0	44.0 ±2.0		
	k-WM	**90.0** ±2.0	**86.7** ±2.3	**55.3** ±1.2	**86.0**† ±3.5	**83.3** ±4.6	**58.0** ±9.2	47.3 ±5.0	**82.0**† ±5.3	+21.2	+25.0
LeWM (concat)	next-step	73.3 ±5.8	**89.3** ±2.0	49.3 ±2.5	**49.3** ±5.7	74.7 ±4.2	17.3 ±5.0	48.7 ±6.4	**10.7** ±4.1		
	k-WM	**90.0** ±0.0	82.0 ±5.3	**62.0** ±3.5	48.0 ±4.0	**83.3** ±4.2	**47.3** ±8.1	**61.3** ±3.1	**10.7** ±5.0	+5.3	+12.8
Prior world models											
LeWM (AdaLN)		87.3 ±5.0	79.3 ±9.2	64.0 ±3.5	24.7 ±3.1	83.3 ±4.2	26.7 ±4.2	68.0 ±3.5	6.7 ±3.1		
Fast LeWM		86.7 ±4.1	79.3 ±4.2	58.0 ±9.2	17.3 ±3.1	88.0 ±5.3	60.0 ±0.0	60.7 ±5.0	6.7 ±3.1		
Straightening		85.3 ±8.3	81.3 ±4.2	67.3 ±6.1	8.0 ±3.5	87.3 ±2.3	69.3 ±4.2	74.7 ±1.2	6.0 ±2.0		

Table 1: **Short-horizon planning (H=5).** Success rate (%) with CEM and gradient descent (GD), mean ± std over 3 seeds × 50 episodes. Top: each world model class with k-WM against its next-step version with the same encoder, predictor and training recipe. Bottom: prior world models, re-run with the same planning protocol. **Bold**: better of each pair; pink: best in the column; †: k-WM model rolled out autoregressively (Section 4.7). Avg. Δ: k-WM minus next-step, averaged over environments from unrounded success counts.

4.3 Planning at Short and Long Horizons

Setup. We plan open-loop from a single initial observation as described in Section 3.4. For PushT, Cube, and Maze, the goals are taken from expert trajectories. For Reacher, the goal is sampled uniformly in joint space with zero initial velocity.

Short horizon. As seen in Table 1, multi-horizon training performs on par with or better than its matched next-step baseline across most settings, despite receiving supervision over substantially longer prediction horizons. The improvement is particularly pronounced for the end-to-end trained encoders, with DINO FT improving by 21.2 and 25.0 percentage points on average under CEM and GD, respectively, and LeWM improving by 5.3 and 12.8 points. k-WM is strongest on Push-T and Reacher, while prior methods remain stronger on Maze and Cube. The frozen-encoder DINO-WM setting helps isolate these effects: multi-horizon training yields little average improvement under CEM but improves GD by 8.7 percentage points. This contrast with the larger gains of DINO FT and LeWM suggests that adapting the encoder to the multi-horizon objective is an important component of the improvement. Interestingly, autoregressive rollout of a multi-horizon model can occasionally further improve planning performance; we analyze this effect in Section 4.7.

Long horizon. Long-horizon planning remains particularly challenging for learned world models, as prediction errors compound over repeated rollout and increasingly distort the objective used for action optimization. Figure 3 extends the results from $H = 5$ to $H = 10$, equivalent to 50 environment steps. DINO FT maintains an overall advantage over its next-step counterpart, and multi-horizon LeWM shows a growing advantage over the original AdaLN LeWM as the horizon increases, with GD improvements of 29.3 and 28.7 percentage points on Push-T and Maze at $H = 10$, respectively. Relative to prior world models, performance remains environment-dependent: Straightening is particularly strong on Push-T and Cube, whereas k-WM models show their clearest advantage on Reacher and increasingly at longer horizons. We highlight that k-WM substantially improves GD performance over the next-step baseline, supporting the benefit of multi-horizon prediction for gradient-based planning.

4.4 Open-Loop Rollout Quality

Here we investigate whether models that plan better also roll out more accurately. For a quantitative comparison we train a linear probe from the encoder latent to the ground-truth state on the training dataset. We then perform open-loop world model rollouts on the validation trajectories, extract the predicted state with the same probe, and compute its error against the ground-truth state trajectory. The result is shown in Figure 4. Visualization of a simple decoder on the frozen encoder for a qualitative comparison is presented in Section B.

