FAST LEWORLDMODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Joint-Embedding Predictive Architectures (JEPAs), including recent LeWorldModel (LeWM), have become a promising foundation for reconstruction-free visual world models. For visual planning, however, LeWM evaluates candidate action sequences by repeatedly applying a local one-step latent transition model. This autoregressive rollout makes planning computationally expensive and causes accumulated latent prediction errors as the horizon grows. We propose **Fast LeWorldModel** (Fast-LeWM), a fast latent world model that replaces repeated local rollout with action-prefix prediction. Given the current latent state and a candidate action sequence, Fast-LeWM encodes its prefixes and predicts the future latents reached after executing those prefixes in parallel. By making action prefixes the basic prediction unit, Fast-LeWM directly models action effects accumulated over different horizons. Training these multi-horizon predictions jointly encourages the model to learn how states continuously evolve under different action prefixes, rather than only fitting one-step state transitions. During planning, the predictor uses the prefix from the encoded action sequence to evaluate the corresponding future latent directly, without explicitly rolling through each intermediate imagined state. Across multiple tasks, Fast-LeWM improves average success over LeWM while substantially reducing planning time. It also achieves lower open-loop latent prediction loss, with errors growing considerably more slowly as the rollout horizon increases.

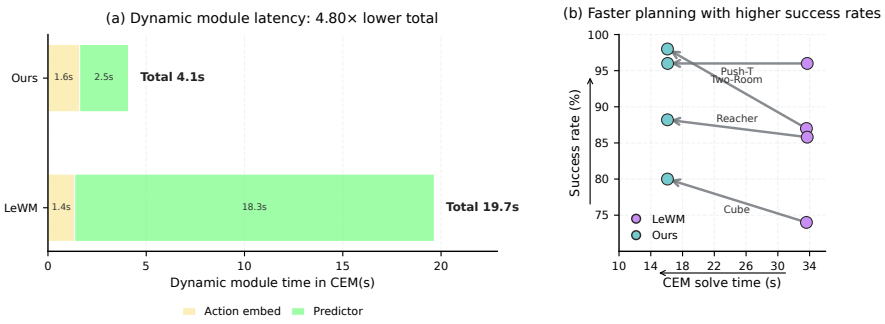


Figure 1: Planning efficiency and task success rates. (a) Runtime breakdown of the two dominant rollout modules. By replacing autoregressive latent rollout with parallel prefix prediction, our method reduces the dynamics-module time in CEM, including action encoding and latent prediction, from 19.7 to 4.1s. (b) Full CEM planning results on four tasks. Each arrow connects LeWM (Maes et al., 2026) and our method under the same planning protocol, showing that our method consistently reduces CEM solve time from about 33.7s to 16.1s while improving or maintaining task success rate.

1 INTRODUCTION

World models enable agents to plan by predicting the consequences of their actions before acting (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2020; Schrittwieser et al., 2020; Wu et al., 2022). For visual planning, reconstruction-free JEPA-style world models learn to predict future embeddings rather than pixels, and recent systems such as LeWorldModel (LeWM) show that latent prediction can support reward-free goal-conditioned planning (LeCun et al., 2022; Assran et al., 2023; Bardes et al., 2024; Assran et al., 2025; Sobal et al., 2025; Zhou et al., 2025; Maes et al., 2026).

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.

Despite this progress, LeWM-style planning remains costly and susceptible to accumulated prediction error. Its dynamics model is inherently local: it predicts only the next latent state $\hat{\mathbf{z}}_{t+1}$ from the current latent $\mathbf{z}_t$ and action $\mathbf{a}_t$. To evaluate a candidate action sequence, the planner must therefore apply this one-step model autoregressively, repeatedly feeding predicted latents back into the dynamics model G_{LeWM}, and the terminal prediction $\hat{\mathbf{z}}_{t+H}$ is obtained by composing local transitions:

$$\hat{\mathbf{z}}_{t+H} = G_{\mathrm{LeWM}}(G_{\mathrm{LeWM}}(\cdots G_{\mathrm{LeWM}}(\mathbf{z}_t, \mathbf{a}_t), \ldots), \mathbf{a}_{t+H-1}).$$

where each prediction depends on the intermediate imagined latents produced along the rollout.

This local transition interface creates two practical limitations. First, candidate evaluation becomes slow because autoregressive rollout must generate the entire imagined latent trajectory step by step, repeating action encoding and latent prediction. Second, errors introduced at early rollout steps can propagate into later predictions, making rollout increasingly unreliable as the horizon grows.

We propose **Fast LeWorldModel** (Fast-LeWM), a fast latent world model built around action-prefix prediction. Instead of advancing the latent state one step at a time, Fast-LeWM treats an action prefix as the basic unit for querying future dynamics. Given the current visual latent z_t and an action prefix $\mathbf{a}_{t:t+k-1} = (\mathbf{a}_t, \ldots, \mathbf{a}_{t+k-1})$, Fast-LeWM predicts the future latent reached after executing that prefix:

$$\hat{\mathbf{z}}_{t+k} = G_{\text{Fast-LeWM}}(\mathbf{z}_t, \mathbf{a}_{t:t+k-1}), \qquad k = 1, \ldots, H.$$

Rather than estimating a long trajectory through repeated adjacent transitions, Fast-LeWM directly predicts the latent outcomes of different action prefixes from the observed anchor latent $\mathbf{z}_t$. Different prefixes contain different degrees of accumulated action effects and correspond to different future latents, so prefix prediction gives the dynamics model a direct interface for multi-horizon state evolution. This formulation also makes each prediction horizon directly supervisable from the same observed anchor latent, enabling dense multi-horizon learning without recursively feeding predicted latents back into the model.

