BEYOND THE NEXT STEP: VARIABLE-LENGTH LATENT WORLD MODELS FOR LONG-HORIZON PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models enable planning directly in representation space, but most existing approaches predict only one step into the future. Evaluating long action sequences therefore requires recursive rollout, causing prediction errors to accumulate and making long-horizon planning increasingly unreliable. In this paper, we introduce Variable-length Latent World Models (VLWM), which directly predict future latent states conditioned on action sequences of different lengths within a trained horizon range. A shared action-token interface allows a single predictor to support multiple temporal horizons, while a horizon curriculum progressively incorporates longer prediction targets without sacrificing shorter-horizon accuracy. This formulation further turns prediction length into a planning variable, enabling planners to compose rollouts with either predefined or adaptive chunk lengths. We instantiate VLWM on both LeWM and DINO-WM, covering jointly learned and frozen pretrained visual representations. Across PushT, OGBench-Cube, and TwoRoom, VLWM consistently improves long-horizon planning over one-step latent world models, with larger gains as the goal horizon increases. Further analyses show more reliable trajectory ranking and improved goal-conditioned action ranking on LIBERO. These results demonstrate that variable-length prediction provides a simple and transferable interface for more reliable long-horizon latent planning.

1 INTRODUCTION

World models support visual control by predicting the consequences of candidate actions before execution (Ha & Schmidhuber, 2018; Hafner et al., 2019b). Latent world models perform this prediction in a representation space and use the resulting rollouts to score action plans (Assran et al., 2025; Zhou et al., 2024; Maes et al., 2026). LeWorldModel (LeWM) learns visual representations jointly with latent dynamics, while DINO-WM predicts transitions over frozen pretrained visual features (Maes et al., 2026; Zhou et al., 2024). Their different representation-learning choices provide two settings in which to study a shared planning question: at what temporal resolution should a world model predict the effects of an action sequence?

For a one-step predictor, evaluating an extended action sequence requires repeated model applications. Each predicted state becomes the input to the next transition, allowing errors to propagate through the rollout (Janner et al., 2019). The one-step objective supervises local transitions, whereas goal-conditioned planning evaluates the consequences of a sequence of actions (Hafner et al., 2019b; Zhou et al., 2024). This distinction motivates training the predictor on the endpoints of temporally extended action segments. Such supervision also offers a way to vary the prediction resolution: a planner could evaluate a longer segment in one call or compose shorter predictions when finer temporal resolution is useful.

We introduce **Variable-length Latent World Models (VLWM)**, training one predictor on action segments of length $k \in \{1, \dots, K_{\max}\}$ (Fig. 1). Each example pairs a state history and an action segment with its latent endpoint. Individual action tokens follow the state-history tokens, providing a shared interface across trained horizons. We instantiate this framework on both LeWM and DINO-WM.

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.

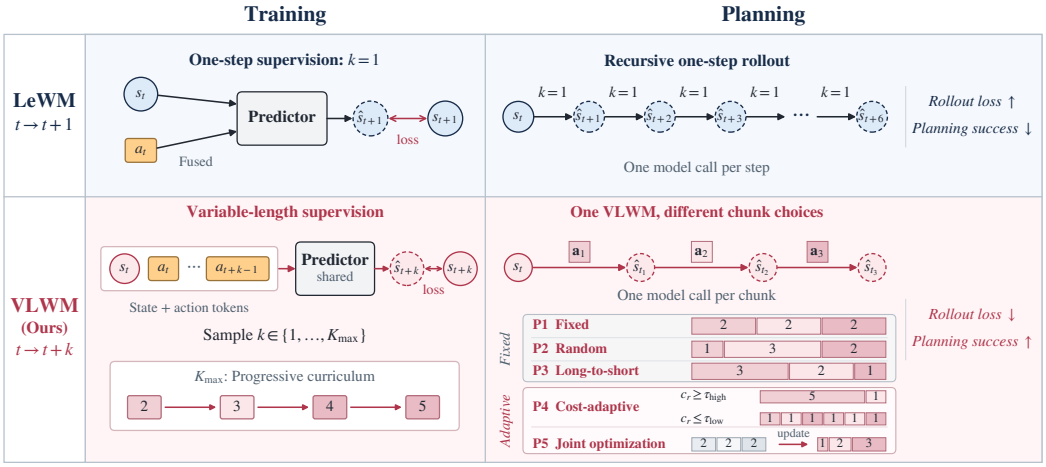

Figure 1: **Overview of VLWM.** LeWM learns one-step transitions; VLWM learns variable-horizon endpoints with action tokens and a curriculum. Colored chunks $\mathbf{a}_i$ end at t_i; strip numbers illustrate partitions of six model steps. P1–P3 use fixed, uniform, or long-to-short rules; P4 uses previous elite cost; P5 jointly optimizes actions and boundaries. All VLWM planners share one predictor, with chunks within the trained range.

Training across horizons mixes prediction tasks of different difficulty. Following curriculum learning (Bengio et al., 2009), we progressively expand the maximum horizon while retaining shorter targets. The schedule improves downstream planning success and reduces local loss and gradient-norm fluctuations in our training diagnostic.

Variable-length prediction also makes a candidate rollout's temporal partition a planning choice. All chunk lengths use the same trained predictor, and their sum covers the chosen planning horizon. We study five ways to select these chunks. Fixed lengths (P1) and uniform sampling (P2) provide reference choices; P3 uses a predefined long-to-short rule for coarse-to-fine prediction. Building on this flexibility, P4 selects chunk lengths using the previous replanning round's final elite cost, while P5 jointly optimizes actions and candidate-specific chunk boundaries with the cross-entropy method (Rubinstein & Kroese, 2004). These strategies adjust the resolution used to predict a plan's consequences within the same latent planning objective.

Across PushT, OGBench-Cube, and TwoRoom with three training seeds, VLWM improves success by 8.1 percentage points with P3 and 8.7 with P5, averaged over both backbones and four goal offsets. Gains reach 11.8 and 12.8 points at the two longer offsets, supporting transfer across learned and pretrained representations. Reference-trajectory analysis finds lower mean prediction-error ranks across all three environments. A separate LIBERO Scene-2 study shows improved goal-conditioned action ranking at $K = 4, 5$ (Sec. 4.2).

Our contributions are:

- A variable-length latent prediction framework that combines multi-horizon endpoint supervision with an action-token interface, allowing one predictor to process action segments across its trained horizon range.
- A cumulative horizon curriculum and planning strategies that exploit the same predictor through predefined chunk schedules, cost-adaptive selection, and joint action–boundary optimization.
- An evaluation on LeWM and DINO-WM across three control benchmarks, with gains that grow at longer goal offsets, supported by curriculum, reference-trajectory ranking, and planner analyses, plus a separate LeWM-based LIBERO study of goal-conditioned action ranking.

2 Related Work

Latent world models for control. Dreamer learns latent dynamics with observation reconstruction (Hafner et al., 2019a), while V-JEPA, V-JEPA 2, DINO-WM, and LeWorldModel predict visual representations without pixel reconstruction (Bardes et al., 2024; Assran et al., 2025; Zhou et al., 2024; Maes et al., 2026). Recent methods improve planning through temporal-progress costs (TD-JEPA) (Bai & Xiong, 2026) or physical-state supervision that shapes representations and their geometry (PhyLatent and SCALE) (Zeng et al., 2026; Hu et al., 2026). VLWM instead focuses on the temporal span of action-conditioned prediction across both learned and pretrained representations.

Direct and multi-horizon prediction. RC-aux combines recursive multi-horizon supervision with budget-conditioned reachability (Li et al., 2026a). DPWM directly predicts endpoint observations from variable-length action sequences using reconstruction supervision, with experiments focused on prediction (Li et al., 2026b). ω-EVA predicts future visual features at sampled horizons from truncated action prefixes within a policy proposal–prediction–refinement loop (Sun et al., 2026). These methods share our interest in temporally extended prediction. VLWM trains one latent predictor on action-token sequences over $1 \leq k \leq K_{\max}$ and composes its predictions into variable-length chunks for goal-conditioned planning, with a horizon curriculum and predefined or optimized chunk boundaries.

Temporal abstraction and hierarchical planning. Jumpy models learn multi-step dynamics alongside skill embeddings (Zhang et al., 2023). HWM supports variable-duration transitions through high- and low-level models, latent macro-actions, and subgoal planning (Zhang et al., 2026). Hi-LeWM studies high-level search support and subgoal execution timing (Caselli et al., 2026); SAGE generates multi-horizon subgoals to guide action proposals (Cheng et al., 2026). VLWM uses one predictor across chunk lengths to evaluate primitive-action sequences, without a separate high-level dynamics model or subgoal generator.