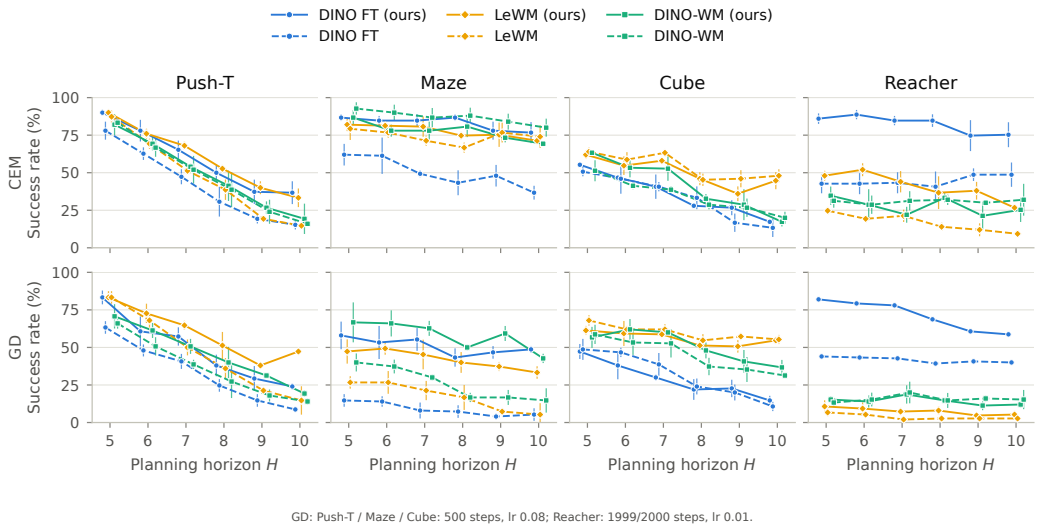

Figure 3: **Long-horizon planning.** Success rate as a function of planning horizon $H \in \{5, \ldots, 10\}$ across four environments, using CEM (top) and gradient-based planning (bottom). Solid lines denote k-WM models; dashed lines denote next-step baselines. DINO FT and DINO-WM use the same encoder and predictor architectures within each pair; the LeWM baseline uses the original AdaLN predictor. Curves show mean success rates, with error bars indicating reported uncertainty. Reacher rollout uses $d = 1$.

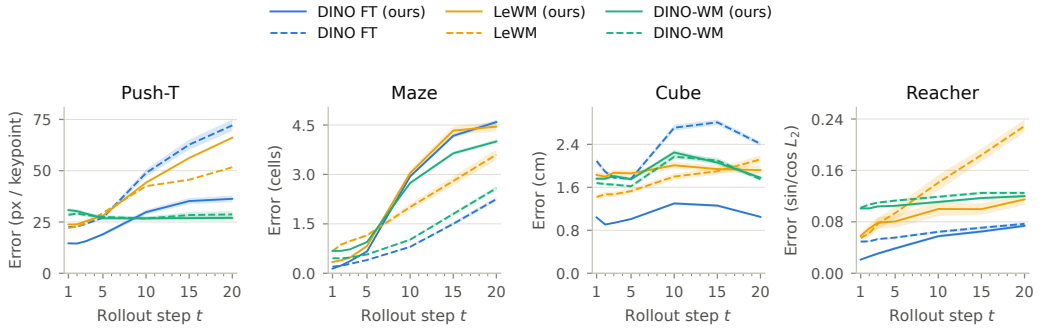

Figure 4: **Open-loop rollout error.** Error between the ground-truth states and states decoded by a linear probe from the predicted latent, vs. rollout step, on validation trajectories (lower is better). Solid: k-WM models rolled out with the largest chunk size; dashed: next-step baselines. The probe is fit per encoder on its training latents. Band: ± 1 standard error over 488 windows for Push-T and 1000 windows for the other environments.

We note that DINO FT and LeWM exhibit slower growth in decoded state error with rollout horizon than their next-step counterparts.

4.5 Inference Cost of Parallel Prediction

By reducing the number of sequential predictor calls, k-WM shows a significant planning speed-up. Here we show the timing gain for both the single-token variant LeWM and the heavier patch variant DINO-WM (Figure 5). CEM is sampling-based and does not use the internals of the world model, whereas GD is the faster alternative and depends strongly on its conditioning. GD is generally faster than CEM, and while our gain for the patch variant is $1.5\times$ at most, the long planning time of that variant makes it valuable. For long-horizon GD planning we see a $3.3\times$ speed-up.

4.6 Choosing the Training Horizon k

How far into the future should the model be trained to predict? Figure 6 highlights two trends. First, planning performance changes systematically as the prediction horizon increases. Second,

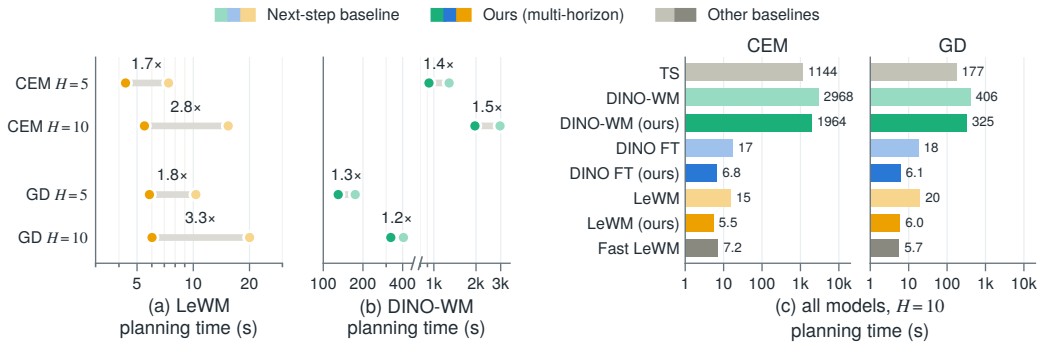

Figure 5: **Planning speed on Push-T.** Planning time is measured in seconds, with compile and bf16, on one L40S. CEM optimizes with 300 samples and 30 iterations; GD optimizes for 500 steps. (a),(b): Speed-up of k-WM over the next-step baseline. (c): Planning time across different models; x-axis is on a logarithmic scale.

different world model classes and planners exhibit different sensitivities to this choice. DINO FT is relatively insensitive to the prediction horizon, whereas LeWM exhibits a substantially sharper response. Reacher is particularly interesting: planning performance increases abruptly at a prediction horizon of five, suggesting threshold-like behavior in the learned representation. We investigate this phenomenon further in Section 4.8.