To implement this interface efficiently, Fast-LeWM uses an action-prefix encoder and a parallel latent predictor. The action-prefix encoder processes a candidate action sequence with a causal mask and converts it into multiple prefix tokens, where the token at position k summarizes only $\mathbf{a}_t, \ldots, \mathbf{a}_{t+k-1}$ together with the current latent context. A parallel predictor then maps the current latent and all prefix tokens to their corresponding future latents in one forward pass. Dense prefix-level supervision ties each prefix token to the future latent reached after executing the corresponding partial action sequence. Through this training objective, the model learns to predict in parallel how the state evolves under different action prefixes, rather than only modeling local one-step state evolution.

This design naturally benefits planning. At planning time, the predictor takes all action prefixes as input and predicts all corresponding future states in parallel. These state predictions do not depend on one another sequentially, so errors are not recursively propagated through the H intermediate predicted latents. This reduces recursive error propagation within the planning horizon and accelerates state-evolution prediction through parallel prefix prediction. As shown in Fig. 1, Fast-LeWM substantially reduces the dynamics-module cost inside the Cross-Entropy Method (CEM) (Rubinstein & Kroese, 2004) while improving or preserving task success. The proposed method also lowers open-loop prediction error and slows its growth over longer horizons.

Our contributions are:

- We identify the local one-step transition interface in LeWM as a key bottleneck, leading to slow autoregressive rollouts and the accumulation of latent prediction errors over long horizons.
- We propose Fast-LeWM, a fast latent world model that reformulates latent dynamics modeling from single-step transitions to action-prefix prediction, which enables dense prefix-level training and turns accumulated action effects into direct dynamic training targets.
- We evaluate Fast-LeWM on all planning tasks from LeWM under the same protocol. Fast-LeWM improves the average success rate from 85.8% to 90.5%, accelerates the dynamics module by 4.8× (19.7s to 4.1s), reduces the full CEM solve time by 52.2% (33.7s to 16.1s), and substantially lowers both open-loop prediction error and its growth over the planning horizon.

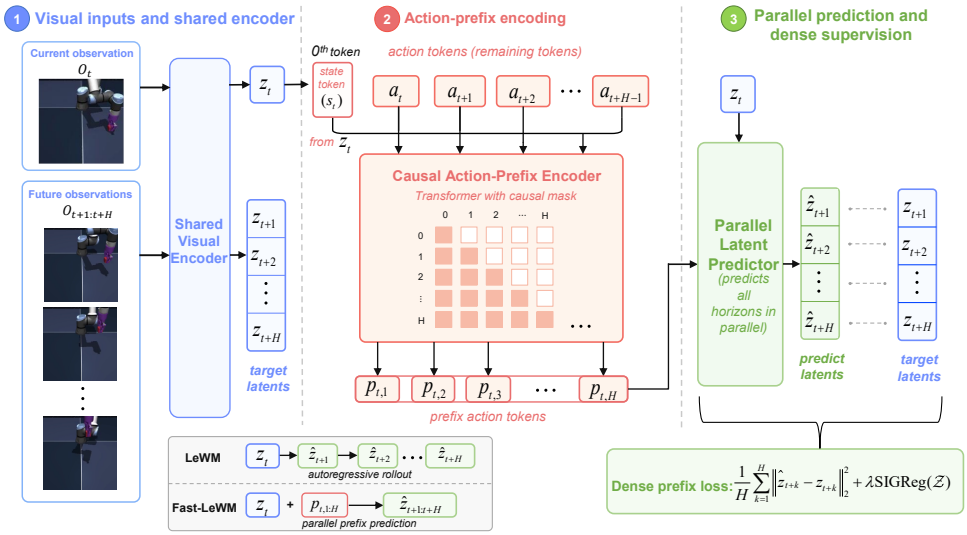


Figure 2: **Fast-LeWM training pipeline.** The current observation o_t is encoded into a latent state z_t. An Action-Prefix Encoder takes a state-action token sequence formed by prepending a state token mapped from z_t as the 0-th token to the action tokens $(a_t, \ldots, a_{t+H-1})$. Under causal masking, the action-position outputs form prefix tokens $(p_{t,1}, \ldots, p_{t,H})$, where $p_{t,k}$ corresponds to the prefix $(a_t, \ldots, a_{t+k-1})$, while the 0-th output token is discarded. Using the prefix tokens together with the anchor latent, the Parallel Predictor directly generates all future latents $(\hat{z}_{t+1}, \ldots, \hat{z}_{t+H})$ in one forward pass, supervised by encoded future observations through a dense multi-horizon loss. Unlike isolated one-step transition learning, dense action-prefix prediction supervises multiple horizons directly, forcing the model to learn the multi-horizon state evolution induced by an action sequence while reducing error propagation into future steps.

2 Related Work

Latent World Models for Planning. World models learn action-conditioned dynamics for planning (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2020; Hou et al., 2026). PlaNet and Dreamer learn latent dynamics from images for planning and behavior learning (Hafner et al., 2019; 2020; 2025), while TD-MPC and TD-MPC2 use decoder-free models for continuous control (Hansen et al., 2022; 2024). In offline reward-free settings, CEM optimizes action sequences by matching predicted future latents to a goal (Rubinstein & Kroese, 2004; Sobal et al., 2025; Zhou et al., 2025; Maes et al., 2026). Repeated candidate evaluation makes rollout speed and long-horizon accuracy critical.

