DWM: SEPARATING WORLD EFFECTS FROM ACTIONS IN LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models provide an effective way to model environment dynamics for control, but existing action-conditioned formulations typically supervise the next-latent transition using a single prediction target, requiring the model to explain all sources of state change through one unified learning signal. In real-world environments, however, state transitions often arise from two distinct sources: changes caused by the agent's actions and changes induced by the environment's intrinsic dynamics. In this paper, we introduce DWM (Decomposed World Model), a supervision-level framework that explicitly separates these two sources of transition. The action-invariant source can occur even under a null action and includes effects such as gravity-driven sliding, inertia, contact rebound, and persistent drift. Modeling these two sources jointly can entangle their effects in the learned latent transition, making it difficult to attribute observed changes to their underlying causes and limiting the transferability of the learned dynamics. DWM augments a latent world model with an auxiliary world head that captures action-invariant changes and is regularized by a normalized world-contrastive objective. The original prediction head is coupled with the world head through an orthogonality constraint, encouraging the predicted transition to decompose into an action-invariant component and a complementary action-driven component. This formulation requires no modification to the underlying world-model architecture or inference pipeline. To evaluate DWM in environments with persistent world effects, we construct W-variants of three standard control benchmarks, PushT-W, Reacher-W, and TwoRoom-W, each introducing a distinct form of action-invariant dynamics. Across multiple representative latent world models, DWM consistently improves both prediction quality and downstream control performance on these benchmarks.

1 INTRODUCTION

Latent world models have gained increasing attention for model-based planning in learned latent spaces (Hafner et al., 2019; 2020; 2021; 2023; Hansen et al., 2022; 2024). Within this family, joint-embedding predictive architectures (Assran et al., 2023; Bardes et al., 2024) predict future content directly in a learned latent space rather than reconstructing raw signals, offering compact and semantically meaningful representations for downstream control. Recent works such as LeWM (Maes et al., 2026) and DINO-WM (Zhou et al., 2024) learn action-conditioned dynamics directly between latent representations: an encoder maps observations to a latent representation, and an action-conditioned predictor is trained to regress the next-step latent, so that the entire latent transition is supervised by a single next-latent target. These predicted latents are then used for goal-conditioned planning across a range of control tasks, where both methods achieve strong planning performance on standard manipulation and navigation benchmarks.

A physical transition, however, is rarely caused by the agent's action alone. Consider a block sliding on a tilted surface. Part of its motion is produced by the agent's push, while another part would occur even if the agent did nothing, because gravity continues to move the block. We refer to these two sources of change as the **action effect** and the **world effect**, respectively. More precisely, the world effect is the part of the transition that would persist under the same state and history if the current action were replaced by a null action, and the action effect is the remaining change attributable to the chosen action. Similar decompositions arise when objects move because of inertia, articulated bodies swing under gravity, or agents drift under persistent external forces.

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

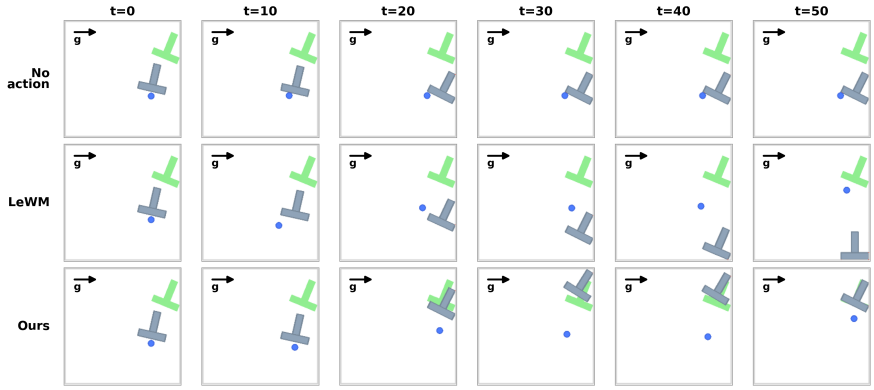

Figure 1: World/action entanglement on PushT-W. The gray T is the current block, the green T is the goal, and the blue circle is the pusher; the arrow g marks the persistent gravity that defines the world effect. **Top:** under a zero action, the block slides along g, isolating the world effect. **Middle:** the single-head LeWM baseline misses this passive motion and drifts away from the goal. **Bottom:** DWM disentangles the two components and brings the block onto the goal.