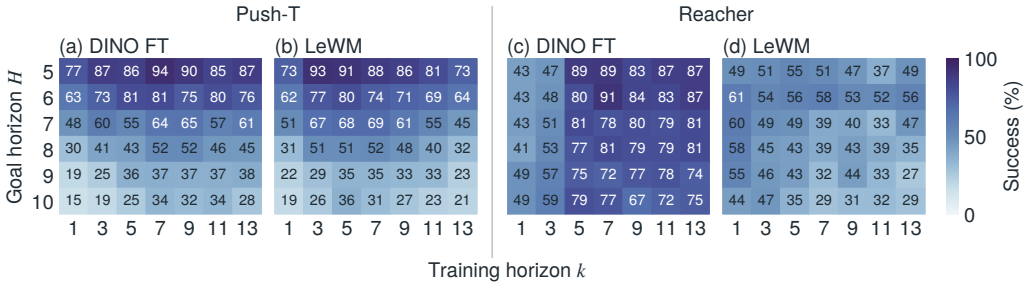

Figure 6: **Training horizon vs. goal horizon.** Planning success rate (%) as a function of the training sequence length (columns) and the planning goal horizon H (rows), for DINO FT and LeWM on Push-T and Reacher, with CEM. Mean over 3 planning seeds $\times$ 50 episodes.

4.7 Separating Multi-Horizon Training from Parallel Inference

To separate the effects of multi-horizon training from parallel rollout, we vary the rollout chunk length at inference and compare against models trained autoregressively. This tests whether the planning gains arise primarily from avoiding repeated model composition or from the long-horizon training objective itself.

Environment	Planner	Rollout chunk length				AR training window		Next-step
		1	2	4	6	10	3	
Push-T	CEM	47.0±9.5	52.7±9.5	**56.0**±5.3	50.0±5.3	48.7±8.3	46.0±8.7	30.7±9.9
	GD	44.7±2.3	50.7±3.1	**53.3**±9.0	38.0±7.2	46.7±10.3	46.7±6.4	24.7±4.2
Reacher	CEM	**84.7**±4.2	80.7±5.0	77.3±7.6	72.0±2.0	76.0±2.0	58.0±9.2	40.7±10.1
	GD	**68.7**±1.2	12.7±2.3	22.0±5.3	60.7±1.2	51.3±13.3	22.0±2.0	39.3±6.4

Table 2: **Rollout and training ablation.** Planning success rate (%, ↑) at H=8 when varying the inference chunk length d or autoregressive (AR) training window (defined in Section A). Bold: highest mean per row.

As shown in Table 2, rollout chunk length has a non-monotonic effect: Push-T peaks at an intermediate chunk size, while Reacher is strongly planner-dependent. More importantly, on Push-T the autoregressively trained models substantially outperform the next-step baseline despite retaining autoregressive inference. Thus, avoiding autoregressive rollout alone cannot explain the gains; the long-horizon training objective itself also contributes. We next examine whether this improvement is reflected in the learned representation.

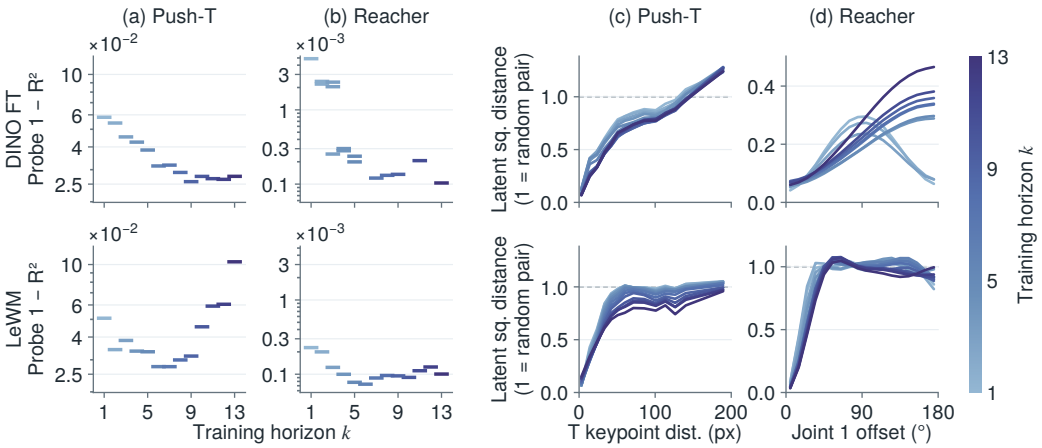

Figure 7: **State alignment of the DINO FT (top) and LeWM (bottom) latents vs. training horizon.** (a, b) Fraction of the ground-truth state variance that cannot be explained by a linear probe from the latent representation ($1 - R^2$, Push-T states: T keypoints, Reacher states: arm pose), with one dash per trained model. (c, d) Latent squared distance, normalized such that the expected distance between a random pair of frames is 1, versus ground-truth state distance for pairs of frames matched on the remaining state variables (agent position on Push-T, second joint on Reacher). Darker curves indicate longer training horizons.