Efficient and adaptive planning. LeFlow and LePlanner amortize planning through latent trajectory generation and learned iterative control, respectively (Huang et al., 2026; Bansal et al., 2026). DREAM-Chunk uses latent feedback to switch among action chunks during execution (Chen et al., 2026), while RISE adapts imagination depth through rollout-or-stop decisions (Lu et al., 2026). VLWM adapts predictive transition durations within a given planning horizon: P4 selects chunk lengths using planning feedback, and P5 jointly optimizes actions and chunk boundaries.

3 Variable-length Latent World Model

3.1 Preliminaries

Latent world models encode observations and predict action-conditioned transitions in representation space (Bardes et al., 2024; Assran et al., 2025; Zhou et al., 2024; Maes et al., 2026). We formalize one-step training and latent-space planning below.

Training objectives of latent world models. We express the one-step prediction setting considered here using an observation encoder f_θ, a backbone-dependent target encoder $\bar{f}_{\bar{\theta}}$, and a predictor g_ϕ. Consider an agent interacting with an environment, producing observations $\{o_t\}_{t=0}^{T}$ and actions $\{a_t\}_{t=0}^{T-1}$. Let $s_t = f_\theta(o_t)$ denote the online latent representation and $e_t = h_\psi(a_t)$ the action embedding. The objective is to predict the latent representation of the next observation conditioned on the current latent state and action:

$$\mathcal{L} \;=\; \mathbb{E}_t\Big[\, D\big(\, g_\phi(s_{t-L+1:t},\, e_t)\,,\; \mathrm{sg}\big(\bar{f}_{\bar{\theta}}(o_{t+1})\big)\,\big)\,\Big] \;+\; \lambda\,\mathcal{R}(s), \tag{1}$$

where $s_{t-L+1:t}$ denotes a history of $L \geq 1$ latent states, D is a distance in latent space, $\mathrm{sg}(\cdot)$ is stop-gradient, and $\mathcal{R}$ is an optional anti-collapse regularizer. Related latent world models differ in encoder construction, prediction objectives, and representation regularization (Bardes et al., 2024; Assran et al., 2025; Zhou et al., 2024; Maes et al., 2026).

Within this formulation, the action enters the model through the predictor as auxiliary conditioning. In some world models (Maes et al., 2026), this is implemented by fusing the action embedding e_t

inside transformer layers through additive conditioning on the hidden states, i.e., the action is wired into the architecture of each block rather than placed in the input sequence itself. This choice is convenient when there is exactly one action per prediction step, but, as we argue next, it tightly couples the model to a fixed action layout.

Planning with latent world models. At each model predictive control (MPC) round, the planner encodes the current observation as $s_t = f_\theta(o_t)$ and the goal as $s_g = f_\theta(o_g)$, then searches for an H-step action sequence $\mathbf{a} = (a_t, \ldots, a_{t+H-1})$:

$$\mathbf{a}^\star = \arg\min_{\mathbf{a}} \|\hat{s}_{t+H}(\mathbf{a}) - s_g\|_2^2. \tag{2}$$

Here $\hat{s}_{t+H}(\mathbf{a})$ is the endpoint predicted under $\mathbf{a}$. We use the Cross-Entropy Method (CEM) for this search. LeWM, V-JEPA-2, and VLWM share this latent-space MPC framework (Maes et al., 2026; Assran et al., 2025); our focus is how the predicted endpoint is generated.

3.2 From One-Step Prediction to Variable-Length Horizon Modeling

The standard formulation above exhibits two coupled limitations: **(i) Single-step supervision.** Eq. 1 supervises only the one-step transition $t \to t{+}1$, resulting in a low-information target for redundant video data and providing no explicit long-horizon training signal, which limits long-horizon planning (Villegas et al., 2017; Zhang et al., 2026). **(ii) Architecture-locked action conditioning.** For world models with architecture-locked action conditioning, actions are fused inside transformer layers through a fixed action layout. Supplying multiple actions through this layout compresses them into shared hidden representations, making their individual contributions to the predicted dynamics difficult to disentangle. The conditioning architecture thus dictates how many actions can be supplied per prediction. In the following, we introduce the Variable-length Latent World Model (VLWM), which directly addresses these two limitations.

Variable-length supervision for latent prediction. To supervise horizons beyond the next step, we train a single predictor g_ϕ to forecast the latent at a sampled future offset $k \in \{1, \ldots, K_{\max}\}$. Given the recent latent history and the action segment driving the agent forward, our objective is

$$\mathcal{L}_{\text{VLWM}} = \mathbb{E}_{t,\, k\sim p(k)} \Big\| g_\phi\big(s_{t-L+1:t},\, a_{t:t+k-1}\big) - \mathrm{sg}\big(\bar{f}_{\bar{\theta}}(o_{t+k})\big) \Big\|_2^2, \tag{3}$$

where $s_t = f_\theta(o_t)$, $p(k)$ is a distribution supported on the trained horizon range $\{1, \ldots, K_{\max}\}$ (specified in Sec. 3.3), and $a_{t:t+k-1}$ is the variable-length action segment of length k that drives the latent from time t to time $t{+}k$. Equation 3 replaces the prediction term while retaining the backbone-specific regularization and target-encoder settings. We note that the current world model is a special case of Eq. 3 with $p(k) = \delta_{k=1}$. In the next part, we will present the architectural modifications to latent world models that enable this objective.

Variable-length action conditioning via tokenization. Our variable-length interface replaces layer-internal action conditioning with action tokens. We embed each action with a shared head $e_\tau = h_\psi(a_\tau)$ and treat e_τ as a **first-class transformer token**, placed alongside state tokens in a single input sequence:

$$\mathbf{X} \;=\; \big[\, s_{t-L+1},\; \ldots,\; s_{t-1},\; s_t,\; e_t,\; e_{t+1},\; \ldots,\; e_{t+k-1} \,\big]. \tag{4}$$

The causal transformer processes $L \geq 1$ state-history tokens followed by k action tokens and reads out $\hat{s}_{t+k}$ from the final token. Type and position embeddings distinguish the inputs (App. C). The same predictor accepts action segments across $1 \leq k \leq K_{\max}$ without architectural changes.

Consequently, VLWM combines three properties: (1) supervision across the trained horizons $1 \leq k \leq K_{\max}$, (2) variable-length action conditioning by construction, and (3) backbone agnosticism—the encoder f_θ can be a JEPA-style, DINO-style, or any other latent encoder, and prior fixed-horizon world models are recovered as special cases of $p(k)$.

3.3 A Curriculum Training Strategy of VLWM

Training Eq. 3 over a broad range of action-sequence lengths from the outset mixes prediction tasks of different difficulty. We adopt a **curriculum** strategy that first develops shorter-horizon dynamics

and progressively introduces longer-range targets. App. E examines the resulting loss and gradient-norm fluctuations.

Let T denote the number of curriculum training steps and $K_{\max} \geq 2$ the maximum prediction horizon. We specify stage endpoints $0 < T_2 < \cdots < T_{K_{\max}} = T$ and progressively expand the active maximum horizon from 2 to $K_{\max}$:

$$j(\tau) \;=\; 2 + \sum_{m=2}^{K_{\max}-1} \mathbb{1}[\tau > T_m], \qquad \tau \in \{1, \ldots, T\}. \tag{5}$$

At training step τ, we sample the prediction horizon k **uniformly** from $\{1, \ldots, j(\tau)\}$:

$$p_\tau(k) \;=\; \frac{1}{j(\tau)} \mathbb{1}\big[\, 1 \leq k \leq j(\tau) \,\big]. \tag{6}$$

The first stage therefore learns both one- and two-step prediction. With $K_{\max} = 5$, the upper horizon follows $2 \to 3 \to 4 \to 5$, with stage durations specified in App. C. This cumulative-uniform schedule continually revisits shorter horizons as longer ones are introduced, and retains full-range sampling after the final expansion. We evaluate its downstream benefit in Sec. 4.3 and analyze training dynamics in App. E.

3.4 Planning with Variable-length Latent World Models

Within the receding-horizon MPC framework introduced in Sec. 3.1 (Eq. 2), our planner varies the temporal resolution used to compute $\hat{s}_{t+H}(\mathbf{a})$. We partition the planning horizon H into contiguous chunks $H = k_1 + k_2 + \cdots + k_M$ and evaluate the rollout as a sequence of **variable-length latent jumps**:

$$s_t \xrightarrow{k_1} \hat{s}_{t+k_1} \xrightarrow{k_2} \hat{s}_{t+k_1+k_2} \xrightarrow{k_3} \cdots \xrightarrow{k_M} \hat{s}_{t+H}, \tag{7}$$

i.e., the predictor takes the current latent and the full k_i-step action segment and directly predicts the latent k_i steps ahead in a single call (cf. Eq. 3). Standard one-step rollout is recovered as the special case $k_1 = \cdots = k_M = 1$. Larger k_i leverage the multi-horizon supervision introduced in Sec. 3.2 and reduce the number of recursive applications of g_ϕ. Different chunking schedules k_i thus induce different planners built on top of the **same** VLWM:

(P1) Fixed-length chunking. P1 partitions each candidate rollout using a constant chunk length $k^\star$, with the last chunk shortened to cover the remaining horizon. We use $k^\star = K_{\max} = 5$ in our evaluation.

