PRETRAINED FEATURES AS TEACHERS, NOT STATES: COMPACT LATENT WORLD MODELS FOR VISUAL PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A key design choice in visual world models is what information the latent state is encouraged to preserve. Models that learn a latent space with only a latent-prediction objective adapt to environment dynamics, but the predictive objective alone admits collapsed solutions and provides no explicit constraint on the visual information retained by the latent. Models built directly on pretrained visual features inherit a strong visual prior, but tie prediction and planning to a frozen state space. We propose Anchor-WM, which instead uses pretrained visual features as a teacher for learning a compact latent state rather than as the world-model state itself. Anchor-WM learns this compact latent with action-conditioned dynamics, while distilling the dense DINO representations through a training-only distillation head. This encourages the compact latent to retain visual information captured by the pretrained representation while discouraging representational collapse. At inference, the DINO encoder and distillation head are discarded, allowing prediction and planning to operate entirely in the learned latent space. Anchor-WM achieves the best average success rate across the four evaluated environments while enabling substantially faster planning than world models operating directly on DINO patch features. These results show that pretrained visual features can effectively guide world-model learning without defining the state space used for prediction and planning.

1 INTRODUCTION

World models aim to capture how an environment evolves over time conditioned on an agent's actions so as to support prediction, planning, and control (Ha & Schmidhuber, 2018). In visual domains, they often rely on learned latent states to represent observations compactly, and a central challenge is determining what information these states should retain. A common approach is to learn such representations with observation reconstruction as part of the training objective (Hafner et al., 2020; 2023), but a reconstruction objective requires modeling fine-grained visual details, such as textures and lighting variations, even when these details are irrelevant to downstream control.

One alternative is to model dynamics in a pretrained visual representation rather than in pixel space. DINO-WM learns a world model that operates directly on pretrained DINO patch features, providing a spatially structured visual representation for offline model-based control and planning (Zhou et al., 2025). While this avoids predicting raw observations and provides a strong visual prior enabling zero-shot generalization, directly adopting the frozen pretrained representation as the world-model state imposes two limitations. First, the state cannot adapt to the dynamics and control requirements of the target environment. Second, DINO represents an image as a spatial grid of patch features, yielding a dense state that must be propagated through the dynamics model for every candidate action sequence during planning, which can make planning computationally expensive.

A different approach is to learn the world-model state end-to-end from pixels, while predicting future states in latent space rather than reconstructing pixels, following the Joint Embedding Predictive Architecture (JEPA) (LeCun, 2022). Recent JEPA-based world models, such as LeWM, take this route (Maes et al., 2026; Terver et al., 2026), allowing the learned state to freely adapt to the target environment rather than inheriting a fixed pretrained state space. In reward-free settings, however,

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.

the predictive objective alone does not determine which visual information the representation should preserve and, in the extreme, admits collapsed solutions. These methods therefore rely on architectural or regularization strategies to prevent collapse. Avoiding collapse, however, does not by itself determine which visual information the learned state should retain.

These approaches differ in how they define the world-model state. Using a pretrained visual representation directly provides a semantically rich state, but fixes the representation on which dynamics and planning must operate. Learning the state end-to-end allows it to adapt to the task and dynamics, but provides no explicit constraint on what visual information it should preserve. This raises a natural question: *Can pretrained visual representations instead guide the learning of a compact predictive state, without being used directly as the state for prediction and planning?*

We propose Anchor-WM, which uses pretrained visual features only as training-time supervision to learn a compact latent space for prediction and planning. The compact latent is trained jointly to predict future latent states conditioned on actions and to reconstruct dense DINO features through a lightweight distillation head. This encourages the learned latent to retain visual information captured by DINO while remaining free to adapt to the environment dynamics. The distillation objective also helps prevent representational collapse by providing an additional constraint on the learned latent space. At inference, the DINO encoder and distillation head are discarded, and prediction and planning operate entirely in the learned compact latent space instead of dense DINO feature space.

We evaluate Anchor-WM across diverse visual control environments against multiple world-model baselines including DINO-WM and LeWM. Anchor-WM achieves the highest average success rate across the considered environments, while remaining competitive with the best-performing method on each individual task, suggesting that it benefits from the visual prior provided by pretrained features while retaining the flexibility to adapt its latent state to the target dynamics. Meanwhile, Anchor-WM plans approximately $100\times$ faster than DINO-WM because planning is performed entirely in a compact learned latent space rather than directly in the high-dimensional DINO feature space, with planning speed comparable to LeWM. These results show that pretrained visual features can guide the learning of a compact latent state that supports strong planning performance without incurring the inference-time cost of operating directly in the pretrained feature space.

Overall, we contribute the following: (1) We introduce Anchor-WM, a world model that uses pretrained visual features as training-time guidance while performing prediction and planning entirely in a compact latent space; (2) We develop a feature-distillation objective that distills pretrained visual information into the latent space while helping prevent representational collapse; and (3) We demonstrate across diverse visual control environments that this design yields strong planning performance with substantially lower planning cost than operating directly in pretrained feature space.

2 Related Work

Latent World Models. Many world-modeling approaches learn dynamics in latent spaces. PlaNet (Hafner et al., 2019), Dreamer (Hafner et al., 2023), and IRIS (Micheli et al., 2023) use observation reconstruction as part of representation learning, whereas TD-MPC (Hansen et al., 2024) learns a decoder-free latent model shaped by reward and value supervision. JEPA-based world models instead learn representations by predicting future latent without reconstructing pixels or predicting rewards (LeCun, 2022; Assran et al., 2023; Bardes et al., 2024). A central challenge when learning such representations end-to-end is representational collapse, which prior work addresses through target encoders or explicit representation regularization (Bardes et al., 2021; 2024; Sobal et al., 2022; 2025; Assran et al., 2023; Balestriero & LeCun, 2025; Maes et al., 2026; Terver et al., 2026). Recent work further constrains the information or structure captured by the learned state through action-based objectives (Ivashkov et al., 2026; Zhang et al., 2026) or temporal relationships derived from offline trajectories (Bai & Xiong, 2026). Our proposed approach constrains the latent representation learning through distilling DINO features, encouraging the learned state to retain visual information captured by DINO while directly discouraging representational collapse.

Pretrained Visual Representations for Control and World Modeling. Large-scale visual pretraining provides representations that transfer across perception and embodied-control tasks. Self-supervised models such as DINO and DINOv2 learn general-purpose visual features (Caron et al., 2021; Oquab et al., 2024), while R3M, MVP, VIP, and related work study visual pretraining specif-

ically for robotic control (Nair et al., 2023; Xiao et al., 2022; Ma et al., 2023; Radosavovic et al., 2023). Pretrained representations have also been incorporated into world models. DINO-WM operates directly on frozen DINOv2 patch features, predicting future spatial representations from offline trajectories and planning toward visual goals in the DINO feature space (Zhou et al., 2025). V-JEPA 2 leverages large-scale self-supervised video pretraining and post-trains an action-conditioned latent world model for robot planning (Assran et al., 2025). Temporal Straightening learns planning-oriented latent representations by projecting pretrained DINO features through a lightweight projector and applying a curvature regularizer to shape the latent geometry (Wang et al., 2026). Across these pretrained-feature approaches, the pretrained representation remains part of the inference procedure—as the state space, as the pretrained backbone, or as the input to a learned projector. Our approach instead uses the pretrained representation only as a training target, so neither the pretrained encoder nor its representation is required for prediction or planning.