Reconstruction-Free Visual Dynamics. JEPAs predict future or masked embeddings instead of reconstructing pixels (LeCun et al., 2022; Assran et al., 2023; Bardes et al., 2024). For visual planning, PLDM learns from reward-free trajectories with collapse-prevention regularization (Bardes et al., 2022; Sobal et al., 2022; 2025), while DINO-WM predicts frozen DINOv2 patch features (Oquab et al., 2024; Zhou et al., 2025; Yin et al., 2026). LeWM trains an end-to-end JEPA world model using next-embedding prediction and SIGReg-style Gaussian regularization (Balestriero & LeCun, 2025; Maes et al., 2026); recent work further develops this regularization (Zhao et al., 2026; Thil et al., 2026; Wu et al., 2026). Our contribution instead concerns the dynamics-query interface: Fast-LeWM uses action prefixes as multi-horizon queries, conditioning each future latent on the observed anchor latent and the corresponding action prefix rather than solely on the immediately preceding imagined latent, which mitigates the compounding dependence of sequential rollout.

3 Method

3.1 Reward-Free Latent World Models

We consider an offline, reward-free dataset of observation-action trajectories

$$\mathcal{D} = \{\tau^{(n)}\}_{n=1}^{N}, \qquad \tau = \{(o_t, \mathbf{a}_t)\}_{t=1}^{T},$$

where o_t is a pixel observation and $\mathbf{a}_t \in \mathbb{R}^{d_a}$ is a continuous action. The goal is to learn a latent world model that supports goal-conditioned planning without reward labels. A visual encoder maps observations into latent embeddings $\mathbf{z}_t = f_\theta(o_t) \in \mathbb{R}^d$, and a goal observation o_g is similarly encoded as $\mathbf{z}_g = f_\theta(o_g)$.

At test time, the planner evaluates candidate action sequences by predicting the latent states that they would reach from the current state. Each candidate is then assigned a planning cost according to the discrepancy between its predicted future latent and the goal latent $\mathbf{z}_g$. The dynamics model therefore serves as the interface that maps a candidate action sequence to the future latent representation used for planning.

3.2 Autoregressive Rollout in LeWorldModel

A common latent world model design, adopted by LeWM, learns a local one-step transition predictor $\hat{\mathbf{z}}_{t+1} = F_\phi(\mathbf{z}_t, \mathbf{a}_t)$, which is trained with a next-latent prediction loss:

$$\mathcal{L}_{\text{pred}}^{\text{1step}} = \|\hat{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}\|_2^2 .$$

Following reconstruction-free JEPA-style world models, an anti-collapse regularizer, such as SIGReg (Maes et al., 2026), is applied to a batch of latent embeddings Z:

$$\mathcal{L}_{\text{1step}} = \mathcal{L}_{\text{pred}}^{\text{1step}} + \lambda \,\text{SIGReg}(Z).$$

Given a candidate sequence $\mathbf{a}_{t:t+H-1}$, the one-step model estimates the future trajectory by autoregressive rollout,

$$\hat{\mathbf{z}}_{t+k} = F_\phi(\hat{\mathbf{z}}_{t+k-1}, \mathbf{a}_{t+k-1}), \qquad k = 1, \ldots, H,$$

with initialization $\hat{\mathbf{z}}_t = \mathbf{z}_t$. Thus, obtaining the terminal prediction $\hat{\mathbf{z}}_{t+H}$ requires sequentially generating all intermediate latent states $\hat{\mathbf{z}}_{t+1}, \ldots, \hat{\mathbf{z}}_{t+H-1}$. Next, for goal-conditioned planning, the candidate sequence is assigned a terminal latent-space cost:

$$C_{\text{AR}}(\mathbf{a}_{t:t+H-1}) = \|\hat{\mathbf{z}}_{t+H} - \mathbf{z}_g\|_2^2 . \tag{1}$$

The main drawback of this interface is the long autoregressive chain required during candidate evaluation. Although the planning cost in Eq. 1 depends only on the terminal latent, reaching those states requires generating every preceding imagined latent and repeatedly invoking the action encoder and latent transition modules. This repeated rollout becomes expensive when CEM evaluates a large population of candidate sequences over multiple optimization iterations. Moreover, autoregressive rollout can also amplify prediction errors over long horizons. Each intermediate prediction is reused as the input to the next transition, causing approximation errors introduced at early steps to accumulate through all subsequent predictions.

3.3 Fast LeWorldModel

We propose Fast-LeWM to address these issues. Instead of producing $\hat{\mathbf{z}}_{t+1}, \ldots, \hat{\mathbf{z}}_{t+H}$ through an autoregressive chain, our key idea is to predict each future state directly from the observed anchor latent and the action prefix that leads to that state. Under this view, different future states are not generated by feeding one predicted state into the next; they can be queried independently once their corresponding action prefixes are encoded. This motivates the design of an action-prefix encoder together with a parallel latent predictor. Fig. 2 gives an overview of the resulting training pipeline.

For a candidate action sequence $\mathbf{a}_{t:t+H-1}$, each horizon k corresponds to the prefix $\mathbf{a}_{t:t+k-1}$. The target associated with this prefix is the latent state reached after executing these k actions, namely $\mathbf{z}_{t+k}$. Fast-LeWM maps each prefix to a learned prefix token

$$\mathbf{p}_{t,k} = E_\psi^{(k)}(\mathbf{a}_t, \ldots, \mathbf{a}_{t+k-1}),$$

where $\mathbf{p}_{t,k}$ summarizes the accumulated effect of the first k actions. The latent predictor then uses the anchor latent $\mathbf{z}_t$ and each prefix token $\mathbf{p}_{t,k}$ to predict the corresponding future latent $\hat{\mathbf{z}}_{t+k}$.