(P2) Uniform chunk sampling. P2 samples each chunk length uniformly from the feasible range $\{1, \ldots, \min(K_{\max}, H - u)\}$, where u is the number of model steps already covered. This provides a simple stochastic baseline for selecting the prediction resolution.

(P3) Long-to-short. P3 realizes a predefined coarse-to-fine sampling rule within each rollout. A position-dependent softmax favors large chunks early to capture long-range structure and small chunks near the predicted terminal state for finer refinement. A single bias-strength parameter controls this preference.

Adaptive horizon selection. Variable-length training makes every chunk length within $1 \leq k \leq K_{\max}$ available to the same predictor. Beyond P3's predefined coarse-to-fine rule, this interface allows the prediction resolution to respond to planning feedback or to be optimized for individual candidate trajectories.

(P4) Cost-adaptive chunking. While P3 uses rollout position to select the prediction scale, P4 uses the previous replanning round's final elite cost. A low cost selects one-step chunks for precise prediction, a high cost selects the largest feasible chunk, and intermediate costs interpolate between these choices. The cost signal is held fixed during the next solve, while each chunk is selected subject to the remaining horizon.

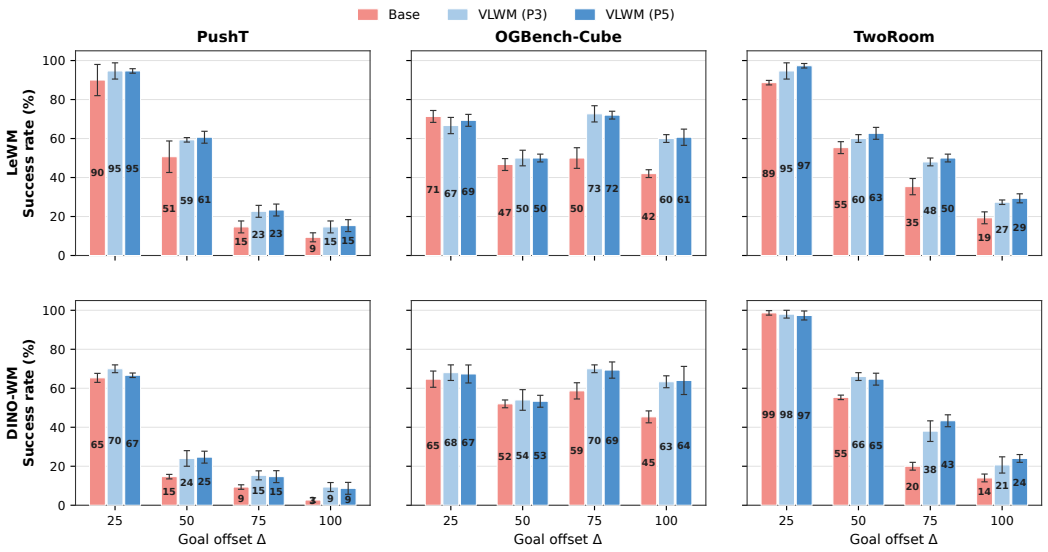

Figure 2: Goal-conditioned planning success rate on PushT, OGBench-Cube, and TwoRoom across goal offsets $\Delta \in \{25, 50, 75, 100\}$. Rows correspond to the LeWM and DINO-WM backbones. P3 denotes the predefined fixed-schedule planner, while P5 jointly optimizes actions and chunk boundaries. Bars and error bars report the mean and standard deviation over three training seeds.

(P5) Joint action–chunk optimization. P4 uses a shared cost signal, while different candidate trajectories may benefit from different temporal partitions. P5 therefore optimizes action sequences together with binary chunk-boundary variables. Each candidate is $\mathbf{z} = (\mathbf{a}_{1:H}, \mathbf{b})$, where $\mathbf{b} \in \{0,1\}^{H-1}$ induces a partition $\pi(\mathbf{b})$ with chunk lengths $1 \le k_i \le K_{\max}$, and the objective is

$$\min_{\mathbf{a}_{1:H},\mathbf{b}} \left\| \hat{s}_{t+H}\big(\mathbf{a}_{1:H}, \pi(\mathbf{b})\big) - s_g \right\|_2^2 . \tag{8}$$

Each CEM iteration samples actions from a Gaussian distribution and boundaries from Bernoulli distributions, scores the resulting pairs, and updates both distributions from the same elite set. The joint update adapts both the action plan and its prediction resolution.

All five planners share the trained VLWM. P1–P3 use fixed or predefined chunk-selection rules, P4 adapts chunk lengths using replanning feedback, and P5 searches over candidate-specific partitions. Detailed sampling rules and solver updates are provided in App. D.

4 Experiments

4.1 Experimental Setup

Environments. We evaluate VLWM on four image-based continuous-control benchmarks. **PushT** requires a point agent to push a T-shaped block toward a target configuration, **OGBench-Cube** evaluates 3D robotic pick-and-place control, and **TwoRoom** requires an agent to navigate through a narrow doorway to a goal in another room (Zhou et al., 2024; Park et al., 2025; Sobal et al., 2025). On these three benchmarks, we construct goal-conditioned tasks by sampling a future observation at offset $\Delta \in \{25, 50, 75, 100\}$ from a held-out trajectory. We additionally study goal-conditioned action ranking on **LIBERO Scene-2**, a contact-rich 7-DoF manipulation setting derived from LIBERO (Liu et al., 2023). We report this model-scoring evaluation separately from control success on the first three benchmarks. Dataset statistics, task definitions, and data-collection details are provided in App. A.

Base world models. We instantiate VLWM on **LeWM**, which jointly learns visual representations and dynamics, and **DINO-WM**, which predicts over frozen pretrained DINO features (Maes et al., 2026; Zhou et al., 2024). Each original one-step model serves as its baseline. VLWM uses Eq. 3

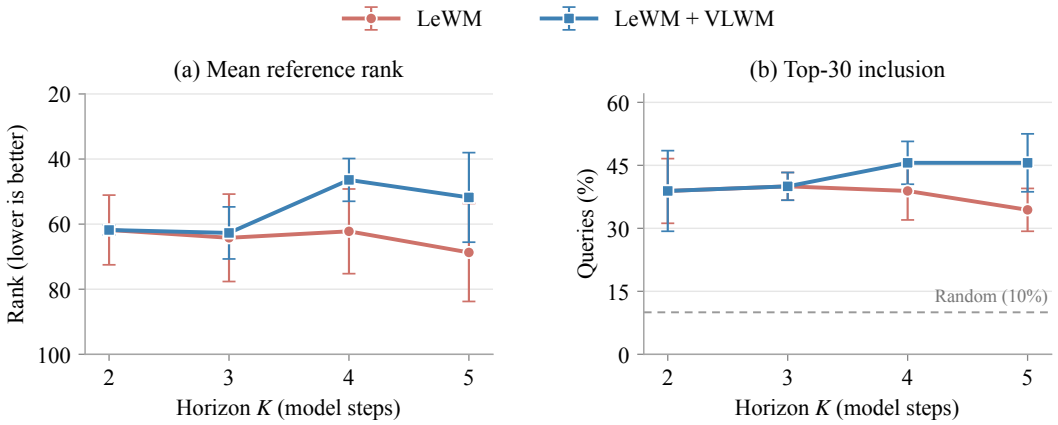

Figure 3: LIBERO Scene-2 goal-conditioned action ranking at $K = 2, \ldots, 5$ (10–25 environment steps): recursive LeWM versus direct LeWM+VLWM. Points and error bars show mean and standard deviation over three independently trained checkpoints, each evaluated on 30 held-out queries with 300 candidates. (a) Mean reference rank (inverted axis). (b) Top-30 inclusion, with random inclusion at 10%.

with either P3 (*fixed-schedule planning*) or P5 (*adaptive planning*). Backbone-specific components, capacity, training data, and training budgets are matched within each pair. Full architecture and optimization settings are in App. C.