Pretrained Representations as Teachers. Our approach is also related to knowledge distillation, which transfers information from a teacher to a separately learned student, including through intermediate representations and their relational structure (Hinton et al., 2015; Romero et al., 2015; Zagoruyko & Komodakis, 2017; Ahn et al., 2019; Tung & Mori, 2019). More recently, pretrained visual representations have provided auxiliary supervision for models optimized for other objectives. REPA aligns projected diffusion-transformer representations with frozen pretrained visual features (Yu et al., 2025), a principle extended to multimodal world models by M^2-REPA (Xiao et al., 2026) and to action grounding by AGRA (Qiu et al., 2026). Related work has also distilled pretrained visual features into compact descriptors for matching and retrieval, where the compressed descriptor is itself the deployed representation (Lee et al., 2021). Our approach uses pretrained feature distillation to learn a compact world-model state. Rather than matching the teacher's representation space, a training-only distillation head encourages the dense DINO representation to remain recoverable from a single compact latent. The teacher thus guides representation learning without defining the state space used for prediction and planning.

3 Method

We consider an offline visual control setting with a dataset of M trajectories $\mathcal{D} = \{\tau^{(i)}\}_{i=1}^{M}$, where $\tau^{(i)} = (o_0, a_0, o_1, a_1, \ldots, o_T)$, o_t denotes an image observation, and a_t the action taken at time t. Our goal is to learn a compact latent dynamics model from these offline trajectories that supports both prediction and visual planning. We next introduce Anchor-WM.

3.1 Latent World Model

To learn a latent world model, Anchor-WM uses an encoder $\mathcal{E}_\theta(\cdot)$ to map an image observation $o_t \in \mathbb{R}^{H\times W\times C}$ to a compact latent state $z_t \in \mathbb{R}^d$:

$$z_t = \mathcal{E}_\theta(o_t). \tag{1}$$

Dynamics are modeled directly in this compact latent space. Given a context of K latent states $z_{t-K+1:t}$ and actions $a_{t-K+1:t}$, Anchor-WM uses a transition model $\mathcal{T}_\phi$ to predict the next latent state as

$$\hat{z}_{t+1} = \mathcal{T}_\phi\left(z_{t-K+1:t}, a_{t-K+1:t}\right). \tag{2}$$

We train the encoder and transition model by minimizing the mean squared error between the predicted and target next latent states:

$$\mathcal{L}_{\text{pred}} = \mathbb{E}_{(o_{t-K+1:t}, a_{t-K+1:t}, o_{t+1})\sim\mathcal{D}} \left\|\hat{z}_{t+1} - z_{t+1}\right\|_2^2. \tag{3}$$

Training the world model with only this predictive objective admits a degenerate solution in which the encoder maps every observation to a constant latent and the transition model predicts the same constant. We therefore introduce DINO-guided latent learning to constrain this degeneracy while providing a pretrained visual learning signal.

3.2 DINO-Guided Latent Learning

We guide the latent representation learning using features from a frozen pretrained DINOv2 encoder $\mathcal{E}_{\text{DINO}}(\cdot)$. For each observation o_t, we extract the post-LayerNorm DINO patch features

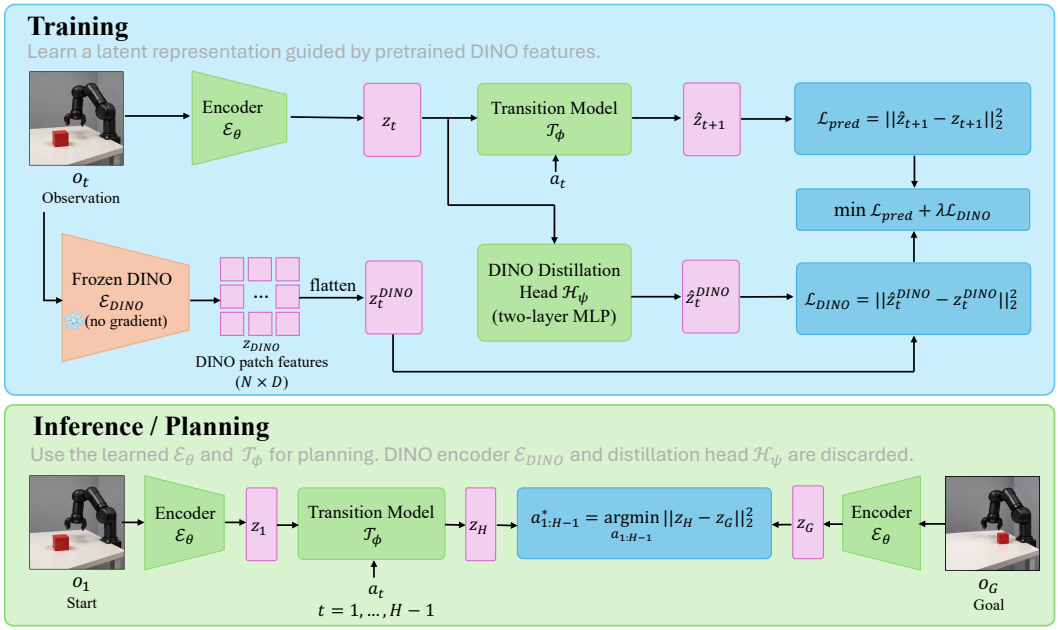

Figure 1: **Training and inference of Anchor-WM.** During training, the observation encoder $\mathcal{E}_\theta$, transition model $\mathcal{T}_\phi$, and distillation head $\mathcal{H}_\psi$ are optimized jointly, while the pretrained DINO encoder remains frozen. The compact latent is trained with both a dynamics prediction objective and a distillation objective that reconstructs DINO features. At inference, the DINO encoder and distillation head are discarded, and planning is performed entirely in the learned latent space by optimizing an action sequence whose predicted terminal latent approaches the encoded goal latent.

$\mathcal{E}_{\text{DINO}}(o_t) \in \mathbb{R}^{N\times D}$ and flatten them in their fixed spatial order to form the distillation target $z_t^{\text{DINO}} \in \mathbb{R}^{ND}$, where N is the number of patches, and D is the embedding dimension. We retain the full patch-level representation rather than pooling it into a global target to retain spatially localized visual information. We validate this design choice against global and patch-wise distillation variants in Sec. 4.4.

Rather than using z_t^{DINO} as the world-model state like DINO-WM, we use it only as a target for representation learning. We introduce a lightweight two-layer MLP distillation head with GELU activation, $\mathcal{H}_\psi : \mathbb{R}^d \to \mathbb{R}^{ND}$, which maps the compact latent z_t to the flattened DINO target space:

$$\hat{z}_t^{\text{DINO}} = \mathcal{H}_\psi(z_t). \tag{4}$$

This design confines the high-dimensional patch representation to the training objective: the world model itself continues to predict and plan using a single compact latent state.

We apply the distillation in Eq 4 to every observation in the sampled context window. Thus, each latent state is trained to regress onto the post-LayerNorm DINO patch features of its corresponding observation by minimizing

$$\mathcal{L}_{\text{DINO}} = \mathbb{E}_{o_t\sim\mathcal{D}}\|\mathcal{H}_\psi(z_t) - z_t^{\text{DINO}}\|_2^2. \tag{5}$$

The distillation objective encourages DINO features z_t^{DINO} to remain recoverable from the compact latent z_t without directly constraining z_t to match z_t^{DINO}. A learnable distillation head maps z_t into the DINO feature space, allowing z_t to have a different dimensionality and internal structure as long as z_t^{DINO} can be predicted from it. This objective also discourages representational collapse: if the encoder mapped every observation to the same latent, the distillation head would receive the same input for all observations and could only produce a constant prediction. Therefore, minimizing the distillation objective requires the encoder to produce observation-dependent latents.