This design has two direct benefits. During training, dense supervision over all prefix tokens forces the model to learn how action effects accumulate over different horizons, rather than only fitting

local one-step state evolution. During rollout, all action prefixes can be processed together and their corresponding future states can be generated in parallel. Because these predictions are anchored at the observed latent $\mathbf{z}_t$ and do not depend sequentially on one another, the model reduces recurrent error accumulation while accelerating rollout through parallel state-evolution prediction.

3.4 Action-Prefix Encoder

The role of the action-prefix encoder is to output horizon-specific prefix tokens $\mathbf{p}_{t,1:H}$, where each token $\mathbf{p}_{t,k}$ encodes only the first k actions of the sequence and cannot access any subsequent actions. Given an action sequence, the encoder outputs a dense sequence of prefix tokens

$$\mathbf{p}_{t,1:H} = E_\psi(\mathbf{a}_{t:t+H-1}) = (\mathbf{p}_{t,1}, \mathbf{p}_{t,2}, \dots, \mathbf{p}_{t,H}).$$

We instantiate E_ψ as a causal Transformer over action tokens. With the causal mask, the representation at horizon k can attend only to actions $\mathbf{a}_t, \dots, \mathbf{a}_{t+k-1}$. Thus, $\mathbf{p}_{t,k}$ is aligned with the target $\mathbf{z}_{t+k}$ and does not depend on future actions $\mathbf{a}_{t+k}, \dots, \mathbf{a}_{t+H-1}$.

In practice, an action prefix does not determine its outcome by itself. The same open-loop actions may move the agent toward different regions, produce different contacts, or have different object effects depending on the current configuration of the scene. To provide this context with minimal overhead, we map the current latent through a lightweight MLP into a state token and prepend it to the action-token sequence before the causal Transformer as shown in Fig. 2.

3.5 Parallel Latent Predictor

Given the current latent and prefix tokens, the predictor estimates future latents for all prefix horizons in parallel:

$$\hat{\mathbf{z}}_{t+1:t+H} = G_\phi(\mathbf{z}_t, \mathbf{p}_{t,1:H}).$$

Specifically, the prediction at horizon k is given by

$$\hat{\mathbf{z}}_{t+k} = G_\phi(\mathbf{z}_t, \mathbf{p}_{t,k}), \qquad k = 1, \dots, H. \tag{2}$$

Here, each prefix token $\mathbf{p}_{t,k}$ conditions the predictor on the cumulative effect of executing the first k actions, while the current latent $\mathbf{z}_t$ serves as a shared anchor for all prediction horizons. The model therefore learns to map action prefixes of different lengths directly to their corresponding future latent states. This design is computationally favorable for planning: all queried horizons share one action-prefix encoding pass and one parallel latent-prediction pass, without repeatedly iterating one-step transitions.

3.6 Dense Prefix Prediction Objective

For a training segment

$$(o_t, \mathbf{a}_t, o_{t+1}, \mathbf{a}_{t+1}, \dots, \mathbf{a}_{t+H-1}, o_{t+H}),$$

we encode the current and future observations as

$$\mathbf{z}_{t+i} = f_\theta(o_{t+i}), \qquad i = 0, \dots, H.$$

The action-prefix encoder and predictor produce prefix-level predictions $\hat{\mathbf{z}}_{t+1:t+H}$. Every prediction receives its own latent target $\mathbf{z}_{t+1:t+H}$, leading to our loss function:

$$\mathcal{L}_{\text{prefix}} = \frac{1}{H} \sum_{k=1}^{H} \|\hat{\mathbf{z}}_{t+k} - \mathbf{z}_{t+k}\|_2^2 .$$

This dense objective supervises not only the terminal outcome but also the intermediate states induced by partial action prefixes, forcing the model to learn how the latent state evolves as more actions are appended to the sequence.

We retain the SIGReg regularizer used by reconstruction-free latent world models to prevent latent collapse:

$$\mathcal{L}_{\text{AP}} = \mathcal{L}_{\text{prefix}} + \lambda \, \text{SIGReg}(Z).$$

3.7 Planning with Action Prefixes and Self-Consistency

During planning, we use the CEM-based goal-conditioned planning framework similar to LeWM. Let m index a candidate action sequence sampled by CEM, LeWM reaches $\hat{\mathbf{z}}_{t+H}^{(m)}$ by repeatedly applying a one-step dynamics model along imagined latents. In contrast, Fast-LeWM treats the action prefix as the rollout unit, so the corresponding prefix token $\mathbf{p}_{t,H}^{(m)}$ provides a direct path from the current latent to that future latent.

The candidate is evaluated according to its terminal latent distance from the goal:

$$C_{\text{goal}}^{(m)} = \left\| \hat{\mathbf{z}}_{t+H}^{(m)} - \mathbf{z}_g \right\|_2^2 .$$

CEM uses the candidates with the lowest costs to update its sampling distribution, and the selected action sequence is executed until the next decision point.

This prefix-level formulation also enables an optional self-consistency signal during planning. In addition to predicting the terminal latent directly from the length-H prefix, the model can take one intermediate prefix step and then predict the remaining horizon from the intermediate latent, yielding another terminal estimate $\tilde{\mathbf{z}}_{t+H}^{(m)}$. The direct and decomposed predictions correspond to two different ways of predicting the terminal latent from the same candidate sequence. The discrepancy between these two terminal estimates can be added as a model-consistency penalty:

$$C^{(m)} = C_{\text{goal}}^{(m)} + \beta \left\| \hat{\mathbf{z}}_{t+H}^{(m)} - \tilde{\mathbf{z}}_{t+H}^{(m)} \right\|_2^2 , \tag{3}$$

Here, β is a hyperparameter that weights the optional self-consistency term. This penalty does not directly measure whether a candidate action sequence achieves the task. Instead, it encourages CEM to favor candidates whose terminal predictions are stable under different prefix decompositions.