Planning and evaluation protocol. On the three control benchmarks, CEM uses 300 candidates, 30 elites, and an execution budget of 2Δ raw steps. The baselines and P1–P4 use 30 iterations; P5 uses 20 to offset candidate-specific partition overhead. Each main method uses three training seeds and the same initial–goal pairs; success follows the environment criterion and is reported as mean $\pm$ standard deviation across seeds. All hyperparameters are selected on validation data. LIBERO uses three independently trained checkpoints per method, each evaluated on 30 held-out queries. Full protocols are in Apps. B and G.

4.2 Main Results

Long-horizon goal-conditioned planning. We compare LeWM, LeWM+VLWM, DINO-WM, and DINO-WM+VLWM on PushT, OGBench-Cube, and TwoRoom across $\Delta \in \{25, 50, 75, 100\}$. For each VLWM instantiation, we report P3 as the fixed-schedule planner and P5 as the adaptive planner; these choices remain fixed across test offsets, with no per-cell selection. As shown in Fig. 2, VLWM improves both backbones, with the gains becoming more pronounced as the goal offset increases. Averaged over all datasets, offsets, and backbones, VLWM improves the corresponding one-step baseline by 8.1 percentage points with P3 and by 8.7 points with P5. At the longer offsets $\Delta \in \{75, 100\}$, these gains increase to 11.8 and 12.8 points, respectively. Both planners improve over the corresponding baseline in every dataset–backbone pair at these two offsets, with gains of up to 23.3 points. This widening advantage supports our motivation that direct variable-horizon prediction is especially useful when one-step models must be recursively rolled out over long horizons.

The improvement transfers across representation-learning settings: VLWM+P5 improves LeWM by 9.3 points and DINO-WM by 8.1 points on average. These results support VLWM across jointly learned visual encoders and pretrained DINO features. P5 provides a modest additional gain over P3 on average, particularly on long-horizon TwoRoom, while their similar performance in several settings suggests that much of the improvement already comes from variable-length training and coarse-to-fine rollout.

Goal-conditioned action ranking on LIBERO Scene-2. We evaluate how the learned goal-conditioned cost prioritizes reference actions in contact-rich manipulation. Each query contains one reference and 299 distractor action sequences sharing an initial observation and a visual goal

Table 1: Effect of the horizon curriculum on LeWM+VLWM with P3 at $\Delta = 100$. Values are success rate (%) over three training seeds. The best mean in each column is bold.

Training schedule	PushT	OGBench-Cube	TwoRoom	Average
No curriculum	12.0±2.0	54.0±2.0	20.7±3.1	28.9
Curriculum	**14.7±3.1**	**60.0±2.0**	**27.3±1.2**	**34.0**

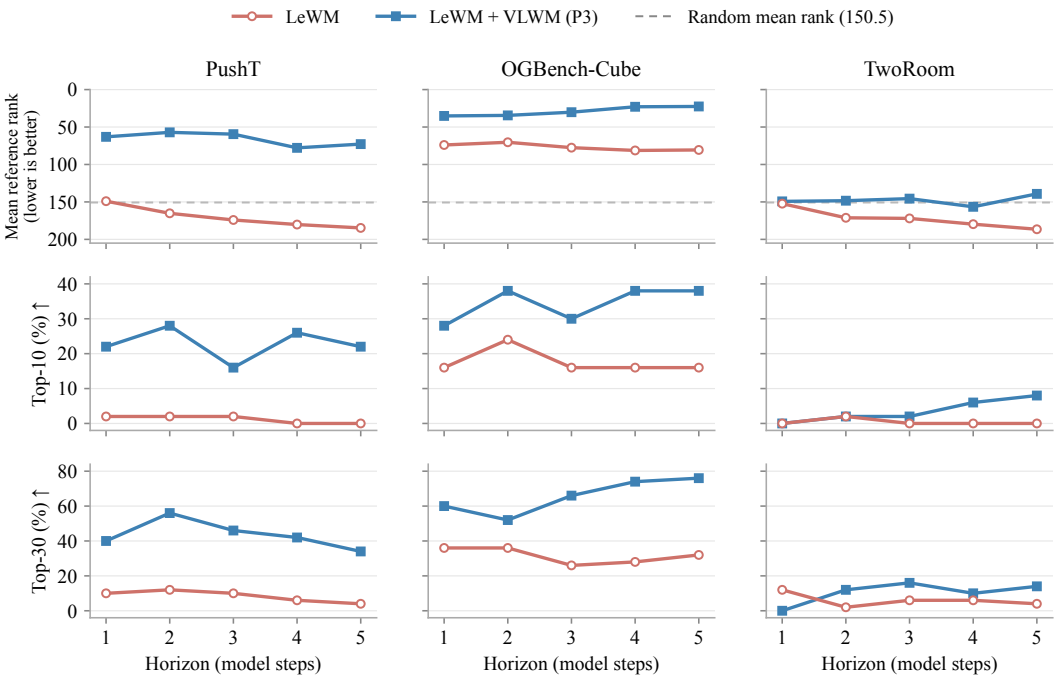

Figure 4: Reference-trajectory ranking across three environments. Rows show mean rank (inverted axis), Top-10, and Top-30 inclusion; higher is better in every panel. Each method uses one checkpoint and 50 queries with 300 candidates per horizon. One model step spans five environment steps; the dashed line marks random mean rank (150.5).

at the reference endpoint. LeWM recursively predicts one step at a time, while LeWM+VLWM directly predicts the K-step endpoint for $K = 2, \ldots, 5$ (10–25 environment steps). Candidates are ranked by latent MSE to the shared goal. Each method uses three independently trained checkpoints, each evaluated on 30 held-out queries; Fig. 3 reports mean reference rank and Top-30 inclusion for multi-step horizons $K = 2, \ldots, 5$.

LeWM has a lower mean rank at $K = 1$, and the methods have similar mean ranks and identical Top-30 inclusion at $K = 2, 3$. VLWM's advantage emerges at $K = 4, 5$, where it achieves lower mean ranks and higher Top-30 inclusion. At $K = 5$, VLWM lowers mean rank from 68.70 to 51.79 and raises Top-30 inclusion from 34.4% to 45.6%. These results support the use of variable-length prediction to prioritize reference actions over longer segments. Full results for $K = 1, \ldots, 5$ and protocol details are provided in App. G.

4.3 Analysis

Effect of curriculum training. We evaluate the horizon curriculum using LeWM+VLWM with P3 at the longest goal offset $\Delta = 100$, comparing progressive horizon sampling with full-range sampling from the start. Across three training seeds, curriculum training improves average success by 5.1 percentage points, with gains on all three benchmarks (Table 1). A separate PushT training diagnostic in App. E shows smaller early prediction-loss fluctuations and reduced local loss and gradient-norm fluctuations during full-range training under the curriculum schedule.

Table 2: LeWM and VLWM planners at $\Delta = 100$. Success is mean $\pm$ standard deviation over three training seeds. Depth is per candidate (P2/P3: expected depth at $H = 5$); time is per CEM iteration for 50 batched environments, averaged across benchmarks. Accounting and iteration budgets are in App. D. Best success, lowest depth, and shortest time are bold, including ties.

Method	PushT	OGBench-Cube	TwoRoom	Avg.	Pred./cand.	Time/CEM iter (s)
LeWM	9.3±2.3	42.0±2.0	19.3±3.1	23.6	5.0	11.58
Fixed (P1: fixed length; P2: uniform sampling; P3: long-to-short)						
P1	14.0±2.0	54.0±2.0	20.7±3.1	29.6	**1.0**	11.22
P2	12.0±2.0	51.3±4.2	22.7±1.2	28.7	~ 2.28	12.38
P3	14.7±3.1	60.0±2.0	27.3±1.2	34.0	~ 1.48	12.15
Adaptive (P4: cost-adaptive chunks; P5: joint action–chunk optimization)						
P4	**15.3±3.1**	58.0±2.0	26.0±2.0	33.1	$\sim$ 2–3	10.78
P5	**15.3±3.1**	**60.7±4.2**	**29.3±2.3**	**35.1**	~ 3.0	**10.72**

Long-horizon trajectory ranking. We examine the relative prediction-error rank of a reference trajectory. Each query contains one reference and 299 distractor action sequences sharing an initial state and goal. We replay each sequence in the simulator and rank candidates by the MSE between their predicted endpoints and their own encoded ground-truth endpoints. We compare recursive LeWM rollouts with LeWM+VLWM using P3 over 1–5 model steps (5–25 environment steps), using one training checkpoint per method and 50 queries per horizon on each environment.