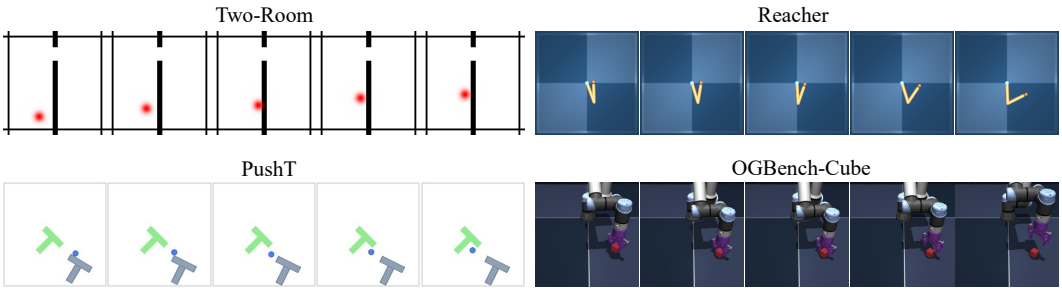

Figure 2: **Evaluation Environments.** Top left: Two-Room for 2D navigation. Top right: Reacher for planar two-link arm control. Bottom left: Push-T for contact-based 2D object manipulation. Bottom right: Cube for 3D robotic cube manipulation. Further details are given in Section B.

3.3 Training Anchor-WM

To train Anchor-WM, we jointly optimize the predictive and distillation objectives as

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda \mathcal{L}_{\text{DINO}}, \tag{6}$$

where λ controls the strength of the distillation objective. Both objective terms shape the learned encoder $\mathcal{E}_\theta$: prediction encourages the latent to capture information useful for modeling action-conditioned dynamics, while distillation encourages it to retain visual information represented by the pretrained DINO features. The transition model $\mathcal{T}_\phi$ and distillation head $\mathcal{H}_\psi$ are optimized by their respective objectives, while the DINO encoder $\mathcal{E}_{\text{DINO}}$ remains frozen throughout training.

3.4 Planning with Anchor-WM

At inference, the encoder $\mathcal{E}_\theta$ maps the observed context into the latent space, and the transition model $\mathcal{T}_\phi$ autoregressively rolls out future latent states under candidate action sequences. Since observations beyond time t are unavailable, the context window is updated with predicted latent states as the rollout proceeds. We select actions by approximately solving

$$a^*_{t:t+H-1} = \arg\min_{a_{t:t+H-1}} \mathcal{C}(\hat{z}_{t+1:t+H}), \tag{7}$$

where $\hat{z}_{t+1:t+H}$ is obtained by rolling out $\mathcal{T}_\phi$ and $\mathcal{C}$ denotes the planning objective.

For the goal-reaching tasks considered in this work, we minimize the distance between the predicted terminal latent $\hat{z}_{t+H}$ and the latent representation of the goal observation o^{goal}:

$$\mathcal{C}(\hat{z}_{t+1:t+H}) = \left\| \hat{z}_{t+H} - \mathcal{E}_\theta\left(o^{\text{goal}}\right) \right\|_2^2. \tag{8}$$

We approximately solve Eq. 7 using the Cross-Entropy Method (CEM) (Rubinstein & Kroese, 2004). CEM iteratively samples candidate action sequences, evaluates their predicted trajectories, and refines the sampling distribution toward lower-cost candidates. We execute the selected action sequence for the full planning horizon and then replan from the newly observed state. Planning horizons and CEM hyperparameters are given in Section C of the Appendix.

Because planning rolls out a single compact latent rather than a grid of DINO patch features, its computational cost is independent of the dimensionality of the teacher representation. This is particularly advantageous for sampling-based planning, which repeatedly evaluates many candidate action sequences. We quantify the resulting difference in planning time in Sec 4.2.

4 Experiments

4.1 Experimental Setup

Environments. We evaluate Anchor-WM on TwoRoom (Sobal et al., 2025), Reacher (Tassa et al., 2018), Push-T (Chi et al., 2025), and OGBench-Cube (Park et al., 2025) as illustrated in Fig. 2, with further details provided in the Appendix.

Table 1: **Planning Performance Comparison**. Success Rate (↑) across four environments, with the final column reporting the average across tasks. Bold indicates the highest mean success rate in each environment.

Method	Latent Size	TwoRoom	Reacher	PushT	Cube	Average
Predictive	1×192	35.2 ± 2.2	10.6 ± 5.3	3.2 ± 1.1	46.5 ± 2.9	23.9
PLDM	1×192	93.0 ± 2.0	69.5 ± 5.8	71.8 ± 6.1	51.2 ± 1.6	71.4
LeWM	1×192	86.1 ± 1.8	78.4 ± 5.7	$\mathbf{90.4 \pm 4.3}$	64.8 ± 2.1	79.9
DINO-WM	196×384	$\mathbf{99.6 \pm 0.9}$	67.2 ± 1.8	65.2 ± 6.6	$\mathbf{74.4 \pm 2.6}$	76.6
DINO-WM (CLS)	1×384	59.3 ± 1.7	45.2 ± 3.4	37.8 ± 5.6	50.2 ± 3.8	48.1
Anchor-WM (ours)	1×192	$\mathbf{99.6 \pm 0.9}$	$\mathbf{82.4 \pm 4.3}$	84.8 ± 5.4	72.0 ± 4.2	**84.7**

Baselines. We compare Anchor-WM with several baselines. *Prediction-Only* is a latent world model trained solely with a prediction objective, without any additional regularization, serving as an ablation baseline. *PLDM* (Sobal et al., 2025) and *LeWM* (Maes et al., 2026) are JEPA-based world models that further employs statistical regularization to mitigate representational collapse. *DINO-WM* (Zhou et al., 2025) instead uses frozen DINO patch features, thereby avoiding representation collapse. We focus on learning world models solely from visual observations, without using additional modalities such as proprioceptive inputs. We therefore evaluate all methods under the same visual-only setting to ensure a fair comparison.

Architecture. We instantiate the encoder $\mathcal{E}_\theta$ as a ViT-Tiny (patch size 14, hidden width 192, 12 layers, and 3 attention heads) and the transition model $\mathcal{T}_\phi$ as a Transformer (6 layers, 16 attention heads, hidden width 2048, head dimension 64, and dropout 0.1), with action conditioning via AdaLN following (Maes et al., 2026). The Anchor-WM latent state, $z_t \in \mathbb{R}^{192}$, is obtained by projecting the CLS token produced by the ViT-Tiny encoder. To distill DINO features, we use a lightweight two-layer MLP distillation head $\mathcal{H}_\psi$ that predicts the flattened DINOv2 patch features, $\mathbb{R}^{196\times384}$, from z_t. Additional architectural details are provided in the Appendix.

4.2 Planning Evaluation

Planning Results. Table 1 reports the planning performance of Anchor-WM and the baselines, where the goal is selected 25 time steps after the start state along an expert trajectory, with a planning budget of 50 steps. Anchor-WM achieves the highest success rate on Reacher and ties DINO-WM (patch) for the best performance on TwoRoom, while remaining competitive on PushT and Cube. Although LeWM performs best on PushT and DINO-WM (patch) performs best on Cube, Anchor-WM attains the highest average success rate across all four environments. These results suggest that using DINO features as training-time supervision enables Anchor-WM to learn compact, task-specific representations that support strong visual planning performance across diverse environments.