4.8 What the Multi-Horizon Objective Teaches the Encoder

The DINO FT setting lets us isolate how prediction horizon shapes the representation without LeWM's additional latent regularization. As shown in Figure 7, ground-truth state becomes increasingly linearly decodable as the training horizon grows on both Push-T and Reacher. LeWM instead exhibits a U-shaped trend, consistent with its greater sensitivity to training horizon in Figure 6.

We next ask whether latent distances better reflect task progress. We compare distances between latent states with their ground-truth state distances while matching the remaining state variables. On Push-T, this relationship becomes increasingly monotonic with training horizon. Reacher shows a sharper transition: the second joint is well organized even at short horizons, while the first joint is poorly distinguished near 0° and 180° until the horizon increases. Linear-probe error similarly drops by nearly an order of magnitude around horizon three, with different seeds falling on either side of the transition, preceding the jump in planning success around horizon five (Figure 6).

These suggest a threshold-like change in which longer-horizon supervision causes the encoder to retain previously weakly represented state information. In Section C, a linear stochastic model exhibits an analogous transition, providing one possible mechanism for this behavior.

5 Conclusion

In this work, we introduce k-WM, a minimal multi-horizon extension to JEPA world models that lets the predictor output a sequence of future latents in a single forward pass. Across three world model classes and four control environments, k-WM improves planning performance under both sampling-based and gradient-based planners while substantially reducing long-horizon planning time. Our results identify prediction horizon as an important world-model design choice: it affects dynamics prediction, downstream planning performance, and — when the encoder is trained jointly — the information and geometry represented in latent space.

Limitations and Future Work. The phase-transition-like behavior observed on Reacher and in our toy model raises theoretical questions about when and why increasing the prediction horizon changes the representation selected by the learning objective. An additional direction is to study how multi-horizon prediction interacts with different representation regularizers.

AI USE STATEMENT

We used generative AI tools to assist with generating synthetic datasets, developing toy mathematical models for conceptual understanding, proposing and refining experimental hypotheses, designing and providing feedback on experiments, implementing methods, cleaning and reformatting datasets, and interpreting results. We also used these tools for brainstorming, searching for information and relevant literature, summarizing and analyzing prior work, creating figures and other artifacts, writing and editing code, drafting and editing the manuscript, and formatting references. We also used these tools to help formulate and derive mathematical arguments, including the toy-model analysis in the appendix. We did not use generative AI tools to translate text. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including all text, claims, and artifacts produced with AI assistance.

ETHICS STATEMENT

This work studies world models for visual prediction and planning in simulated control environments. Our experiments do not involve human participants or personal data. We do not identify specific ethical concerns arising from the experiments reported in this paper.

REPRODUCIBILITY STATEMENT

All of our datasets, training, and evaluation code will be made publicly available. Videos of our planning results can be seen here: `https://anon-k-wm.github.io`.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629, 2023.

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

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15791–15801. IEEE, 2025.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first international conference on machine learning*, 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Yuntian Gao and Xiangyu Xu. Fast LeWorldModel. *arXiv preprint arXiv:2606.26217*, 2026.

Raktim Gautam Goswami, Amir Bar, David Fan, Tsung-Yen Yang, Gaoyue Zhou, Prashanth Krishnamurthy, Michael Rabbat, Farshad Khorrami, and Yann LeCun. World models can leverage human videos for dexterous manipulation. *arXiv e-prints*, pp. arXiv–2512, 2025.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 6112–6121, 2019.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15706–15734, 2024.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

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 (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565, 2019.

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025.

Yilun Kuang, Yash Dagade, Quentin Le Lidec, Lucas Maes, Randall Balestriero, and Yann LeCun. LpWM: A case for sparse representations in world models. *arXiv preprint arXiv:2608.22764*, 2026.

Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*, 2022.

Yann LeCun. A path towards autonomous machine intelligence. OpenReview, 2022. Version 0.9.2, `https://openreview.net/forum?id=BZ5a1r-kVsf`.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *ArXiv*, abs/2403.03181, 2024. URL `https://api.semanticscholar.org/CorpusID:268248763`.

Andy Zeyi Liu, Haoran Sun, Lucas Baker, Randall Balestriero, and John Sous. Semigroup-JEPA: Latent dynamics consistency for zero-shot physics generalization, 2026.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *International Conference on Learning Representations (ICLR)*, 2023.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorld-Model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Nikhil Mishra, Pieter Abbeel, and Igor Mordatch. Prediction and control with temporal segment models. In *International conference on machine learning*, pp. 2459–2468. PMLR, 2017.

Derrick Nguyen and Bernard Widrow. The truck backer-upper: an example of self-learning in neural networks. *IJCNN*, 1989.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318, 2013.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, 2025.