Figure 4 (top) shows that VLWM achieves a lower mean reference rank at every evaluated horizon on all three environments. At five model steps, VLWM lowers mean rank from 184.7 to 72.8 on PushT, from 80.6 to 22.5 on OGBench-Cube, and from 186.5 to 139.2 on TwoRoom. The middle and bottom rows report Top-10 and Top-30 inclusion: the percentage of queries whose reference ranks among the 10 or 30 lowest-error candidates. On PushT and Cube, VLWM improves both metrics at every horizon. At five model steps, Top-30 inclusion rises from 4% to 34%, 32% to 76%, and 4% to 14% on PushT, Cube, and TwoRoom, respectively; on TwoRoom, Top-10 inclusion also reaches 8% versus 0% for LeWM. Together, these curves show that VLWM places reference trajectories among the lower-error candidates more consistently across prediction horizons. Complete numerical results, including Top-1, are provided in App. F.

Comparison of variable-length planners. We compare LeWM with P1–P5 at $\Delta = 100$ across the three control benchmarks; all VLWM planners share weights and evaluation episodes. P5 achieves the highest average success (35.1%), leading on Cube and TwoRoom and tying P4 on PushT (Table 2). P3 leads predefined schedules at 34.0%, compared with 29.6% for P1 and 28.7% for P2. Against LeWM's 23.6%, P5 gains 11.6 percentage points. These results show that chunk selection contributes alongside variable-length training. The table also reports per-candidate predictor depth and amortized time per CEM iteration; App. D specifies the accounting and iteration budgets.

5 Conclusion

VLWM combines variable-length endpoint supervision, a shared action-token interface, and a progressive horizon curriculum. One predictor supports predefined and adaptive chunk planning within its trained range. Experiments on LeWM and DINO-WM demonstrate transfer across jointly learned and frozen representations, with larger gains at longer goal offsets. Curriculum, trajectory-ranking, and planner analyses support the design, while LIBERO goal-conditioned ranking improves at four and five model steps. These findings motivate predictive transition length as a choice for both training and planning, and further study of feedback-driven temporal resolution.

AI USE STATEMENT

We used generative AI tools for drafting manuscript text and language editing. We have **not** used generative AI to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis or interpret results. AI-assisted drafts were revised through author feedback. The authors made the final decisions on the methodology, experiments, and presentation, and take responsibility for the final content of this work.

ETHICS STATEMENT

This work evaluates latent world models in simulated control environments. The reported experiments concern prediction and planning in these settings.

REPRODUCIBILITY STATEMENT

Sections 3.2–3.4 describe the prediction objective, curriculum, and planners. Section 4.1 specifies the evaluation protocols, and Section 4.3 reports the ablations and diagnostic comparisons. Appendices A and B provide dataset and task definitions and control protocols. Appendix C details the LeWM and DINO-WM instantiations, architectures, and optimization settings; Appendix D specifies planner configurations and updates. Appendices E–G provide training diagnostics and the trajectory- and action-ranking procedures and results.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Jiaxin Bai and Jiaxuan Xiong. Temporal-Distance JEPA: Plan-Aware Representation Learning for Latent World Model Predictive Control. *arXiv preprint arXiv:2607.25337*, 2026.

Saksham Bansal, Om Naphade, Chayan Aggarwal, and Vrishin M. LePlanner: An Iterative Amortized Controller For World Models. *arXiv preprint arXiv:2609.13845*, 2026.

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. *arXiv preprint arXiv:2404.08471*, 2024.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

Niccolò Caselli, Francesco Massafra, Samuele Punzo, Salvatore Lo Sardo, Ippokratis Pantelidis, and Sathya Kamesh Bhethanabhotla. Mind the Gap: Promises and Pitfalls of Hierarchical Planning in LeWorldModel. *arXiv preprint arXiv:2607.12547*, 2026.

Wenxi Chen, Kaidi Zhang, Chi Lin, Zhiyuan Zhang, Yu She, Yuejiang Liu, Raymond A. Yeh, Shaoshuai Mou, and Yan Gu. DREAM-Chunk: Reactive Action Chunking with Latent World Model. *arXiv preprint arXiv:2606.18589*, 2026.

Letian Cheng, Qi Zhang, and Yisen Wang. SAGE: Subgoal-Conditioned Action Generation for Latent World Model Planning. *arXiv preprint arXiv:2607.17973*, 2026.

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 *ICML*, 2019b.

Jiaming Hu, Yan Zheng, and Tian Wang. SCALE: State-Calibrated Latent Embeddings for JEPA Planning in the Right Geometry. *arXiv preprint arXiv:2608.16287*, 2026.

Hsiang-Wei Huang, Jianxu Shangguan, Junbin Lu, and Jenq-Neng Hwang. LeFlow: Generative Latent Flow Planning for World Models. *arXiv preprint arXiv:2608.24855*, 2026.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *NeurIPS*, 2019.

Wenyuan Li, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Predictive but Not Plannable: RC-aux for Latent World Models. *arXiv preprint arXiv:2605.07278*, 2026a.

Xinyi Li, Zaishuo Xia, Chenjie Hao, and Yubei Chen. Beyond Myopic World Models: Long-Horizon End-to-End Training for Direct Future Prediction. *arXiv preprint arXiv:2608.07420*, 2026b.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Hongbo Lu, Liang Yao, Chenghao He, Hao Han, Fan Liu, Wenlong Liao, Tao He, and Pai Peng. RISE: Adaptive Imagination for World Action Models. *arXiv preprint arXiv:2608.20430*, 2026.

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.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *ICLR*, 2025.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2004.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Stress-testing offline reward-free reinforcement learning: A case for planning with latent dynamics models. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025.

Zhenguo Sun, Yu Sun, Hande Huang, and Alois Knoll. ω-EVA: Envision, Verify, and Act with Latent Interactive World Models. *arXiv preprint arXiv:2606.09457*, 2026.

Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. In *ICML*, 2017.

Xi Zeng, Haojie Ren, and Ziying Song. PhyLatent: Learning Dynamics-Relevant Representations for JEPA World Models. *arXiv preprint arXiv:2608.05720*, 2026.

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. *arXiv preprint arXiv:2302.12617*, 2023.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, et al. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026.

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.

A Environment and Dataset Details

We use three controlled goal-offset environments following LeWM (Maes et al., 2026) and a separate LIBERO Scene-2 action-ranking evaluation. This section records the full dataset and task definitions.

TwoRoom. TwoRoom is a simple continuous 2D navigation task introduced by (Sobal et al., 2025). The environment consists of two rooms separated by a wall with a single door connecting them. The agent (a red dot) must navigate from a random starting position in one room to a randomly sampled target location in the other room, which requires passing through the door. We collect $10{,}000$ episodes with an average trajectory length of 92 steps. The data are generated using a simple noisy heuristic policy that first directs the agent toward the door along a straight-line path and then toward the target location once the agent has crossed into the other room. Each world model is trained on this dataset for 10 epochs, matching the LeWM reference setup.

PushT. PushT is a continuous 2D manipulation task in which an agent (a blue dot) must push a T-shaped block to match a target configuration, with interactions restricted to pushing actions. We follow the same setup and dataset as (Zhou et al., 2024), which contains $20{,}000$ expert episodes with an average length of 196 steps. Each world model is trained on this dataset for 10 epochs, matching the LeWM reference setup.

OGBench-Cube. OGBench-Cube is a continuous 3D robotic manipulation task in which a robotic arm with an end-effector must pick up a cube and place it at a target location. Originally introduced by (Park et al., 2025), we consider only the single-cube variant. We collect $10{,}000$ episodes, each consisting of 200 steps, using the data-collection heuristic provided in the benchmark library. Each world model is trained on this dataset for 10 epochs, matching the LeWM reference setup.

LIBERO Scene-2. The LIBERO Living Room Scene 2 data contain contact-rich tasks in which a 7-DoF robotic controller places alphabet soup, tomato sauce, butter, milk, or orange juice into a basket (Liu et al., 2023). Our main LIBERO evaluation ranks candidate action sequences against visual goals sampled from reference trajectories. Five consecutive raw actions form one 35-dimensional action block, so one model step corresponds to five environment steps. The ranking protocol is detailed in App. G.

B Control and Success Criteria

For PushT, TwoRoom, and OGBench-Cube, we follow the goal-conditioned control protocol of LeWM (Maes et al., 2026). Control performance is measured by two parameters: the *evaluation budget*, i.e., the maximum number of actions the agent is allowed to execute in the environment, and the *goal offset* Δ, i.e., how many timesteps in the future the goal state is sampled relative to the initial state. During evaluation, trajectories are sampled from the offline dataset; the initial state is randomly sampled from a held-out trajectory and the goal state corresponds to a state occurring Δ timesteps later in the same trajectory, which ensures that the goal is reachable and consistent with the dataset dynamics.