We also evaluate Anchor-WM and the baselines under longer-horizon planning, where the goal observation is selected 50 timesteps after the start observation and the evaluation budget is increased to 100 steps. Table 2 reports the corresponding success rates. Anchor-WM achieves the highest success rate on TwoRoom, Reacher, and PushT, as well as the highest average success rate across all environments, while DINO-WM performs best on Cube. These results suggest that guiding a compact, task-specific latent representation with pretrained DINO supervision remains effective for longer-horizon planning.

We further evaluate the planning performance of Anchor-WM in the zero-shot setting, where at inference time we introduce visual perturbations that are not observed during training, such as changes in

Table 2: **Long-horizon planning performance.** Planning success rates when the goal observation is selected 50 timesteps after the start observation and the evaluation budget is 100 steps.

Method	TwoRoom	Reacher	PushT	Cube	Average
LeWM	79.2±3.6	95.2±1.8	41.6±7.1	44.8±8.3	65.2
DINO-WM	96.0±2.5	90.0±5.7	32.0±2.8	**53.2±5.0**	67.8
Anchor-WM (ours)	**96.4±1.7**	**96.4±3.3**	**46.4±3.6**	45.2±2.7	**71.1**

Table 3: **Zero-shot Planning Performance.** Anchor-WM and the baselines are trained on the original environments shown in Fig. 2. At test time, we introduce unseen visual perturbations to evaluate zero-shot generalization, such as changing the end-effector or target-object color.

Method	TwoRoom	Reacher	PushT	Cube	Average
LeWM	62.8±8.2	56.4±8.4	10.8±3.0	64.0±4.0	48.5
DINO-WM	98.0±2.4	**60.8±7.0**	**62.8±4.8**	**76.4±3.6**	**74.5**
Anchor-WM (ours)	**99.2±1.1**	47.2±10.8	33.2±4.8	68.0±6.5	61.9

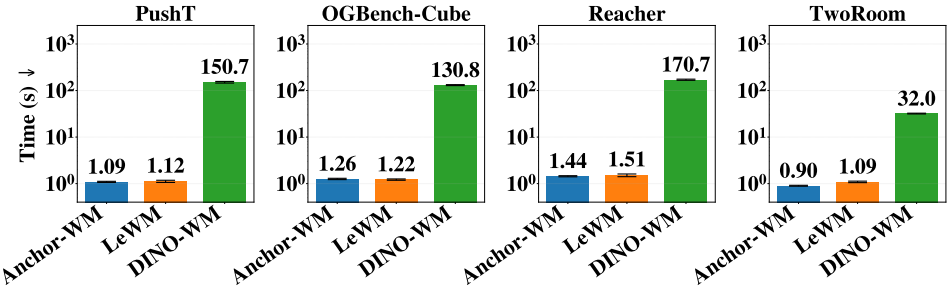

Figure 3: **Planning Time Comparison.** Anchor-WM achieves planning times comparable to LeWM while being substantially faster than DINO-WM across all four environments.

the end-effector or object color. Examples of these perturbations are illustrated in Fig. 9. As shown in Table 3, Anchor-WM substantially improves over LeWM under visual perturbations and achieves encouraging zero-shot generalization, while DINO-WM remains stronger on average. These results suggest that training-time DINO supervision transfers useful visual robustness to the compact latent representation. Although DINO-WM attains a higher average success rate, its planning is approximately $100\times$ slower than Anchor-WM and LeWM.

Additionally, we report planning efficiency in Figure 3. Anchor-WM achieves substantially faster planning than DINO-WM, while maintaining planning speed comparable to LeWM. This efficiency stems from performing planning in a compact learned latent space rather than directly over the high-dimensional DINO feature space.

These results show that Anchor-WM combines strong planning performance across standard, long-horizon, and zero-shot settings with efficient inference. By using DINO features as training-time signal and planning in a compact latent space, Anchor-WM retains the benefits of pretrained visual features without incurring the high planning cost of operating directly in the DINO feature space.

4.3 Physical Understanding

Probing Results. We probe the latent representations to assess whether they preserve task-relevant physical information. We train linear and MLP probes on frozen latent representations to predict ground-truth physical state variables, and evaluate performance using mean squared error (MSE) and Pearson correlation r. On OGBench-Cube, Anchor-WM preserves substantial information about several task-relevant variables. Its linear probe achieves $r = 0.734$ for joint position and $r = 0.997$ for end-effector position, while the MLP probe reaches $r = 0.998$ for block position. For joint velocity, nonlinear probing improves Anchor-WM from $r = 0.254$ to $r = 0.427$. Across all variables, Anchor-WM achieves an overall linear/MLP correlation of $0.521/0.577$, compared with $0.492/0.496$ for LeWM and $0.545/0.560$ for DINO-WM. Thus, DINO-WM achieves the highest overall linear correlation, while Anchor-WM achieves the highest overall MLP correlation among the evaluated methods. These results suggest that DINO distillation transfers task-relevant physical and geometric information into the compact latent space used for dynamics prediction and planning. Additional probing results are provided in the Appendix.

Latent Space Decoding. To qualitatively assess the information captured by the learned latent representation, we decode the open-loop predicted latents back into observation space. Figure 4 presents representative examples from PushT and OGBench-Cube. In PushT, the decoded predictions accurately preserve the positions and orientations of task-relevant objects over the rollout and remain closely aligned with the corresponding ground-truth observations. In the visually more com-

Table 4: **Probing the Latent Space.** Probing results on OGBench-Cube. MSE (↓) and Pearson correlation r (↑) for linear and MLP probes. Best mean among all methods per metric in bold.