Pratyaksh Rao, Wancong Zhang, Randall Balestriero, Yann LeCun, and Giuseppe Loianno. Skyjepa: Learning long-horizon world models for zero-shot sim-to-real control of quadrotors. *arXiv preprint arXiv:2606.23444*, 2026.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation, and logarithmic depth. *arXiv preprint arXiv:2402.09268*, 2024.

Vlad Sobal, Jyothir S V, Siddhartha Jalagam, Nicolas Carion, Kyunghyun Cho, and Yann LeCun. Joint embedding predictive architectures focus on slow features. In *NeurIPS 2022 Workshop on Self-Supervised Learning: Theory and Practice*, 2022. arXiv:2211.10831.

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. *arXiv preprint arXiv:2502.14819*, 2025.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. *arXiv preprint arXiv:2603.12231*, 2026.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Gaoyue Zhou, Zichen Jeff Cui, Ada Langford, Bowen Tan, Yann LeCun, and Lerrel Pinto. Patch policy: Efficient embodied control via dense visual representations. *arXiv preprint arXiv:2607.18236*, 2026.

A Loss Weighting Curriculum

The k-step loss of Section 3.3 weights the error at horizon n by $f(n)$, and every result in the main text uses uniform weights, $f(n) = 1$. Here we test whether a non-uniform weighting profile, or a curriculum that changes f over the course of training, performs better. Since only the relative weight assigned to each horizon matters, we summarize each profile by its mean horizon. In our implementation, the loss also includes one-step predictions at the two teacher-forced history positions, which we count as $n = 1$. We therefore define

$$\bar{n}(f) = \frac{2f(1) + \sum_{n=1}^{k} n f(n)}{2f(1) + \sum_{m=1}^{k} f(m)}. \tag{4}$$

With uniform weights, this gives $\bar{n} = 1$ for $k = 1$, $\bar{n} = 2.43$ for $k = 5$, and $\bar{n} = 3.8$ for $k = 8$.

Profiles. All weighted arms keep the $k = 8$ window and change only f. We use four one-parameter profiles:

$$\begin{aligned}
&\text{cutoff:} && f_c(n) = \min\big(1, \max(0, c - n + 1)\big), \quad c \in [1, k], \\
&\text{geometric:} && f_g(n) = g^{n-1}, \quad g \in [0, 1], \\
&\text{power law:} && f_p(n) = n^{-p}, \quad p \geq 0, \\
&\text{far-only:} && f_a(n) = \mathbf{1}[n = 1] + a\,\mathbf{1}[n = k], \quad a \geq 0.
\end{aligned} \tag{5}$$

The cutoff profile with $d = 1$ is exactly the next-step objective, and with integer $d = j$ it is exactly the uniform $k = j$ objective. The geometric and power-law profiles decay smoothly with n, while the far-only profile trains only on the first and last horizons. Each profile is indexed by $\bar{n}$: at every optimization step, we solve by bisection for the parameter (d, g, p, or a) that produces the target $\bar{n}$.

Curricula. A curriculum moves the target linearly with training progress $\tau \in [0, 1]$, the fraction of optimization steps completed:

$$\bar{n}(\tau) = \bar{n}_0 + \tau(\bar{n}_1 - \bar{n}_0). \tag{6}$$

The three *ramps* (cutoff, geometric, and power law) go from $\bar{n}_0 = 1$ to $\bar{n}_1 = 3.8$, i.e. from the next-step objective to the uniform $k = 8$ objective, so they end on the same objective as our baseline. The *reverse cutoff ramp* runs in the opposite direction, from 3.8 to 1, and ends on the next-step objective. The *slow-tempo geometric ramp* instead moves g linearly from 0 to 1; this stays near the next-step objective for longer ($\bar{n} = 1.48$ halfway through training, compared with 2.4 for the ramp that is linear in $\bar{n}$). Finally, two *fixed* profiles hold $\bar{n} = 2.43$, the value of the uniform $k = 5$ objective, for the entire run: one geometric and one far-only.

Setup. We run this study with DINO FT on Push-T, training every arm for two epochs using the recipe from the main experiments. We compare against the uniform $k = 1$ (next-step) and $k = 8$ objectives. For reference, the figure also shows two autoregressively trained arms, which keep uniform f and $k = 8$ but fill each future input position with the model's own previous prediction (one predictor pass per step, with gradients flowing through the fed-back predictions). The predictor attends either to the full window or only to the last three tokens (sliding, as in the DINO-WM rollout). Planning uses CEM (300 samples, 30 iterations, top 30) and GD (Adam, learning rate 0.08, 500 steps) at goal horizons $H \in \{5, 8, 10\}$, with 3 seeds $\times$ 50 episodes each.

Results. Figure 8 shows that no weighting profile or curriculum outperforms uniform weighting. Each of the seven weighted arms falls below the uniform $k = 8$ baseline in all six planner–horizon settings (baseline CEM $0.90/0.50/0.37$ and GD $0.83/0.38/0.24$ at $H = 5/8/10$). The best-performing arm, the cutoff ramp, comes close at $H = 5$ (0.88 CEM) but falls behind at longer

horizons (0.19 versus 0.37 CEM at $H = 10$). Among the ramps, those that spend less of training emphasizing long horizons perform worse: the cutoff, geometric, and power-law ramps reach 0.46, 0.43, and 0.40 at CEM $H = 8$, respectively, while the slow-tempo ramp reaches only 0.29, below the next-step objective's 0.31.