For PushT, TwoRoom, and OGBench-Cube, goal offsets $\Delta \in \{25, 50, 75, 100\}$ use execution budgets $B = 2\Delta$ raw steps, respectively 50, 100, 150, and 200. Each plan spans 25 raw steps and is fully executed before replanning. Success follows the environment-specific ground-truth criterion: block configuration for PushT, end-effector/cube distance for OGBench-Cube, and agent-to-goal distance for TwoRoom (Maes et al., 2026).

C Implementation Details

For PushT, OGBench-Cube, and TwoRoom, both VLWM instantiations share the training and planning settings below, following the LeWM implementation (Maes et al., 2026). Their visual encoders and state representations are specified separately.

Common training setup. Each frame is rendered at 224×224 resolution. We apply a frame-skip of 5, grouping consecutive actions between frames into a single action block. We train with AdamW (learning rate $5{\times}10^{-5}$, weight decay $1{\times}10^{-3}$), batch size 128, gradient clipping at norm 1.0, and bfloat16 mixed precision. Although the optimiser is configured to run for up to 100 epochs, we always evaluate the checkpoint at the end of epoch 10, which we empirically find to be sufficient to reach the best planning performance (consistent with the observation in the LeWM reference setup). Every main configuration on these three benchmarks is trained with three independent random seeds. PushT, OGBench-Cube, and TwoRoom use a $90/10$ train/validation split; LIBERO ranking queries come from episodes excluded from training, as detailed in App. G.

Encoder. For the LeWM instantiation, the encoder f_θ is a Vision Transformer Tiny (ViT-Tiny) operating on 224×224 inputs with patch size 14, projected to an embedding dimension of 192. The encoder is trained jointly with the predictor and uses ImageNet input normalisation. For DINO-WM, we freeze the official pretrained DINOv2 ViT-S/14 (Small) encoder and retain the spatial patch-feature representation of Zhou et al. (2024). VLWM uses the same action-token interface and predictor adaptation as LeWM+VLWM; all remaining training and planning settings are shared.

Predictor. The predictor g_ϕ is a 6-layer causal transformer with 16 heads, head dimension 64, MLP hidden dimension 2048, and dropout 0.1. Although the formulation in Sec. 3.2 permits a state-history window $L \geq 1$, all experiments in this work use only the current state, i.e., $L{=}1$. For a prediction horizon k, the input sequence is $[s_t, e_t, \ldots, e_{t+k-1}]$. Action embeddings $e_\tau = h_\psi(a_\tau)$ are produced by a shared linear head and inserted as first-class tokens, and the output at the final action-token position is projected to the target latent space to obtain $\hat{s}_{t+k}$. No auxiliary [PRED] or mask token is used. Additive type embeddings distinguish state and action tokens, and a single learnable absolute position embedding is shared across horizons.

Variable-length rollout. Given a partition $(k_1, \ldots, k_M)$ of the planning horizon, we initialize $\hat{s}^{(0)} = s_t$ and recursively compute

$$\hat{s}^{(i)} = g_\phi\Big(\hat{s}^{(i-1)}, a_{t+\sum_{j<i} k_j : t+\sum_{j\leq i} k_j - 1}\Big), \qquad i = 1, \ldots, M. \tag{9}$$

Each predictor call therefore receives the latent at the current chunk boundary and the action segment assigned to the next chunk. Our implementation does not use additional state history or predict intermediate latents within a chunk.

VLWM-specific configuration. VLWM departs from LeWM only in: (i) the variable-length training objective of Eq. 3, with maximum horizon $K_{\max}{=}5$ (after frame-skip of 5, this corresponds to a maximum of 25 environment steps per chunk); (ii) the action-as-token sequence layout described in Sec. 3.2, which distinguishes state and action tokens through additive type embeddings while sharing a single learnable absolute position embedding across all k; (iii) the cumulative-uniform horizon curriculum of Eq. 6, with maximum horizons $2, 3, 4, 5$ for $2, 2, 3, 3$ epochs, respectively, over the first 10 epochs of training ($T_2 = 0.2T$, $T_3 = 0.4T$, and $T_4 = 0.7T$); and (iv) the chunked, variable-length rollout used in CEM cost evaluation (Sec. 3.4). The remaining architecture and training settings follow LeWM. Planning solver settings are specified below.

Planning solver. For planning, we use the Cross-Entropy Method (CEM) (Rubinstein & Kroese, 2004). At each replanning step, CEM samples 300 candidate action sequences from a Gaussian distribution (initial standard deviation 1), and the top 30 trajectories are retained as elites. P1–P4 use 30 CEM iterations, matching the official LeWM solver, while the joint action–boundary solver in P5 uses 20 iterations and a Bernoulli probability-sharpening exponent of 2.0. The planning horizon is 5 planning steps (corresponding to 25 environment timesteps under the frame-skip of 5), and we employ MPC with receding horizon 5, meaning the entire optimised action sequence is executed before replanning. This setup follows (Zhou et al., 2024; Maes et al., 2026). Main results use P3 as the predefined fixed-schedule planner and P5 as the adaptive planner; the complete P1–P5 comparison is reported in the analysis. All planner hyperparameters are selected on the validation split.

D VARIABLE-LENGTH PLANNER DETAILS

Prediction and execution horizons. An action block contains five consecutive raw actions, so one model step spans five environment steps. The planning horizon is $H = 5$ model steps and the predictor uses the current latent state ($L = 1$). Each chunk advances the prediction by $k \leq K_{\max} = 5$ model steps. After planning, the controller executes a prefix of $q = 5$ model steps, or 25 raw actions, before replanning. The predictive partition and the executed prefix are separate choices. The following rules specify the planners introduced in Sec. 3.4.

Endpoint cost and action optimization. For an action sequence $\mathbf{a}$ and a predictive partition π, the planning cost is

$$J(\mathbf{a}, \pi) = \|\hat{s}_{t+H}(\mathbf{a}, \pi) - s_g\|_2^2, \tag{10}$$

with squared differences summed over the endpoint features. CEM maintains a diagonal Gaussian over actions for each environment. Each solve initializes its mean to zero and standard deviation to one. At each iteration, $N = 300$ candidates are generated, with the first candidate set to the current mean. The $E = 30$ lowest-cost candidates update the mean and sample standard deviation. P1–P4 use $I = 30$ iterations. The final action mean is returned to the controller.

P1: Fixed-length chunking. Let u_i denote the number of model steps already covered and define the largest feasible chunk as $M_i = \min(K_{\max}, H - u_i)$. P1 uses $k_i = \min(k^\star, H - u_i)$, with $k^\star = 5$ in the evaluation configuration. At $H = 5$, a single predictor call evaluates the complete action sequence.

P2: Uniform chunk sampling. P2 draws $k_i \sim \mathrm{Uniform}\{1, \ldots, M_i\}$ at every chunk start. The feasible set is restricted before sampling. In a batched cost evaluation, all candidates share the sampled partition; a new partition is drawn on the next call to the rollout scorer.

D.1 P3: LONG-TO-SHORT SCHEDULE

For $H > 1$ and $M_i > 1$, define normalized rollout position $\rho_i = u_i/(H - 1)$ and normalized candidate length $v_{i,k} = (k - 1)/(M_i - 1)$. The long-to-short sampler uses

$$p_i(k) = \frac{\exp[\beta(1 - 2\rho_i)v_{i,k}]}{\sum_{j=1}^{M_i} \exp[\beta(1 - 2\rho_i)v_{i,j}]}, \qquad k_i \sim p_i, \quad \beta = 4. \tag{11}$$

When $M_i = 1$, we set $k_i = 1$. The sign of $1 - 2\rho_i$ favors longer feasible lengths near the start and shorter lengths later. The feasible range itself shrinks with the remaining horizon. As in P2, one partition is shared within each batched scorer call and resampled between calls. For $H = 5$, an illustrative partition is $(3, 1, 1)$, predicting endpoints at model steps 3, 4, and 5. These sampling probabilities encode the coarse-to-fine preference.

D.2 P4: COST-ADAPTIVE CHUNKING

After replanning round r, let c_r be the mean of the final-iteration elite costs, averaged over the environments in the solve. This scalar is shared across environments and held fixed throughout round $r + 1$. At each chunk start, P4 selects

$$k_i = \begin{cases} 1, & c_r \leq \tau_{\text{low}}, \\ M_i, & c_r \geq \tau_{\text{high}}, \\ \mathrm{round}\left(1 + \dfrac{c_r - \tau_{\text{low}}}{\tau_{\text{high}} - \tau_{\text{low}}}(M_i - 1)\right), & \text{otherwise}, \end{cases} \tag{12}$$

using $\tau_{\text{low}} = 5$ and $\tau_{\text{high}} = 100$. Rounding follows Python's nearest-integer rule, with ties to even, and the result is clipped to $[1, M_i]$. Before the first feedback signal is available, P4 samples uniformly from 1 to M_i. The remaining-length bound M_i is recomputed at every chunk start. For example, $H = 5$ and $c_r = 20$ yield $(2, 1, 1, 1)$; $c_r = 50$ yields $(3, 1, 1)$; and $c_r \geq 100$ yields (5).