Property	Model	Linear MSE ↓	Linear r ↑	MLP MSE ↓	MLP r ↑
Joint Position	LeWM	$0.216 \pm .003$	$0.702 \pm .002$	$0.212 \pm .002$	$0.705 \pm .003$
	DINO-WM	$0.193 \pm .003$	$\mathbf{0.736 \pm .002}$	$0.203 \pm .004$	$\mathbf{0.741 \pm .004}$
	Anchor-WM (ours)	$\mathbf{0.192 \pm .003}$	$0.734 \pm .002$	$\mathbf{0.192 \pm .003}$	$0.740 \pm .003$
Joint Velocity	LeWM	$0.987 \pm .011$	$0.140 \pm .005$	$0.997 \pm .011$	$0.117 \pm .009$
	DINO-WM	$\mathbf{0.883 \pm .011}$	$\mathbf{0.315 \pm .004}$	$0.861 \pm .015$	$0.357 \pm .003$
	Anchor-WM (ours)	$0.931 \pm .011$	$0.254 \pm .005$	$\mathbf{0.802 \pm .018}$	$\mathbf{0.427 \pm .006}$
End-Eff. Position	LeWM	$0.017 \pm .001$	$0.991 \pm .000$	$0.003 \pm .000$	$0.998 \pm .000$
	DINO-WM	$0.008 \pm .000$	$0.996 \pm .000$	$0.003 \pm .000$	$0.998 \pm .000$
	Anchor-WM (ours)	$\mathbf{0.006 \pm .000}$	$\mathbf{0.997 \pm .000}$	$\mathbf{0.001 \pm .000}$	$\mathbf{0.999 \pm .000}$
End-Eff. Yaw	LeWM	$1.000 \pm .014$	$0.064 \pm .010$	$1.010 \pm .012$	$0.093 \pm .018$
	DINO-WM	$\mathbf{0.893 \pm .012}$	$\mathbf{0.331 \pm .003}$	$\mathbf{0.583 \pm .012}$	$\mathbf{0.647 \pm .005}$
	Anchor-WM (ours)	$0.902 \pm .012$	$0.315 \pm .004$	$0.907 \pm .012$	$0.316 \pm .010$
Gripper	LeWM	$\mathbf{0.420 \pm .014}$	$\mathbf{0.640 \pm .002}$	$\mathbf{0.388 \pm .014}$	$\mathbf{0.662 \pm .003}$
	DINO-WM	$0.528 \pm .014$	$0.585 \pm .003$	$0.522 \pm .021$	$0.594 \pm .007$
	Anchor-WM (ours)	$0.562 \pm .013$	$0.563 \pm .003$	$0.447 \pm .018$	$0.628 \pm .007$
Block Position	LeWM	$\mathbf{0.007 \pm .000}$	$\mathbf{0.996 \pm .000}$	$\mathbf{0.002 \pm .000}$	$\mathbf{0.999 \pm .000}$
	DINO-WM	$0.046 \pm .001$	$0.976 \pm .001$	$0.007 \pm .000$	$0.996 \pm .000$
	Anchor-WM (ours)	$0.058 \pm .001$	$0.970 \pm .001$	$0.004 \pm .000$	$0.998 \pm .000$
Block Quaternion	LeWM	$\mathbf{0.919 \pm .116}$	$0.061 \pm .010$	$0.935 \pm .106$	$\mathbf{0.073 \pm .020}$
	DINO-WM	$0.926 \pm .115$	$\mathbf{0.065 \pm .007}$	$0.934 \pm .115$	$0.050 \pm .007$
	Anchor-WM (ours)	$0.922 \pm .116$	$0.044 \pm .009$	$\mathbf{0.923 \pm .116}$	$0.035 \pm .010$
Block Yaw	LeWM	$1.003 \pm .013$	$0.040 \pm .012$	$1.021 \pm .011$	$0.068 \pm .009$
	DINO-WM	$\mathbf{0.989 \pm .013}$	$\mathbf{0.134 \pm .007}$	$1.005 \pm .010$	$\mathbf{0.114 \pm .003}$
	Anchor-WM (ours)	$0.999 \pm .012$	$0.064 \pm .008$	$\mathbf{1.004 \pm .013}$	$0.070 \pm .011$
Overall	LeWM	$0.527 \pm .018$	$0.492 \pm .002$	$0.533 \pm .015$	$0.496 \pm .002$
	DINO-WM	$\mathbf{0.511 \pm .018}$	$\mathbf{0.545 \pm .002}$	$0.513 \pm .019$	$0.560 \pm .003$
	Anchor-WM (ours)	$0.526 \pm .019$	$0.521 \pm .002$	$\mathbf{0.485 \pm .018}$	$\mathbf{0.577 \pm .003}$



Figure 4: **Open-Loop Rollouts on PushT and Cube.** DINO-WM produces visually sharper reconstructions, whereas Anchor-WM discards some fine-grained appearance information. Nevertheless, Anchor-WM preserves the task-relevant evolution of object configurations and interactions. Considering the planning results in Table 1 and Table 2, these examples suggest that visual reconstruction fidelity is not necessarily indicative of a world model's utility for planning.

plex Cube environment, the predictions likewise retain the overall scene structure and capture the temporal evolution of both the robot and the manipulated object. These results suggest that the compact latent representation preserves task-relevant visual information and that the learned dynamics propagate this information coherently over extended open-loop rollouts.

Table 5: **Ablation Results.** We evaluate the effects of the distillation objective, distillation target, and DINO guidance strategy across all the environments by reporting planning success rate (↑).

Ablation	Setting	TwoRoom	Reacher	PushT	Cube
Distillation Objective	MSE	99.6 ± 0.9	82.4 ± 4.3	84.8 ± 5.4	72.0 ± 4.2
	Cosine	89.2 ± 1.8	78.6 ± 3.1	79.5 ± 6.8	67.0 ± 3.8
Distillation Target	Patch	99.6 ± 0.9	82.4 ± 4.3	84.8 ± 5.4	72.0 ± 4.2
	Global	83.8 ± 2.4	54.7 ± 3.7	64.3 ± 3.3	49.6 ± 5.2
DINO Guidance	Distillation	99.6 ± 0.9	82.4 ± 4.3	84.8 ± 5.4	72.0 ± 4.2
	Alignment	50.0 ± 5.7	48.1 ± 2.3	42.7 ± 5.1	50.4 ± 3.9

4.4 Ablation

Distillation Objective Ablation. In Eq. 5, we use an MSE objective to match the reconstructed DINO features to the target features. As an alternative, we consider cosine similarity, which measures angular agreement while being insensitive to magnitude, thereby relaxing the constraint imposed by the DINO target and potentially allowing greater flexibility in learning environment-specific dynamics. Table 5 shows that MSE consistently achieves higher planning success rates across all environments. This suggests that preserving both the direction and magnitude of the DINO features provides a more effective training signal than matching directional similarity alone.

Distillation Target Ablation. In Eq. 5, the DINO target can be either the spatial patch features or the global CLS representation. We ablate this choice in Table 5. Using patch-level DINO features yields substantially higher planning success rates across all environments. This suggests that spatially distributed patch features provide richer supervision than a single global representation, helping Anchor-WM preserve task-relevant visual and geometric information during distillation.

DINO Guidance Injection. We also study how DINO supervision is incorporated into representation learning. Anchor-WM reconstructs DINO features from the final compact latent through a distillation head, while an alternative following REPA (Yu et al., 2025) directly aligns layer-8 encoder features with DINO features. Table 5 shows that distillation consistently outperforms intermediate-feature alignment, suggesting that supervising the final latent through a dedicated reconstruction objective is more effective for our model.

We additionally ablate the sensitivity of DINO distillation weights in Fig. 7 and the influence of latent dimension in Fig. 8 in the Section A of the Appendix.

5 Conclusion and Limitations

Conclusion. We introduced Anchor-WM, a latent world model that distills pretrained dense DINO representations into a compact latent space for visual planning. This design combines the semantic structure of pretrained visual features with environment-specific dynamics learning, while avoiding representational collapse. Anchor-WM also allows efficient planning by using the learned compact latent instead of the high-dimensional DINO features. Across the evaluated environments, Anchor-WM achieves success rates that are competitive with or higher than existing latent world-model baselines and obtains the highest average success rate overall.

Limitations. Anchor-WM relies on a pretrained DINO encoder as a training-time teacher and may therefore inherit biases or limitations present in its representation. Exploring alternative pretrained models and more adaptive teacher–student objectives is a promising direction for future work. Furthermore, our evaluation is limited to a relatively small set of offline visual control environments. Extending the evaluation to longer-horizon tasks and real-world robotic settings would provide a more comprehensive assessment of the method.

6 AI DISCLOSURE

In this work, we used generative AI to provide feedback on research methodology and experimental design. We did not use generative AI tools to generate synthetic datasets; develop theoretical models or conceptual frameworks; formulate mathematical claims; provide critical ingredients for proving mathematical claims; assist in writing proofs; propose or refine hypotheses; implement methods; assist with translation; clean or reformat datasets; support qualitative or thematic data analysis; or interpret results. Additionally, we used generative AI tools to polish portions of the writing, implement portions of the code, and analyze existing literature. We carefully checked all AI-polished text and reviewed all AI-assisted work. For example, LLM-generated code was independently verified and tested for correctness by two authors. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the assistance of generative AI.