4 Experiments

We evaluate Fast-LeWM in terms of planning performance and efficiency, open-loop latent prediction, physical state probing, and ablations. Experimental settings and implementation details are provided in Appendices B–E.

4.1 Planning Performance and Efficiency

Table 1 evaluates the planning performance of Fast-LeWM. Under the same planning protocol, Fast-LeWM matches or outperforms LeWM in all four environments, increasing the average success rate from 85.8% to 90.5%. Augmenting candidate scoring with the optional self-consistency term in Eq. 3 further increases the average success rate to 92.0%, suggesting that consistency across different prefix-based terminal estimates provides a useful auxiliary signal for distinguishing reliable candidate sequences.

Table 2 provides a runtime evaluation under the same CEM budget. We report both the dynamics-evaluation time, which includes action encoding and latent prediction, and the full CEM solve time. The latter additionally includes goal and observation encoding, candidate scoring, and other data-processing overhead. Unlike LeWM, which applies its dynamics predictor sequentially over the planning horizon, Fast-LeWM evaluates each candidate sequence using a single action-prefix encoding pass followed by parallel latent prediction. Both variants of Fast-LeWM in Table 2 reduce dynamics-evaluation time and full CEM solve time compared with LeWM. In particular, Fast-LeWM reduces dynamics-evaluation time from 19.7s to 4.1s and CEM solve time from 33.7s to 16.1s, corresponding to speedups of 4.8× and 2.1×, respectively. Overall, the proposed algorithm improves planning success while making the repeated evaluation of candidate action sequences substantially more efficient in CEM-based planning.

4.2 Open-Loop Latent Prediction

We first quantitatively evaluate open-loop prediction accuracy in latent space. Given an initial observation and a sequence of future actions, each model predicts the latent states along the resulting

Table 1: Planning success rate (%) across environments. With the same planning objective as LeWM, Fast-LeWM improves the average success rate from 85.8% to 90.5%. Adding the optional self-consistency term to candidate scoring further increases the average success rate to 92.0%.

Method	Two-Room	Reacher	PushT	Cube	Avg.
PLDM (Sobal et al., 2025)	97	78	78	65	79.5
DINO-WM (Zhou et al., 2025)	**100**	79	74	**86**	84.8
LeWM (Maes et al., 2026)	87	86	<u>96</u>	74	85.8
Fast-LeWM	<u>98</u>	<u>88</u>	<u>96</u>	80	<u>90.5</u>
Fast-LeWM + Self-Consistency	<u>98</u>	**90**	**98**	<u>82</u>	**92.0**

Table 2: Planning efficiency under the same CEM budget. Dynamics time includes action encoding and latent prediction, whereas CEM time covers the complete planning procedure. All runtimes are measured on a single NVIDIA 4090.

Method	Model calls	Dynamics time(s)	CEM time(s)
LeWM	5	19.7	33.7
Fast-LeWM + Self-Consistency	3	11.9	25.7
Fast-LeWM	**1**	**4.1**	**16.1**

open-loop trajectory. We compare these predictions with the latent representations encoded from the corresponding ground-truth future observations.

We evaluate predictions over 50 environment steps. As an action skip of 5 is used, this evaluation corresponds to 10 latent transition steps. With a maximum prediction horizon of $H = 5$, Fast-LeWM predicts each five-step latent segment in parallel and chains two such segments to cover the full evaluation horizon. In contrast, LeWM applies its one-step transition model autoregressively throughout all 10 latent transition steps. We use the same set of test trajectories for both methods.

As shown in Fig. 3, Fast-LeWM achieves substantially lower latent prediction error at every evaluated step across all four tasks. To quantify how rapidly prediction error increases over the open-loop trajectory, we fit a linear trend to each loss curve in Fig. 3 using least squares: $\mathcal{L}_{\text{latent}}(t) \approx \alpha + \gamma t$, where t denotes the horizon in raw environment steps, γ measures the average slope of error growth, and α is the intercept. Across all four tasks, Fast-LeWM yields a smaller fitted slope than LeWM. These results show that action-prefix prediction improves open-loop prediction accuracy and slows the accumulation of latent error over extended horizons.

We further provide a qualitative comparison by visualizing predicted latents with a decoder similar to LeWM. Both models are conditioned on the same initial observation at $t = 0$ and the same future action sequence. As shown in Fig. 4, the decoded predictions of Fast-LeWM remain more consistent with the ground-truth trajectory, whereas LeWM exhibits increasingly visible drift over longer horizons due to accumulated rollout error.

4.3 Physical State Probing

To examine the physical information encoded in the learned latent representations, we conduct probing experiments on PushT following the protocol of LeWM (Maes et al., 2026). We freeze the visual encoder and train lightweight probes to predict three task-relevant physical variables from latent embeddings: agent location, block location, and block angle. We report both linear and MLP probes, where the linear probe measures direct accessibility of the physical information, while the MLP probe evaluates whether the information is retained in a nonlinearly recoverable form.

Table 3 reports the mean squared error and Pearson correlation coefficient r. Under linear probing, Fast-LeWM performs similarly to or slightly better than LeWM for agent and block locations, while LeWM more accurately recovers the block angle, indicating that its latent representation retains substantial linearly accessible physical structure.