The ordering of the curriculum also matters. The reverse ramp passes through the same profiles as the cutoff ramp and matches it at $H = 5$ (0.88), but because it ends on the next-step objective, it falls to 0.16 and 0.04 at $H = 8$ and 10, respectively, below the plain next-step model (0.31 and 0.15). At a fixed mean horizon, the geometric profile (0.86/0.42/0.25 CEM) performs similarly to the uniform $k = 5$ objective (CEM 0.85/0.45/0.27, Figure 6). The far-only profile, which assigns no weight to horizons $2, \ldots, 7$, fails at every horizon (≤ 0.10 with both planners).

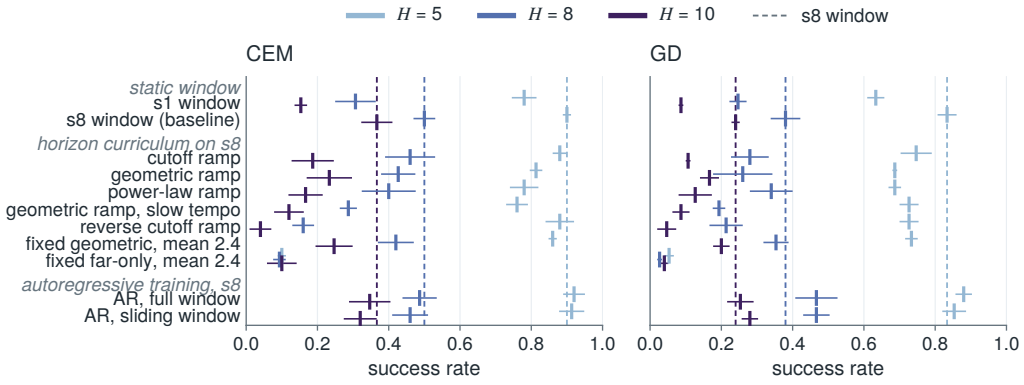

Figure 8: **Loss weighting and curricula on Push-T** (DINO FT, two training epochs). Planning success rate with CEM (left) and GD (right) at goal horizons $H = 5, 8, 10$ (light to dark). Rows: the uniform next-step ($k = 1$, "s1") and $k = 8$ ("s8", baseline) objectives; five curricula and two fixed profiles on the $k = 8$ window (Section A); and two autoregressively trained $k = 8$ models, with a full or three-token sliding attention window. Tick: mean over 3 planning seeds $\times$ 50 episodes; bar: standard error across seeds. Dashed vertical lines: the uniform $k = 8$ baseline at each H.

B Qualitative Rollout Decoding

To visualize imagined rollouts, we train a separate decoder that maps the 384-d DINOv2 CLS embedding back to a 224×224 RGB image. The world model, including its (EMA) image encoder, stays frozen. The decoder has 13M parameters. We train on single frames from the training split for 40k steps with batch size 64, with AdamW. The decoder does not take proprioceptive inputs.

C A Toy Model of the Representation Transition

Let $o_t \in \mathcal{O}$ be observations, $a_t \in \mathcal{A}$ actions, and $\mathcal{T} : \mathcal{O} \times \mathcal{A} \to \mathcal{O}$ the environment dynamics. We learn an encoder $E_\theta : \mathcal{O} \to \mathcal{Z}$ and a latent dynamics model $T_\theta : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$ that agree with the environment, $E_\theta\big(\mathcal{T}(o_t, a_t)\big) = T_\theta\big(E_\theta(o_t), a_t\big)$, together with an anti-collapse term that keeps E_θ non-trivial (LeCun, 2022). Although $\mathcal{T}$ acts one step at a time, we may equally learn an n-step map $T_\theta^{(n)}(z_t, a_t, \ldots, a_{t+n-1}) = z_{t+n}$ that consumes a chunk of actions at once. The following example shows that this choice changes *which features the encoder keeps*, and in particular addresses the concern that JEPA learns slow features (Sobal et al., 2022).

A two-feature model. Consider two independent AR(1) features x_t, y_t with their own action channels, a linear unit-variance encoder with a one-dimensional latent, and the optimal linear n-step predictor:

$$\begin{aligned} x_{t+1} &= \lambda_x x_t + b_x a_t^x + \sigma_x \epsilon_t^x, \\ y_{t+1} &= \lambda_y y_t + b_y a_t^y + \sigma_y \epsilon_t^y, \end{aligned} \quad z_t = \cos\theta\, x_t + \sin\theta\, y_t, \;\; \hat{z}_{t+n} = p\, z_t + \sum_{i=0}^{n-1} \big(q_i^x a_{t+i}^x + q_i^y a_{t+i}^y\big), \tag{7}$$