REFERENCES

Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D. Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

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

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.

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. *arXiv preprint arXiv:2105.04906*, 2021.

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.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019.

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor world models: Perception for action via inverse dynamics. *arXiv preprint arXiv:2606.20104*, 2026.

Yann LeCun. A path towards autonomous machine intelligence. *Open Review*, 2022.

Jongmin Lee, Yoonwoo Jeong, Seungwook Kim, Juhong Min, and Minsu Cho. Learning to distill convolutional features into compact local descriptors. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 898–908, 2021.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations*, 2023.

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

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

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 892–909. PMLR, 2023.

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

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

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Lu Qiu, Yizhuo Li, Yi Chen, Yuying Ge, Yixiao Ge, and Xihui Liu. Making foresight actionable: Repurposing representation alignment in world action models. *arXiv preprint arXiv:2606.12217*, 2026.

Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 416–426. PMLR, 2023.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015.

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.

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

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ 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.

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

Basile Terver, Randall Balestriero, Megi Dervishi, David Fan, Quentin Garrido, Tushar Nagarajan, Koustuv Sinha, Wancong Zhang, Mike Rabbat, Yann LeCun, and Amir Bar. A lightweight library for energy-based joint-embedding predictive architectures. *arXiv preprint arXiv:2602.03604*, 2026.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

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

Junyuan Xiao, Dingkang Liang, Xin Zhou, Yixuan Ye, Tongtong Su, Guangmo Yi, Bin Xia, Qiang Lyu, Shurui Shi, Jun Huang, Jianlou Si, and Wenming Yang. Divide and conquer: Decoupled representation alignment for multimodal world models. *arXiv preprint arXiv:2605.01896*, 2026.

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.

Zhenghao Zhang, Yuanxiang Wang, Zhenyu Guan, Yujia Yang, Bingkang Shi, Tianyu Zong, Hongzhu Yi, Guoqing Chao, Xingchen Chen, Tiankun Yang, Chenxi Bao, Tao Yu, Jingjing Zhou, and Jungang Xu. Delta-JEPA: Learning action-sensitive world models via latent difference decoding. *arXiv preprint arXiv:2606.31232*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025.

Appendix

In this Appendix, we provide additional experimental results in Section A. We further provide details on the dataset used to train Anchor-WM in Section B. Finally, we provide the model architecture, training and planning details of Anchor-WM in Section C.

A Additional Results

Additional Probing Results. We provide additional probing results on PushT in Table 6. Anchor-WM preserves substantial information about the agent and object state despite representing each observation with a single latent token. For agent location, Anchor-WM achieves the lowest linear-probe MSE and nearly perfect recovery with the MLP probe ($r = 0.997$). For block location, its linear probe is weaker than LeWM and DINO-WM, while the MLP probe reaches $r = 0.997$, indicating that the information is readily recoverable through a nonlinear mapping. A similar trend is observed for block angle: although linear probing is relatively weak, the MLP probe reaches $r = 0.984$, outperforming both LeWM and DINO-WM. Overall, Anchor-WM improves from $r = 0.952$ with the linear probe to $r = 0.994$ with the MLP probe, suggesting that the compact latent retains substantial physical information, some of which is encoded nonlinearly.

Additional Open-loop Rollouts. Figure 5 presents additional open-loop rollout examples for PushT and OGBench-Cube. Across multiple trajectories, the decoded predictions from Anchor-WM remain qualitatively aligned with the corresponding ground-truth observations over the rollout horizon. In PushT, the predicted latents preserve the motion and orientation of the manipulated object, while in OGBench-Cube they capture the evolution of the robot and cube despite the greater visual complexity of the scene. These examples further illustrate that the learned compact latent representation retains task-relevant visual and dynamical information over open-loop prediction.

Latent Space Visualization. We further examine whether the learned latent representation reflects the underlying physical structure of the environment. Figure 6 visualizes a grid of PushT physical states together with a two-dimensional projection of their corresponding latent representations. The projected latents form a structured and smoothly varying manifold: neighboring physical configurations tend to appear near one another, and changes in physical state are accompanied by gradual transitions in the latent space. This qualitative organization suggests that Anchor-WM preserves meaningful geometric relationships between physical states rather than mapping observations to an unstructured representation.

Sensitivity to the Distillation Weight. We study the sensitivity of Anchor-WM to the weight λ applied to the DINO distillation objective on OGBench-Cube. As shown in Fig. 7, removing the distillation objective ($\lambda = 0$) leads to a substantial drop in planning success rate. In contrast, performance improves markedly for nonzero values of λ and remains relatively stable over a broad range of settings. This indicates that the benefit of DINO supervision does not depend on carefully tuning the distillation weight, and that Anchor-WM is reasonably robust to the choice of λ.

Sensitivity to Latent Dimensionality. We evaluate the effect of the compact latent dimensionality on planning performance in OGBench-Cube. As shown in Fig. 8, performance varies only slightly when the latent dimension is changed from 64 to 384, with the 192-dimensional representation achieving the highest success rate. Increasing the latent size to 384 does not provide additional improvement, while even a 64-dimensional latent retains similar performance. These results suggest that the planning performance of Anchor-WM is not highly sensitive to the latent dimensionality within the evaluated range and that strong performance can be achieved with a compact representation.

B Datasets

We evaluate Anchor-WM on a set of continuous-control environments spanning navigation and robotic manipulation. For every environment, we reserve 10% of the collected trajectories as a held-out test set and use the remaining 90% for training. Our evaluation includes both in-distribution

(ID) settings, where test trajectories follow the same environment configuration and data distribution as the training data, and out-of-distribution (OOD) settings, where the model is evaluated under modified visual or task conditions not observed during training.

TwoRoom. TwoRoom is a continuous 2D navigation environment consisting of two regions connected by a narrow doorway. The agent must move from an initial position to a goal located in the opposite room, making successful trajectories depend on navigating through the doorway. We use a dataset of 10,000 trajectories generated by a noisy waypoint-based policy that first guides the agent toward the doorway and subsequently toward the sampled goal.

PushT. PushT is a planar manipulation task in which a point-like agent interacts with a T-shaped object through pushing actions. The objective is to move and orient the object so that it matches a desired target configuration. We use the standard PushT dataset containing 20,000 expert demonstrations, following the data setup commonly used in prior visual world-model work.

OGBench-Cube. OGBench-Cube is a 3D robotic manipulation task in which a robot arm must grasp a cube and transport it to a specified target position. We use the single-object version of the environment and collect 10,000 trajectories of 200 environment steps using the benchmark's provided data-collection policy.

Reacher. Reacher is a continuous-control task from the DeepMind Control Suite in which a two-link arm must position its end effector at a target location in the plane. We use 10,000 trajectories of 200 steps collected with a Soft Actor-Critic policy. Following prior world-model evaluations, success requires the arm to reach the target configuration with high precision.

ID and OOD evaluation. The held-out 10% split is used for ID evaluation, providing unseen trajectories from the same data-generating process as the training set. OOD evaluation instead measures performance under visual perturbation as illustrated in Fig. 9, so that the model is tested on observations or task configurations that differ from those encountered during training.

C Architecture, Training, and Planning

We use a Vision Transformer (ViT-Tiny, patch size 14; (Dosovitskiy et al., 2020)), trained from scratch, to parameterize the Anchor-WM encoder. Each 224×224 observation is mapped to a single 192-dimensional latent representation given by the `[CLS]` token. Actions are z-scored independently in each dimension, grouped over the frame-skip interval of 5 environment steps, and embedded into the same 192-dimensional space.