In current representative methods such as LeWM and DINO-WM, the world effect and the action effect are conflated into a single next-latent target, without any explicit signal to separate them. As a result, the model struggles to isolate what the action actually does, ultimately degrading downstream planning performance. Figure 1 illustrates this issue on PushT-W, a gravity-perturbed variant of the standard PushT task. Under a zero action, the block continues to slide along the direction of gravity. A conventional single-head latent world model fails to reproduce this passive motion accurately, and its predicted trajectory consequently deviates from the goal. These observations motivate the central question of this work:

> *Can latent world models be improved by explicitly separating what the world does on its own from what the agent's action changes?*

We introduce Decomposed World Model (DWM), a world/action-decomposed training framework for action-conditioned latent world models. During training, DWM adds an auxiliary world head alongside the original prediction head; by perturbing the current action while keeping the history fixed, we require the world-head prediction to remain stable across actions yet discriminative across states, so it captures the action-invariant component of the transition. We then decompose the complete prediction into the world-head prediction and its complementary action-driven component, and add an orthogonality regularizer to encourage the two components to encode complementary transition information. Because the auxiliary branch is used only during training, DWM leaves the inference-time architecture and planning pipeline unchanged, and can be integrated with existing latent world models.

To evaluate DWM, we construct three controlled benchmarks, PushT-W, Reacher-W, and TwoRoom-W, by augmenting the original PushT, Reacher, and TwoRoom tasks with gravity-driven sliding, vertical-plane gravity, and constant environmental drift, respectively. We further evaluate DWM on Ball-in-Cup, a more challenging environment in which gravity, string tension, and ball-cup contact jointly give rise to a compound world effect. Across the three W-variants, integrating DWM with LeWM and DINO-WM yields average absolute improvements of 13.1% and 10.2% in CEM planning success, respectively. DWM also retains its advantage under OOD gravity ($+14\%$ on PushT-W), improves planning on Ball-in-Cup, and consistently reduces the autoregressive rollout error across all tasks. Our main contributions are as follows:

- We identify a supervision-level limitation of joint-embedding latent world models trained against a single next-latent target: this target merges changes caused by the agent with changes that would occur without the current action, providing no explicit signal for separating the two sources of transition.

- We introduce a controlled benchmark suite consisting of PushT-W, Reacher-W, and TwoRoom-W, which adds persistent action-invariant dynamics to standard control tasks while retaining their original objectives and flat counterparts as controls.
- We propose DWM, a unified training framework whose auxiliary world head and world/action separation objectives can be integrated with representative latent world models while preserving the original inference pipeline.
- We demonstrate that DWM substantially outperforms the corresponding baselines when integrated with different representative latent world models, and across multiple realistic settings involving diverse world effects.

2 World and Action Effects In Latent Dynamics

2.1 Preliminaries

Action-conditioned latent world model. We consider action-conditioned latent world models such as LeWM (Maes et al., 2026) and DINO-WM (Zhou et al., 2024), together with their recent variants (Gao & Xu, 2026; Huang et al., 2026; Zhao et al., 2026; Thil et al., 2026; Wang et al., 2026; Du et al., 2026). These models operate on trajectories of raw observations o_t (e.g., pixels) and executed actions a_t, and share three components:

- an encoder ϕ that maps each raw observation o_t to a compact latent state $z_t = \phi(o_t)$, i.e., the representation on which all downstream prediction and planning operates;
- an action-conditioned predictor g that summarizes a short context window of past latents and actions $(z_{t-h+1:t},\, a_{t-h+1:t})$ into an intermediate rollout state $r_t = g(z_{t-h+1:t},\, a_{t-h+1:t})$, where h is the context length; r_t can be read as "what the model believes about the world after observing history up to t and taking a_t";
- a projection head that reads out r_t and produces the predicted next latent $\hat{z}_{t+1}$[1].