D.3 P5: JOINT ACTION–CHUNK OPTIMIZATION

P5 represents a candidate as an action sequence $\mathbf{a} \in \mathbb{R}^{H \times d_a}$ and binary boundaries $\mathbf{b} \in \{0,1\}^{H-1}$, where $b_j = 1$ closes a chunk after action block j. A terminal boundary at H completes the partition. With the evaluated setting $H = K_{\max} = 5$, every binary pattern induces an admissible partition. For example, $\mathbf{b} = (1,0,1,0)$ gives chunk lengths $(1,2,2)$.

For each environment, P5 maintains Gaussian action parameters $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ and Bernoulli boundary probabilities $\mathbf{p}$. The initial values are $(\mathbf{0}, \mathbf{1})$ and $p_j = 0.5$. Each iteration samples actions and binary boundaries, evaluates the paired candidates, and selects a common elite set $\mathcal{E}$. The action parameters become the elite mean and sample standard deviation, with each standard deviation floored at 10^{-3}. For each boundary coordinate, the update is

$$\widehat{p}_j = \frac{1}{E} \sum_{n \in \mathcal{E}} b_j^{(n)}, \qquad p_j \leftarrow \mathrm{clip}_{[0.02,0.98]} \left(\frac{\widehat{p}_j^{\gamma}}{\widehat{p}_j^{\gamma} + (1 - \widehat{p}_j)^{\gamma}} \right), \quad \gamma = 2. \tag{13}$$

The same elite candidates fit both distributions, while the exponent sharpens the boundary probabilities. P5 uses $N = 300$, $E = 30$, and $I = 20$.

Iteration budget. P5 uses 20 iterations to offset the overhead of evaluating candidate-specific partitions and updating their boundary distribution, retaining the same 300 candidates and 30 elites as P1–P4. Table 2 reports success rates and amortized time per iteration.

Boundary-conditioned rollout. Candidates with the same boundary pattern are grouped for batched prediction. For each group, the predictor processes the corresponding action segments consecutively and returns the final latent. The controller executes the final Gaussian action mean; the boundary pattern of the last iteration's lowest-cost candidate is retained as a diagnostic output.

Planning procedure. At each replanning round, the following steps are performed independently for each environment, with candidates grouped for evaluation:

1. Initialize $\boldsymbol{\mu} = \mathbf{0}$, $\boldsymbol{\sigma} = \mathbf{1}$, and $\mathbf{p} = 0.5$.
2. For each of $I = 20$ CEM iterations, sample N action sequences and N boundary vectors, replacing the first action sequence with $\boldsymbol{\mu}$.
3. Form the partitions and score all candidate pairs using $J(\mathbf{a}, \pi(\mathbf{b}))$.
4. Select the E lowest-cost pairs; update $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ and $\mathbf{p}$ from their action and boundary statistics, then repeat the sampling and scoring steps until all iterations are complete.
5. Return $\boldsymbol{\mu}$, execute its first q model steps, acquire the next observation, and replan.

Compute accounting. The candidate cost counts per environment and replan are $NI = 9{,}000$ for P1–P4 and 6,000 for P5. Both updates in P5 reuse the same scores. A candidate's logical predictor depth equals its number of chunks. At $H = 5$, the sampling rules for P2 and P3 ($\beta = 4$) give expected depths of 2.28 and 1.48, respectively. For P5 this is $1 + \sum_j b_j$, whose initial expectation is 3 at $H = 5$ and $p_j = 0.5$. The actual number of batched predictor calls in one boundary-conditioned scorer invocation is $\sum_{\mathbf{b} \in \mathcal{U}} (1 + \sum_j b_j)$, where $\mathcal{U}$ is the set of sampled patterns. With $H = 5$ there are at most 16 patterns; evaluating all patterns requires 48 batched calls. This distinguishes per-candidate prediction depth from the number of batch operations. For the eight-replan timing runs, Table 2 averages $t_{\mathrm{plan}}/(8I)$ over the three benchmarks, including solver overhead for 50 batched environments; $I = 30$ for LeWM and P1–P4 and $I = 20$ for P5.

E ADDITIONAL CURRICULUM DIAGNOSTICS

Diagnostic setup. We complement Table 1 with a separate pair of PushT runs using half the training data for 10 epochs (63,100 steps each). Curriculum follows the $2 \to 3 \to 4 \to 5$ schedule in App. C; full-range training samples $k \in \{1, \ldots, 5\}$ throughout. Learning-rate warmup and decay restart at each curriculum stage, while full-range training uses a single cycle; both peak at 3×10^{-5} (Fig. 5).

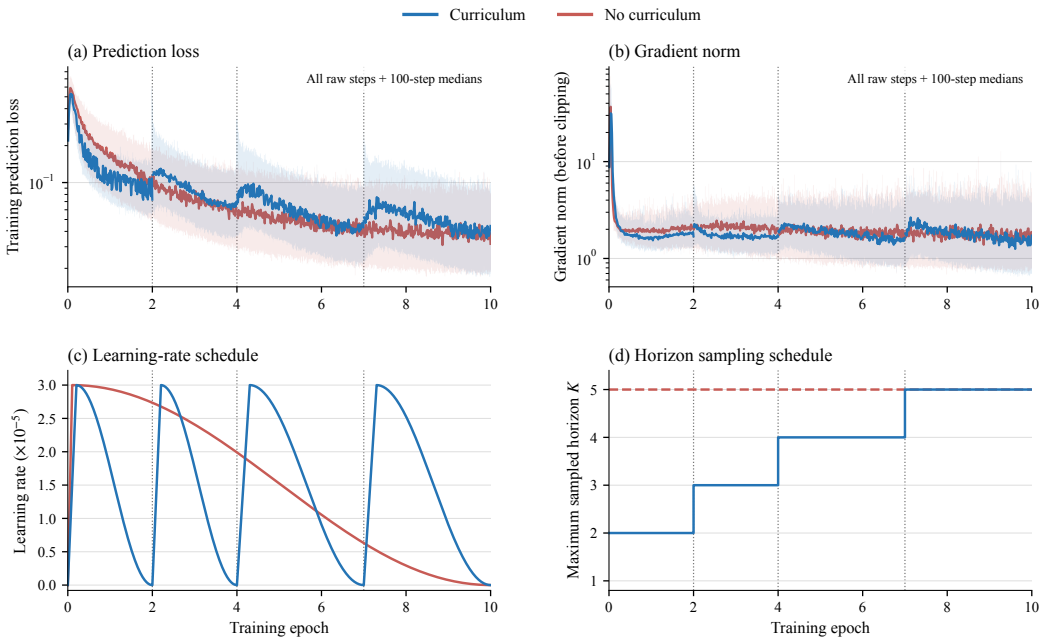

Figure 5: PushT training dynamics, one run per schedule. Faint traces show all raw steps; solid traces show 100-step block medians. Loss and gradient norm use log axes. Dotted lines mark stage transitions; the bottom row shows learning-rate and horizon schedules.

Local fluctuations. We measure multiplicative variation as $d_t = |\log x_t - \log x_{t-1}|$. In the first two epochs, the 99th percentile of prediction-loss variation is 28.0% lower in the curriculum run (0.798 versus 1.109), under its narrower horizon mixture. We then examine the final three epochs, when both runs sample the full range, retaining only originally adjacent steps with $k_t = k_{t-1}$: 3,766 pairs for curriculum and 3,872 for full-range training.

Across these pairs, curriculum reduces prediction-loss variation from 0.474 to 0.436 (7.9%) and gradient-norm variation from 0.878 to 0.720 (17.9%) at the 99th percentile, with reductions also at the median and 95th percentile. At $k = 5$, the 99th-percentile reductions are 19.9% and 18.5%, respectively (Fig. 6). These smaller local fluctuations under the evaluated curriculum schedule complement its downstream planning gains.