The advantage of Fast-LeWM becomes more pronounced with MLP probes. It achieves the lowest MSE and highest correlation for all three physical variables, including a substantial reduction in

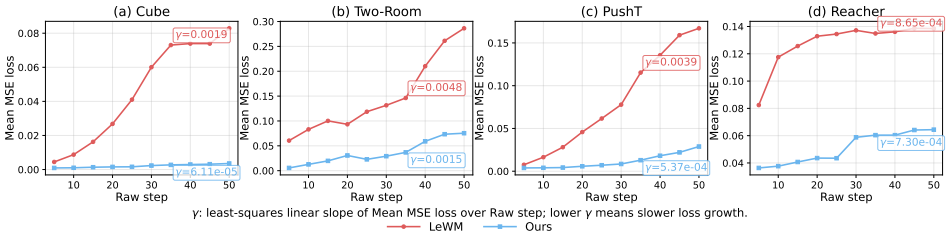


Figure 3: Open-loop latent prediction loss over 50 raw environment steps across four tasks. The loss is computed between the predicted latent and the latent encoded from the corresponding ground-truth frames. The fitted linear slope γ measures the average growth rate of latent prediction error with respect to the prediction horizon. Compared with LeWM, Fast-LeWM achieves lower loss at every evaluated step and smaller fitted slopes across all four tasks. This shows that the action-prefix design substantially mitigates open-loop error accumulation and provides a more error-resistant interface for modeling continuous state transitions over multiple horizons.

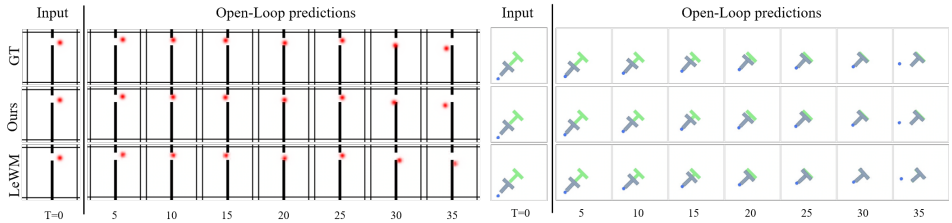


Figure 4: Qualitative comparison of open-loop latent predictions, visualized by decoding the predicted latents. Both methods are initialized from the same observation and conditioned on the same future action sequence. The predictions are similar at short horizons, but the difference becomes more pronounced after 20 environment steps: Fast-LeWM remains closer to the ground-truth trajectory, whereas LeWM exhibits increasingly visible drift.

block-angle error relative to LeWM. These results indicate that Fast-LeWM retains physical state information that may not always be linearly organized but is more accurately recoverable through a lightweight nonlinear probe. We attribute this to the prefix-level training objective: by forcing the model to predict how the state evolves under different action prefixes, rather than only local one-step state evolution, the latent representation must preserve fine-grained physical variables that influence future motion, contact, and object configuration. This encourages the encoder and dynamics module to learn more informative state-aware features rather than representations that only support local one-step changes.

4.4 Ablation Study

We analyze the roles of the prediction interface, dense multi-horizon supervision, and state conditioning in Table 4. We first examine whether direct long-horizon prediction can be achieved by simply enlarging the temporal span of LeWM. The resulting baseline, termed Long-Action LeWM, expands the original frame-skipped action encoding from 5 to 25 primitive actions and predicts the terminal latent after the longer action block in a single transition. Its average success rate drops to 71.0%, indicating that naively extending the temporal span of the original LeWM transition is insufficient for accurate one-pass long-horizon prediction.

We next study the role of dense multi-horizon supervision in Fast-LeWM. Without dense supervision, Fast-LeWM achieves an average success rate of 84.5%, which is comparable to the 85.8% of the original LeWM while requiring substantially fewer dynamics calls during planning. Adding dense prefix-level supervision improves the average success rate to 90.5%. This result shows that dense supervision is important for learning the more challenging direct multi-horizon prediction problem induced by the action-prefix interface.

To further examine whether the benefit comes simply from adding multi-step supervision, we apply an analogous dense objective to LeWM. "LeWM w/ Dense Loss" receives the same variable-horizon inputs as Fast-LeWM, performs H autoregressive rollout steps, and averages the latent prediction

Table 3: Physical latent probing on PushT. We report MSE and Pearson correlation r for both probes.

Property	Model	Linear		MLP	
		MSE ↓	r ↑	MSE ↓	r ↑
Agent Loc.	PLDM	0.090	0.955	0.014	0.993
	LeWM	0.052	0.974	0.004	0.998
	Ours	**0.048**	**0.976**	**0.001**	**1.000**
Block Loc.	PLDM	0.122	0.938	0.011	0.994
	LeWM	**0.029**	0.986	0.001	0.999
	Ours	**0.029**	**0.987**	**0.000**	**1.000**
Block Angle	PLDM	0.446	0.745	0.056	0.972
	LeWM	**0.187**	**0.902**	0.021	0.990
	Ours	0.314	0.828	**0.009**	**0.995**

Table 4: Ablation study. All variants use the same training protocol and CEM budget.

Variant	Two-Room	Reacher	PushT	Cube	Avg.
LeWM	87	86	96	74	85.8
Long-Action LeWM	76	70	80	58	71.0
LeWM w/ Dense Loss	90	78	90	58	79.0
Fast-LeWM w/o Dense Loss	96	80	90	72	84.5
Fast-LeWM	98	88	96	80	**90.5**
Fast-LeWM w/o State Token	94	82	92	80	<u>87.0</u>

losses along the rollout. Its average success rate decreases to 79.0%, below the original LeWM. This contrast suggests that dense multi-horizon supervision is not uniformly beneficial across prediction interfaces. In Fast-LeWM, each horizon is predicted directly from the observed anchor latent and its corresponding action-prefix representation, whereas in autoregressive LeWM later-horizon predictions depend recursively on earlier predicted latents. The effectiveness of dense supervision therefore depends on how multi-horizon dynamics are modeled.