The dynamics model is a 6-layer causal Transformer with 16 attention heads and an MLP dimension of 2048. It is conditioned on the action embeddings using adaptive layer normalization (AdaLN-zero; (Peebles & Xie, 2023)). Given a history of 3 latent frames, the model autoregressively predicts the next latent representation and is trained using a mean-squared error loss on the predicted next embedding.

To anchor the learned latent space, we additionally train a 2-layer MLP head to reconstruct the final-layer patch tokens of a frozen DINOv2 ViT-S/14 teacher (Oquab et al., 2024). The teacher features are computed from images at 196×196 resolution, and their reconstruction from the Anchor-WM latent is optimized with a mean-squared error loss weighted by λ. This auxiliary objective replaces the anti-collapse regularizer used in LeWM (Maes et al., 2026). The DINOv2 teacher and reconstruction head are used only during training and discarded at inference, so planning relies only on the compact encoder and dynamics predictor.

We learn a decoder only for visualization purpose. Anchor-WM is trained without any pixel reconstruction, so to visualize its latents we train a separate decoder post hoc, after world-model training. The world-model encoder is frozen and kept in evaluation mode; only the decoder is optimized, and it is never used during world-model training or planning. The decoder maps the 192-dimensional latent to a 224×224 RGB image: a linear layer projects the latent to a $256 \times 7 \times 7$ feature map, followed by five upsampling stages, each consisting of bilinear $2\times$ upsampling and two 3×3 convolutions with group normalization and GELU activations (channels $256 \to 128 \to 64 \to 32 \to 16 \to 16$),

and a final 3×3 convolution with a sigmoid output ($\sim$3.0M parameters in total). It is trained with a pixel-wise mean-squared error on RGB values in $[0, 1]$ using AdamW (learning rate 3×10^{-4}, weight decay 10^{-4}, batch size 64) for 20,000 steps on frames sampled every 5 environment steps from the training data, and we keep the checkpoint with the lowest reconstruction error on 1,024 frames from episodes held out from decoder training. For open-loop rollouts, the encoder embeds the context frames, the predictor rolls the latent forward under the ground-truth actions, and each predicted latent is decoded independently. For comparison, LeWM decoders are trained with the identical architecture and protocol.

All world models are evaluated using the same planner and evaluation protocol. We use the cross-entropy method (CEM) with 300 sampled action sequences, 30 optimization iterations, and the top 30 samples as elites. Each latent step corresponds to a block of 5 environment actions (frame skip 5), and plans span 5 latentsteps, i.e., 25 environment steps. We execute the full planned sequence and then replan from the currentobservation (receding horizon of 5 latent steps), resulting in two planning calls per 50-step episode. Planning is performed in a receding-horizon manner from the current observation. The planning objective minimizes the squared distance between the final predicted latent and the latent representation of the goal observation.

For standard planning evaluation, the goal is selected 25 environment steps after the start state along an expert trajectory, and each episode has a budget of 50 environment steps. For long-horizon evaluation, the goal is selected 50 environment steps ahead and the budget is increased to 100 environment steps. Start–goal pairs are sampled from the 10% of episodes held out from training. An episode is considered successful if the agent reaches the goal within the evaluation budget. We report the mean success rate over 5 random seeds, with 50 evaluation episodes per seed. We additionally report planning time as a measure of inference efficiency, with all methods evaluated on a single NVIDIA A100 GPU.

The model, training, and planning hyperparameters of Anchor-WM are summarized in Table 7.

Table 6: Probing results on PushT. MSE (↓) and Pearson correlation r (↑) for linear and MLP probes. Values are mean ± standard deviation over five splits. Best of {Anchor-WM, LeWM} per metric in bold.

Property	Model	Linear MSE ↓	Linear r ↑	MLP MSE ↓	MLP r ↑
Agent Location	LeWM	0.060 ± .002	0.970 ± .001	**0.005 ± .001**	**0.998 ± .001**
	DINO-WM	0.031 ± .002	0.984 ± .001	0.012 ± .003	0.994 ± .001
	Anchor-WM (ours)	**0.027 ± .002**	**0.987 ± .001**	0.007 ± .002	0.997 ± .001
Block Location	LeWM	**0.028 ± .001**	**0.986 ± .000**	**0.001 ± .000**	**0.999 ± .000**
	DINO-WM	0.026 ± .001	0.987 ± .000	0.008 ± .001	0.996 ± .000
	Anchor-WM (ours)	0.043 ± .002	0.978 ± .001	0.005 ± .001	0.997 ± .000
Block Angle	LeWM	**0.173 ± .005**	**0.909 ± .005**	0.038 ± .004	0.981 ± .002
	DINO-WM	0.216 ± .003	0.885 ± .002	0.036 ± .005	0.982 ± .002
	Anchor-WM (ours)	0.312 ± .009	0.828 ± .004	**0.032 ± .004**	**0.984 ± .002**
Overall	LeWM	**0.070 ± .001**	**0.964 ± .001**	**0.011 ± .001**	**0.995 ± .001**
	DINO-WM	0.066 ± .001	0.965 ± .000	0.019 ± .001	0.991 ± .001
	Anchor-WM (ours)	0.090 ± .003	0.952 ± .001	0.013 ± .001	0.994 ± .000

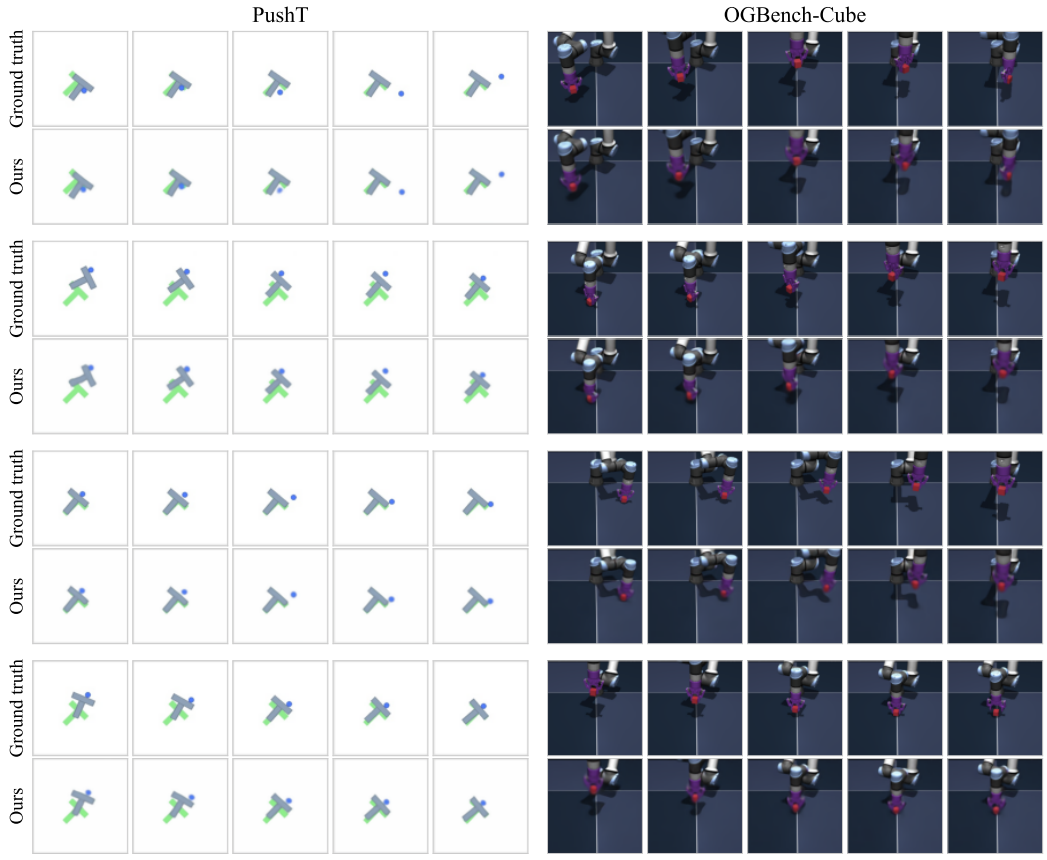

Figure 5: **Additional open-loop rollout results.** Additional qualitative rollouts on PushT and OGBench-Cube. Each pair of rows compares ground-truth observations with observations decoded from the open-loop latent predictions of Anchor-WM.