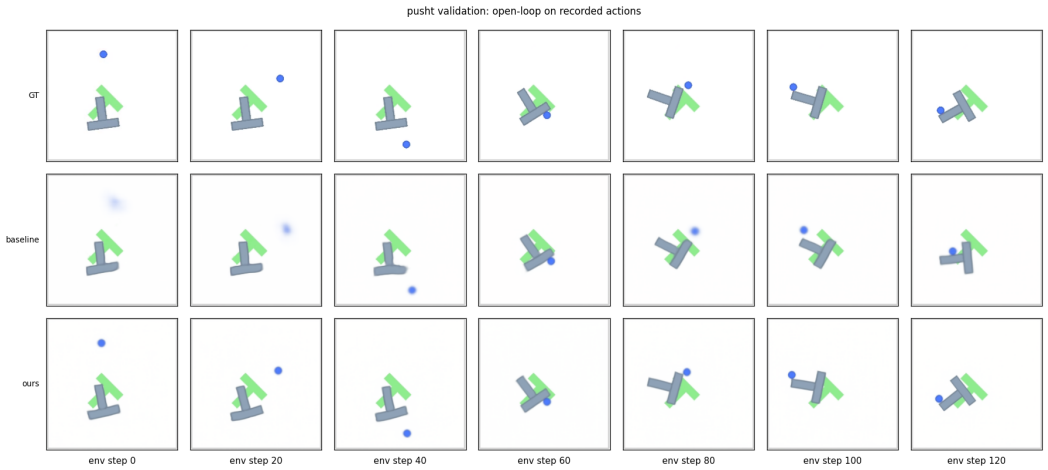

Figure 9: Push-T rollout: Decoded latent of DINO FT with k-WM vs. next-step DINO FT, 100-environment-step rollout.

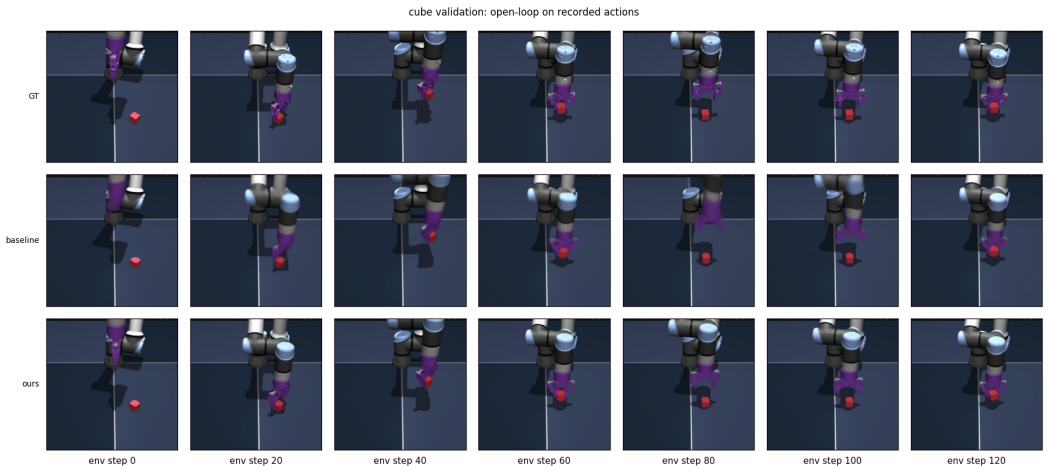

Figure 10: Cube rollout: Decoded latent of DINO FT with k-WM vs. next-step DINO FT, 100-environment-step rollout.

where $a_t^{x,y}, \epsilon_t^{x,y} \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ and $b^2 + \sigma^2 = 1 - \lambda^2$ so that x_t, y_t have unit variance. Each feature is characterized by its memory λ and its *noise fraction* $\eta = \sigma^2/(b^2+\sigma^2)$: $\eta = 0$ is fully action-driven, $\eta = 1$ an uncontrollable distractor. Minimizing over $p, \boldsymbol{q}$ leaves a loss in the encoder angle alone,

$$L^{(n)}(\theta) = \ell_x(n)\cos^2\theta + \ell_y(n)\sin^2\theta + \Delta_n^2 \sin^2\theta\cos^2\theta, \quad \ell(n) = \eta\,(1-\lambda^{2n}),\ \Delta_n = \lambda_x^n - \lambda_y^n. \tag{8}$$

The first two terms are the irreducible error of each feature; the last is a mixing penalty. The encoder always commits to a single feature, $\theta = 0$ (x_t) or $\theta = \pi/2$ (y_t), whichever has the smaller $\ell(n)$. At $n = 1$, $\ell(1) = \sigma^2$, so a slowly drifting feature (high λ) is nearly free whether it is driven by action or noise. As $n \to \infty$, $\ell \to \eta$: long-horizon prediction charges the *fraction* of a feature's motion that the actions do not explain. Moreover, the ratio $\ell_x(n)/\ell_y(n)$ is monotone in n (increasing when $\lambda_x > \lambda_y$), so the preferred feature switches at most once: lengthening the horizon can only move the latent from the low-σ^2 feature toward the low-η one. The analogy with visual control is that agent and object motion is fast on the frame-skip timescale while background and lighting drift slowly (and uncontrollably). For a slow distractor x with memory time $\tau_x = 1/(1-\lambda_x) \gg 1$ and a fast feature y ($\lambda_y^{2n} \approx 0$), we have $\ell_x(n) \approx 2\eta_x n/\tau_x$ for $n \ll \tau_x$, so the switch occurs at $n^\star \approx \eta_y \tau_x/(2\eta_x)$; beyond that, longer horizons change nothing. Averaging the loss uniformly over $n \le k$, as we do in training, preserves this qualitative behavior: the averaged loss is still concave in $\sin^2\theta$, so the encoder still commits to a single feature, and the switch moves to a larger k.