Finally, removing the state token of the action-prefix encoder also degrades performance. While the state token introduces little overhead, it provides essential context for interpreting the candidate action sequence. The same action prefix can produce different outcomes depending on the initial agent position, object configuration, scene geometry, and contact conditions. Conditioning the prefix encoder on the current state therefore helps disambiguate these effects and improves the prediction of the future latent associated with each action prefix.

5 Limitations and Future Work

Although Fast-LeWM improves long-horizon latent prediction and substantially slows error growth, it still relies on CEM to optimize action sequences at planning time. As the planning horizon increases, CEM must search over an increasingly high-dimensional action space, making it difficult to identify effective action sequences. A promising direction is to combine Fast-LeWM with hierarchical planning (Zhang et al., 2026), where high-level planning operates over temporally extended subgoals rather than individual actions. We leave this extension to future work.

6 Conclusion

We introduced Fast-LeWM, a fast latent world model for visual planning. Instead of relying on autoregressive one-step rollout, Fast-LeWM predicts future latents directly from the current state and action-prefix representations. This design reduces repeated dynamics evaluations, mitigates latent error accumulation, and enables dense supervision over continuous state evolution. Experiments across four planning tasks show that Fast-LeWM improves planning success while substantially reducing CEM solve time and open-loop prediction error. These results suggest that action-prefix prediction is an effective and efficient dynamics interface for latent world-model planning.

AI USE STATEMENT

In this work, we used generative AI tools for translation, method implementation, and assistance with model architecture refinement. We have not used generative AI tools for other tasks requiring disclosure, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for code completion and refactoring, language polishing and editing to improve manuscript clarity and readability, and literature search. We have reviewed all AI-assisted work. AI-assisted code was manually inspected and tested, manuscript text and translations were reviewed by the authors, and suggested references were verified against the original sources.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the source code in an anonymous, downloadable repository at `https://anonymous.4open.science/r/anonymous-3080`. The repository includes the model implementation, training and evaluation scripts, and configuration files specifying model, optimization, and planning hyperparameters. The accompanying README provides instructions for environment setup, dataset organization, training, and evaluation, while the dependency versions are specified in `requirements.txt`.

REFERENCES

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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, 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.

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

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Adrien Bardes, Quentin Garrido, Jean Ponce, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv:2404.08471*, 2024.

David Ha and Jurgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

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

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

Bohan Hou, Gen Li, Jindou Jia, Tuo An, Xinying Guo, Sicong Leng, Haoran Geng, Yanjie Ze, Tatsuya Harada, Philip Torr, et al. World model for robot learning: A comprehensive survey. *arXiv preprint arXiv:2605.00080*, 2026.

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

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:2603.19312*, 2026.

Maxime Oquab, Timoth'ee Darcet, Th'eo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herv'e J'egou, 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 (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*, volume 133. Springer, 2004.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Vlad Sobal, Jyothir SV, Siddhartha Jalagam, Nicolas Carion, Kyunghyun Cho, and Yann LeCun. Joint embedding predictive architectures focus on slow features. *arXiv:2211.10831*, 2022.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Lucas Thil, Jesse Read, Rim Kaddah, and Guillaume Doquet. Subspace-decomposed jepas: Disentangling progression and content in latent world models. *arXiv preprint arXiv:2605.31111*, 2026.

Haiyu Wu, Randall Balestriero, and Morgan Levine. Visreg: Variance-invariance-sketching regularization for jepa training. *arXiv preprint arXiv:2606.02572*, 2026.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. Daydreamer: World models for physical robot learning. *Conference on Robot Learning*, 2022.

Shicheng Yin, Kaixuan Yin, Weixing Chen, Yang Liu, Guanbin Li, and Liang Lin. Ddp-wm: Disentangled dynamics prediction for efficient world models. *arXiv preprint arXiv:2602.01780*, 2026.

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

Kai Zhao, Dongliang Nie, Yuchen Lin, Zhehan Luo, Yixiao Gu, Deng-Ping Fan, and Dan Zeng. Sub-jepa: Subspace gaussian regularization for stable end-to-end world models. *arXiv preprint arXiv:2605.09241*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning (ICML)*, 2025.

APPENDIX

A MULTI-SEED PLANNING RESULTS

In the main paper, we follow the evaluation protocol of the baselines and use the same random seed as LeWM for direct comparison. To further evaluate the robustness of planning performance, we additionally evaluate both LeWM and Fast-LeWM over seven random seeds, while keeping the planning protocol identical across methods. As shown in Table 5, Fast-LeWM achieves higher mean success rates on all four tasks, improving the overall average from 81.2% to 85.7%. These multi-seed results are consistent with the main-paper comparison obtained under the original LeWM seed, indicating that our performance gains are not specific to a particular random seed.

Table 5: Planning success rate (%) over seven random seeds. We report mean $\pm$ standard deviation across seven seeds for each task. The last column reports the average success rate across the four tasks.