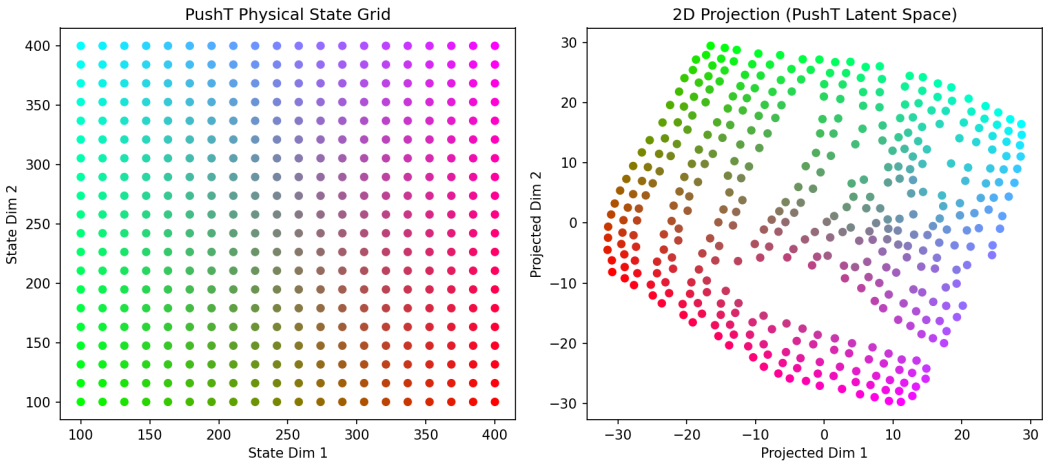

Figure 6: **Latent-space organization in PushT.** Left: a grid of physical states in PushT, with color indicating the state location. Right: a two-dimensional projection of the corresponding learned latent representations, using the same color assignment. The projected latents exhibit a smooth and coherent organization, with nearby physical states generally remaining close in the learned representation.

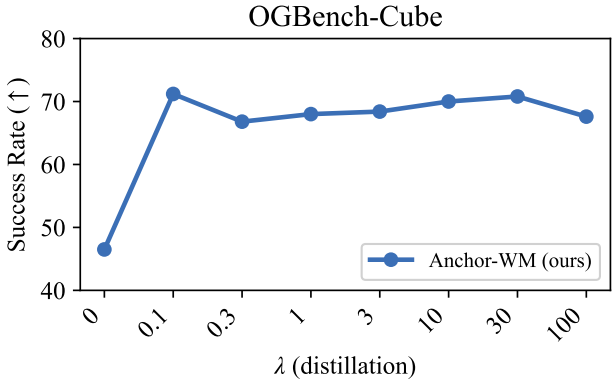

Figure 7: **Effect of the distillation weight λ.** Planning success rate of Anchor-WM on OGBench-Cube as the weight λ of the DINO distillation objective is varied. Setting $\lambda = 0$ removes the distillation objective. Performance improves substantially when distillation is introduced and remains relatively stable across a broad range of nonzero values, indicating that Anchor-WM is not highly sensitive to the precise choice of λ.

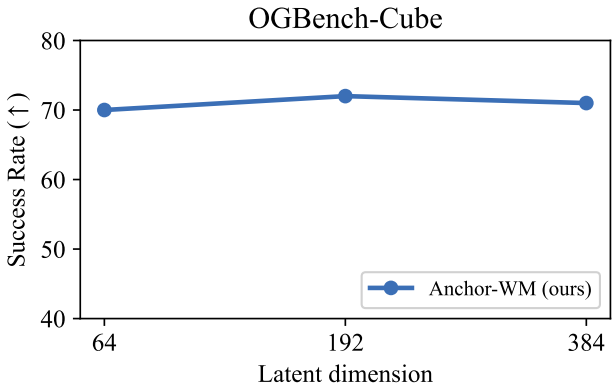

Figure 8: **Effect of latent dimensionality.** Planning success rate of Anchor-WM on OGBench-Cube with latent dimensions of 64, 192, and 384. Performance remains similar across all three settings, indicating that Anchor-WM is relatively insensitive to the latent dimensionality within this range.

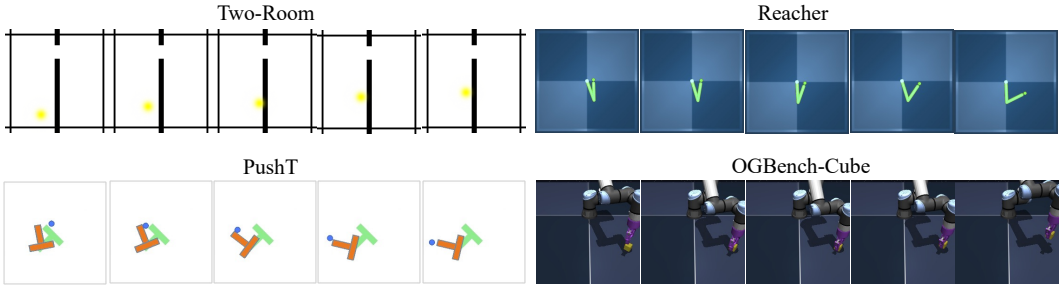

Figure 9: **Out-of-Distribution Environment Illustration.** Top Left: Two-Room, we change the agent color from red to yellow. Top-Right: Reacher, we change the arm color from blue to green. Bottom Left: Push-T, we change the object color from grey to orange. Bottom Right: OGBench-Cube, we change the cube color from red to brown.

Table 7: Anchor-WM hyperparameters (shared across environments).

Hyperparameter	**Value**
Encoder	ViT-Tiny (patch size 14), from scratch
Input resolution	224×224
Latent dimension	192 (CLS token)
Predictor	Causal Transformer, 6 layers
Attention heads / head dim	16 / 64
Predictor MLP dimension	2048
Predictor dropout	0.1
Action conditioning	AdaLN-zero
History length	3 frames
Frame skip	5
Distillation teacher	DINOv2 ViT-S/14 (frozen)
Teacher input resolution	196×196
Distillation target	Last-layer patch tokens
Distillation head	2-layer MLP (GELU)
Distillation loss	MSE
Optimizer	AdamW
Learning rate	5×10^{-5}
Weight decay	10^{-3}
Batch size	128
Precision	bfloat16
Gradient clipping	1.0
Planner	CEM
CEM samples / iterations / elites	300 / 30 / 30
Planning horizon	5
Evaluation episodes	50
Goal offset / step budget	25 / 50