D Baselines

D.1 LeWM

LeWorldModel is an end-to-end trained joint-embedding predictive architecture (JEPA) that utilizes SIGReg to prevent latent collapse by encouraging the full embedding distribution to match an isotropic Gaussian. The LeWM training objective is defined as:

$$\mathcal{L}_{\text{LeWM}} = \mathcal{L}_{\text{pred}} + \lambda \text{SIGReg}(\mathbf{Z}). \tag{9}$$

A complete table of hyperparameters can be found in Section E.

E Hyperparameters

E.1 Predictor Architecture

All predictors are causal transformers. The k-WM and next-step models of a class share the same architecture and differ only in the training sequence length. The full architecture settings are given in Table 3.

Table 3: k-WM hyperparameters.

	DINO-WM	DINO FT	LeWM
Encoder	DINOv2 ViT-S, frozen	DINOv2 ViT-S, fine-tuned	ViT-Tiny, scratch
Tokens per frame	256 (patches)	1 (CLS)	1 (CLS)
Latent dim	384	384	192
Predictor width	404	404	202
Depth	6	6	6
Heads × head dim	16 × 64	16 × 64	16 × 64
MLP	SwiGLU, 1365	SwiGLU, 1365	SwiGLU, 1365
Dropout	0.1	0.1	0.1
History frames	3	3	3
Predictor params	20.1M	20.4M	10.0M

E.2 Planning

Table 4 lists the planner settings, shared by every method including the baselines. Both planners optimize actions in normalized units, and each episode is planned once and executed open-loop.

CEM refits its sampling distribution to the top-k elites at each iteration. GD uses learning rate 0.08 for 500 steps on Push-T, Maze and Cube. On Reacher, 500 steps are not enough at any learning rate we tried, so every method uses learning rate 0.01 for 1999 steps.

Table 4: Planning hyperparameters.

CEM (all environments)		
Samples per iteration	300	
Elites (top-k)	30	
Iterations	30	
GD	500 steps	1999 steps
Used for	Push-T, Maze, Cube	Reacher
Optimizer	Adam	Adam
Learning rate	0.08	0.01
Optimization steps	500	1999
Initialization	zero actions	zero actions
Action noise	std 0.1, every 200 steps	std 0.1, every 200 steps

E.3 LeWM

For LeWM, we select hyperparameters via a grid search over SIGReg (0.09, 0.16, 0.32, 0.48, 0.64, 0.8, 0.96) and training epochs (5, 20). We report the best hyperparameters found for k-WM and the baseline in Table 5. We tune each method independently using the same search space and evaluation budget.

Table 5: Selected hyperparameters for k-WM and LeWM across environments.

	k-WM		LeWM	
Environment	Epochs	SIGReg	Epochs	SIGReg
Push-T	20	0.09	5	0.09
Cube	5	0.64	20	0.09
Maze	5	0.64	5	0.09
Reacher	5	0.09	5	0.09

F Environment & Dataset

Push-T is a 2D continuous rigid-body manipulation task in which the agent (a blue dot) pushes a T-shaped block toward a specified goal configuration. We use the standard dataset from DINO-WM, containing 18,000 episodes that replay expert trajectories with varying levels of noise, with an average length of 196 steps.

Maze is a 2D navigation task in which an agent navigates through a maze. We adopt the D4RL environment with the Large Maze layout (9×12) and port it to MuJoCo 3. The training dataset consists of 500K procedurally generated frames. For rollout and planning, we use a BFS expert dataset consisting of trajectories toward randomly sampled reachable goals.

Reacher is a 2D robot-control task in which a two-joint robotic arm must reach a target joint configuration. We follow the DINO-WM environment setup and generate 1.9M frames with actions sampled uniformly from $[-5, 5]$, with the environment running at 50 Hz. Rollouts use the same distribution as training, and we plan toward randomly sampled goals, since the small configuration space makes nearly all configurations mutually reachable.

OGBench Cube (Cube) is a 3D robot-manipulation task in which a robotic arm with a gripper picks up and places a cube, introduced in OGBench (Park et al., 2025). We follow LeWM, using a dataset

of 1.8M frames from scripted expert trajectories generated with their data-collection procedure. The rollout and planning dataset follows the same distribution as the training data.

Proprioception. Prior world-model work varies in how proprioceptive information is incorporated. To make comparisons across models more consistent, we provide the same proprioceptive information to our models and the corresponding baselines. For Push-T and Cube, we provide the world model with proprioceptive state in addition to visual observations: agent position and velocity for Push-T, and end-effector position and yaw for Cube. For Maze and Reacher, we provide velocity only. Providing absolute position would substantially reduce the role of visual state estimation in these environments, while omitting velocity would leave the state under-specified when rolling out from a single ground-truth frame.