Method	Two-Room	Reacher	PushT	Cube	Avg.
LeWM	87.0 ± 4.9	83.4 ± 5.7	87.1 ± 7.2	67.4 ± 8.6	81.2
Fast-LeWM	$\mathbf{95.4 \pm 3.2}$	$\mathbf{84.0 \pm 5.9}$	$\mathbf{91.7 \pm 4.1}$	$\mathbf{71.7 \pm 8.4}$	**85.7**

B EXPERIMENTAL SETUP AND IMPLEMENTATION DETAILS

B.1 DATASETS AND SHARED PROTOCOL

Unless otherwise specified, we follow LeWM for dataset preprocessing, model training, and CEM-based goal-conditioned evaluation, including image resolutions, optimizer settings, and all remaining training and evaluation hyperparameters. Experiments are conducted on the same offline datasets from Two-Room (Sobal et al., 2025), PushT (Zhou et al., 2025), Reacher (Tassa et al., 2018), and OGBench-Cube (Park et al., 2025).

B.2 MODEL ARCHITECTURE

We adopt the same ViT visual encoder as LeWM. The action-prefix encoder is implemented as a causal Transformer over a sequence of action tokens. The starting token is obtained by mapping the current latent $\mathbf{z}_t$ through a two-layer MLP with hidden width 768, while the remaining tokens are embeddings of the future actions $\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1}$. The token dimension is 192. The action-prefix Transformer has 3 layers and 6 attention heads, with per-head dimension 32. We use sinusoidal positional encoding with exponentially spaced frequencies.

For each horizon k, the latent dynamics predictor takes the current latent $\mathbf{z}_t$ and the corresponding prefix token $\mathbf{p}_{t,k}$ to predict the future latent $\hat{\mathbf{z}}_{t+k}$ as introduced in Eq. 2. It is implemented as a 6-layer residual MLP in which the prefix representation modulates the latent features through AdaLN-Zero. The predictor uses a latent dimension of 192, a hidden width of 2048, a modulation width of 768, and a dropout rate of 0.1. The complete model contains 17.9M parameters, comparable to the 18.0M parameters of LeWM.

B.3 TRAINING SETUP

We train Fast-LeWM with a batch size of 128 on all environments except Cube, for which we use a batch size of 32 to obtain more stable and faster convergence. All models are trained for 10 epochs, following the LeWM protocol. During training, the prediction horizon is determined adaptively from the loaded trajectories and clamped to the range $[1, 5]$.

Algorithm 1 Dense Action-Prefix Training for Fast-LeWM

Require: Dataset $\mathcal{D}$, horizon H, models $(f_\theta, E_\psi, G_\phi)$, objective $\mathcal{L}_{\text{train}}$

repeat
  Sample $(\mathbf{o}_{t:t+H}, \mathbf{a}_{t:t+H-1}) \sim \mathcal{D}$
  $\mathbf{z}_{t:t+H} \leftarrow f_\theta(\mathbf{o}_{t:t+H})$
  $\mathbf{s}_t \leftarrow \mathrm{MLP}_{\text{state}}(\mathbf{z}_t)$
  $\mathbf{x}_{t,0:H} \leftarrow [\mathbf{s}_t; \mathrm{Emb}(\mathbf{a}_{t:t+H-1})]$
  $[\mathbf{p}_{t,0}, \mathbf{p}_{t,1:H}] \leftarrow E_\psi(\mathbf{x}_{t,0:H}; \mathbf{M}_{\text{causal}})$
  $\hat{\mathbf{z}}_{t+1:t+H} \leftarrow G_\phi(\mathbf{z}_t, \mathbf{p}_{t,1:H})$
  Take a gradient step on $\nabla_{\theta,\psi,\phi}\mathcal{L}_{\text{train}}$
until converged

C Training Procedure

Algorithm 1 summarizes the training procedure of Fast-LeWM. Given a trajectory segment, the shared visual encoder first maps the current and future observations to latent representations. The current latent is transformed into a state token and prepended to the embedded action sequence. A causal Transformer then produces one representation for every action prefix, and the parallel predictor maps these prefix representations to their corresponding future latents.

Under causal masking, the prefix representation $\mathbf{p}_{t,k}$ can depend only on the first k actions $\mathbf{a}_t, \ldots, \mathbf{a}_{t+k-1}$. The shared anchor $\mathbf{z}_t$ and the prefix representations $\mathbf{p}_{t,1:H}$ are then used to predict all H future latents in one parallel forward pass.

D Planning Protocol

At test time, we follow the same goal-conditioned latent planning protocol as LeWM. We use the same frame-skipped control interface with an action skip of 5 and set the planning horizon to $H = 5$. Therefore, each planned action is executed for five primitive environment steps, and the complete planning horizon covers 25 primitive steps. The optimized action sequence is executed according to the same model-predictive control schedule used by LeWM.

For the self-consistency variant, we additionally compare two predictions of the terminal 25-step latent. The first prediction is obtained directly from the complete action prefix. The second is obtained by first predicting an intermediate 10-step latent and then predicting the remaining horizon from this intermediate state. We set the consistency-loss weight to $\beta = 1$.

E Runtime Measurement

Runtime is measured on a single NVIDIA RTX 4090 using the same evaluation workload, candidate action sequences, and CEM settings for both methods. Each CEM iteration evaluates 300 candidates and retains 30 elites, with the same number of optimization iterations for both methods. We separately report dynamics-evaluation time and full CEM solve time.

Dynamics-evaluation time covers action encoding and latent prediction, while full CEM time includes all encoding and CEM optimization costs. Both exclude environment stepping and rendering.

Both metrics are accumulated over all planning calls across all evaluation samples and trajectories. The reported runtimes therefore represent the total planning cost of the complete evaluation workload, rather than an average per planning call. This difference in aggregation explains their larger magnitude compared with the averaged planning times reported in LeWM.