The predictor is trained by regressing $\hat{z}_{t+1}$ towards the true next latent z_{t+1} with a mean-squared loss $\mathcal{L}_{\text{pred}}$. To prevent the learned representation from collapsing, some methods such as LeWM additionally introduce a latent-space regularizer $\mathcal{L}_{\text{reg}}$ (instantiated as SIGReg in LeWM). We write the base training objective as $\mathcal{L}_{\text{WM}} = \mathcal{L}_{\text{pred}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}$, where the regularizer term is optional.

CEM planning in latent space. For planning, we use model-predictive planning with the Cross-Entropy Method (CEM) directly in the latent space induced by the encoder ϕ and the predictor g (Rubinstein, 1997; Chua et al., 2018). Given a raw image goal o_g, the goal is embedded once into a latent target $z_g = \phi(o_g)$, and planning is carried out entirely by rolling the world model forward from the current context. Concretely, at each planning step the current history $(z_{t-h+1:t},\, a_{t-h+1:t-1})$ is fixed, and CEM optimizes an action sequence $a_{t:t+H-1}$ of horizon H under a Gaussian proposal $\mathcal{N}(\mu_i, \Sigma_i)$ that is refined over iterations $i = 1, \dots, I$:

$$\{a^{(k)}_{t:t+H-1}\}_{k=1}^{N} \sim \mathcal{N}(\mu_i, \Sigma_i), \qquad \hat{z}^{(k)}_{t+j+1} = \pi\Big(g\Big(\hat{z}^{(k)}_{t+j-h+1:t+j},\, a^{(k)}_{t+j-h+1:t+j}\Big)\Big), \tag{1}$$

where π denotes the projection head reading out r_t (in Section 3, $\pi = h_p$), and $\hat{z}^{(k)}_t = z_t$ initializes the rollout with the encoded context. Each candidate sequence is scored by a cost function $c(\hat{z}^{(k)}_{t+1:t+H};\, z_g)$ that we instantiate either as a terminal latent distance

$$c\Big(\hat{z}^{(k)}_{t+1:t+H};\, z_g\Big) \;=\; \|\hat{z}^{(k)}_{t+H} - z_g\|_2^2 \tag{2}$$

for tasks where the raw latent distance is informative, or as $c(\hat{z}^{(k)}_{t+1:t+H};\, z_g) = -s(\hat{z}^{(k)}_{t+1:t+H};\, z_g)$ for Ball-in-Cup (Section 4.5), with s a learned success head that scores the entire predicted rollout. The elite set $\mathcal{E}_i$ of the top-K lowest-cost samples is then used to update the proposal,

$$\mu_{i+1} = \frac{1}{K}\sum_{k\in\mathcal{E}_i} a^{(k)}_{t:t+H-1}, \qquad \Sigma_{i+1} = \frac{1}{K}\sum_{k\in\mathcal{E}_i} \big(a^{(k)}_{t:t+H-1} - \mu_{i+1}\big)^2, \tag{3}$$

[1]We denote the generic one-step prediction $\hat{z}_{t+1}$ simply by $\hat{z}$ when no ambiguity arises, while retaining full time indices for multi-step rollouts.

and the first action of the resulting mean, $\mu_I[0]$, is executed in the environment; the context is then advanced and planning repeats in receding-horizon fashion.

2.2 Defining World and Action Effects

The introduction described the world effect as the part of a transition that would persist if the current action were replaced by a null action. We now formalize this counterfactual notion in latent space. Fixing the same history $z_{t-h+1:t}$ and past actions $a_{t-h+1:t-1}$, but replacing the current action a_t with the zero action $\mathbf{0}$, we define the action-invariant component of the latent transition as

$$\Delta z_t^{\text{world}} \;:=\; \mathbb{E}\big[\, z_{t+1} \;\big|\; z_{t-h+1:t},\, a_{t-h+1:t-1},\, a_t{=}\mathbf{0} \,\big] \;-\; z_t. \tag{4}$$

Intuitively, $\Delta z_t^{\text{world}}$ is what the world would do on its own if the agent chose not to intervene at step t. It captures the persistent phenomena that our benchmarks are designed to expose, such as gravity-driven sliding, inertia, and drift. Crucially, Eq. 4 depends only on the environment dynamics and the current context, not on the actual a_t, so $\Delta z_t^{\text{world}}$ is well-defined for every transition regardless of whether the two effects combine additively in the true dynamics.