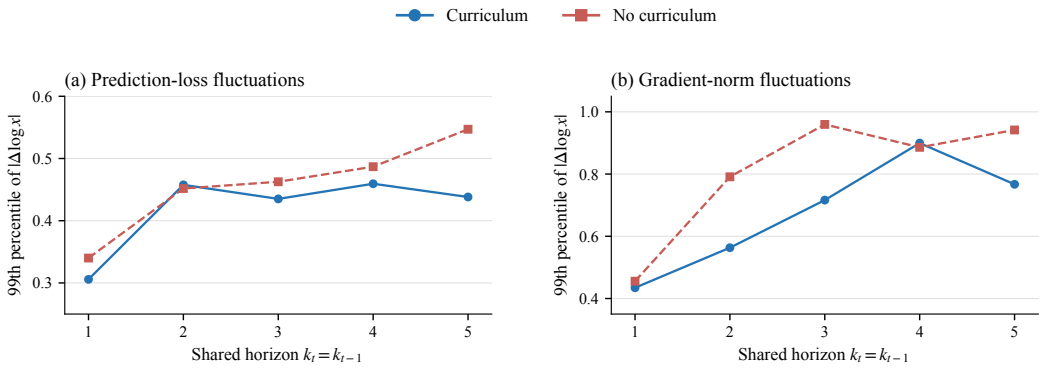

Figure 6: Local fluctuations during the final three epochs. Each point is the 99th percentile of $|\Delta \log x|$ for originally adjacent steps sharing the indicated horizon (706–834 pairs per horizon and run). Lower is better.

F Additional Trajectory-Ranking Results

For each query and prediction horizon h, let $\hat{s}^{(i)}_{t+h}$ be the predicted endpoint of candidate i, and let $s^{(i),\mathrm{GT}}_{t+h}$ be the embedding of its own simulator-replayed terminal observation. We compute $e_i = \mathrm{MSE}(\hat{s}^{(i)}_{t+h}, s^{(i),\mathrm{GT}}_{t+h})$ and rank the reference candidate by increasing e_i. Table 3 reports mean reference rank and Top-k inclusion, the percentage of queries whose reference rank is at most k. Each result uses 50 queries with 300 candidates and one training checkpoint per method. Horizons $h = 1, \ldots, 5$ correspond to $5, 10, 15, 20, 25$ environment steps. The uniform-random reference values are mean rank 150.5 and Top-1/10/30 inclusion of approximately $0.33\%/3.33\%/10\%$.

Table 3: Reference-trajectory ranking across three environments. Each entry uses one training checkpoint per method and 50 queries with 300 candidates each. Mean rank is lower-is-better; Top-k inclusion is higher-is-better. Bold marks the better value within each environment–horizon pair, including ties.

Environment	Model	h	Mean rank ↓	Top-1 (%)	Top-10 (%)	Top-30 (%)
PushT	LeWM	1	149.0	0	2	10
	+VLWM (P3)	1	**63.1**	**4**	**22**	**40**
	LeWM	2	165.2	0	2	12
	+VLWM (P3)	2	**57.1**	**4**	**28**	**56**
	LeWM	3	174.1	0	2	10
	+VLWM (P3)	3	**59.5**	**2**	**16**	**46**
	LeWM	4	180.2	0	0	6
	+VLWM (P3)	4	**77.8**	**10**	**26**	**42**
	LeWM	5	184.7	0	0	4
	+VLWM (P3)	5	**72.8**	**6**	**22**	**34**
OGBench-Cube	LeWM	1	74.0	4	16	36
	+VLWM (P3)	1	**35.2**	**6**	**28**	**60**
	LeWM	2	70.3	4	24	36
	+VLWM (P3)	2	**34.5**	**10**	**38**	**52**
	LeWM	3	77.5	2	16	26
	+VLWM (P3)	3	**30.2**	**8**	**30**	**66**
	LeWM	4	81.3	2	16	28
	+VLWM (P3)	4	**23.0**	**12**	**38**	**74**
	LeWM	5	80.6	2	16	32
	+VLWM (P3)	5	**22.5**	**12**	**38**	**76**
TwoRoom	LeWM	1	152.3	**0**	**0**	**12**
	+VLWM (P3)	1	**149.3**	**0**	**0**	0
	LeWM	2	171.2	**0**	**2**	2
	+VLWM (P3)	2	**148.4**	**0**	**2**	**12**
	LeWM	3	172.0	**0**	0	6
	+VLWM (P3)	3	**145.6**	**0**	**2**	**16**
	LeWM	4	179.7	**0**	0	6
	+VLWM (P3)	4	**156.5**	**0**	**6**	**10**
	LeWM	5	186.5	**0**	0	4
	+VLWM (P3)	5	**139.2**	**0**	**8**	**14**

G LIBERO ACTION RANKING AND PREDICTION DIAGNOSTICS

Queries and evaluation units. We evaluate three independently trained checkpoints for each of LeWM and LeWM+VLWM. Each checkpoint is evaluated on 30 queries drawn from episodes excluded from training. Each query comprises one reference action sequence and 299 distractors, sharing an initial observation and the reference endpoint as a visual goal. We evaluate $K = 1, \ldots, 5$ model steps, corresponding to $5, 10, 15, 20, 25$ raw environment steps. Both models use the same 35-dimensional action-block representation, comprising five consecutive 7-dimensional raw actions, and their respective encoders to embed the initial and goal observations.

Prediction and scoring. LeWM recursively applies its one-step predictor K times, feeding each predicted latent into the next step. LeWM+VLWM predicts the endpoint in one direct call conditioned on all K action blocks. For candidate i, we compute the shared-goal cost

$$c_i = \mathrm{MSE}\Big(\hat{s}^{(i)}_{t+K}, f_\theta(o^{\mathrm{ref}}_{t+K})\Big).$$

With candidate 0 denoting the reference, its average-tie rank is

$$r = 1 + \sum_{i=1}^{299} \mathbb{I}[c_i < c_0] + \frac{1}{2} \sum_{i=1}^{299} \mathbb{I}[c_i = c_0].$$

For each checkpoint and horizon, we compute the mean and median reference rank and Top-k inclusion, $100 \times \Pr(r \leq k)$ for $k \in \{10, 30\}$, over the 30 queries. Table 4 reports the mean and standard deviation of these checkpoint-level statistics across the three independently trained models. Under uniform-random ranking, mean rank is 150.5 and Top-10/30 inclusion is approximately $3.33\%/10\%$.

Table 4: Complete LIBERO goal-conditioned action-ranking results. Each method uses three independently trained checkpoints, each evaluated on 30 held-out queries with 300 candidates per query. Entries report the mean and standard deviation of checkpoint-level statistics. Bold marks the better mean within each horizon, including ties. One model step corresponds to five raw environment steps.

K	Model	Mean rank ↓	Median rank ↓	Top-10 (%) ↑	Top-30 (%) ↑
1	LeWM	**69.46±11.33**	**51.50±9.18**	**25.6±6.9**	**36.7±3.3**
	LeWM+VLWM	79.56±7.55	63.50±24.75	17.8±6.9	28.9±9.6
2	LeWM	**61.79±10.71**	**44.00±10.40**	23.3±3.3	**38.9±7.7**
	LeWM+VLWM	61.80±1.55	45.50±13.11	**26.7±6.7**	**38.9±9.6**
3	LeWM	64.20±13.43	46.00±9.10	16.7±5.8	**40.0±3.3**
	LeWM+VLWM	**62.70±8.00**	**41.17±2.25**	**26.7±5.8**	**40.0±3.3**
4	LeWM	62.21±13.01	52.50±14.29	17.8±1.9	38.9±6.9
	LeWM+VLWM	**46.41±6.57**	**36.00±8.53**	**30.0±3.3**	**45.6±5.1**
5	LeWM	68.70±15.06	49.00±11.82	16.7±0.0	34.4±5.1
	LeWM+VLWM	**51.79±13.76**	**40.83±13.01**	**25.6±6.9**	**45.6±6.9**

Direct versus recursive prediction within VLWM. We further examine recursive error accumulation by comparing the direct K-step and recursive one-step prediction modes of VLWM. Table 5 reports endpoint MSE over $K = 1, \ldots, 5$. The two modes coincide at $K = 1$, while direct prediction yields lower error at every multi-step horizon. At $K = 5$, direct prediction reduces MSE from 0.02434 to 0.00229 (90.6%). Across $K = 2, \ldots, 5$, direct MSE remains between 0.00212 and 0.00234, while recursive error grows with the horizon. This diagnostic supports direct multi-step prediction as a way to reduce recursive error accumulation.

Table 5: Direct and recursive prediction within VLWM on LIBERO. Endpoint MSE is averaged over three evaluation groups of 512 windows per horizon and prediction mode. One model step corresponds to five environment steps. Lower MSE is better; the best values in each row are bold, including ties. Reduction is relative to recursive prediction.

K	Direct MSE ↓	Recursive MSE ↓	Reduction (%)
1	**0.00272**	**0.00272**	0.0
2	**0.00234**	0.00568	58.8
3	**0.00215**	0.01038	79.3
4	**0.00212**	0.01674	87.3
5	**0.00229**	0.02434	90.6