Given $\Delta z_t^{\text{world}}$, we define the action-driven component as the residual, $\Delta z_t^{\text{action}} \;:=\; \Delta z_t - \Delta z_t^{\text{world}}$, so that

$$\Delta z_t \;=\; \Delta z_t^{\text{world}} \;+\; \Delta z_t^{\text{action}} \tag{5}$$

holds by construction. This is a modelling identity rather than a claim that the two effects combine additively at the physical level. In real environments they may couple nonlinearly, for instance when the same push produces different net motion depending on the current gravity-induced velocity. Our formulation does not assume such coupling away; instead, the residual absorbs whatever interaction the two effects have, and the interaction itself is later carried by the shared latent representation that both prediction paths read from (Section 3.1).

2.3 Diagnosing Monolithic Transition Supervision

LeWM (Maes et al., 2026) and DINO-WM (Zhou et al., 2024) report strong visual planning performance, such as on standard PushT, but these standard tasks contain little to no world effect, whereas world effects are pervasive in the real world. To probe prediction and planning under this more realistic setting, we introduce PushT-W, a gravity-perturbed variant of the standard 2D pushing task in which the block keeps sliding under gravity irrespective of the action (its full construction is deferred to Section 4.1), and use LeWM, a recent representative method, for our analysis.

Monolithic supervision degrades under a world effect. On standard PushT, where the block barely moves without the agent's input and the transition is essentially action-driven, LeWM under the standard single-target supervision reaches a CEM planning success rate of 94.0% (Appendix E). On PushT-W, gravity continually drags the block regardless of the action, so a substantial share of each transition is action-invariant; the same model, trained to convergence, achieves only 32%. This sharp drop indicates that motion produced by gravity is folded into the same next-latent target as action-driven motion, and the resulting predictor is not accurate enough to support CEM planning. We next examine whether this shortfall can be attributed to the training data.

Injecting zero-action data does not separate world from action. A natural hypothesis is that the failure above stems from insufficient exposure to purely action-invariant transitions: if the training set rarely contains frames in which the block moves under gravity alone, the model may never see the world effect in isolation. To test this, we augment the PushT-W training set with a 30% mixture of zero-action frames and retrain the same single-target model. Despite explicit exposure to pure gravity-driven transitions, the retrained model attains a CEM planning success rate of 30%, statistically indistinguishable from the 32% baseline. The bottleneck therefore lies not in data coverage but in the training objective itself.

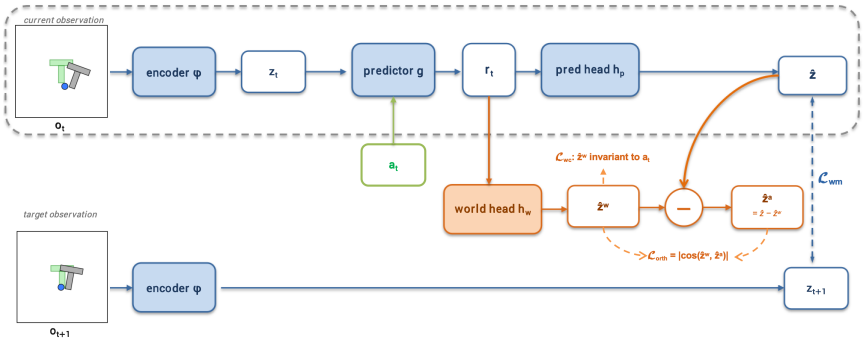

Figure 2: **DWM implementation.** DWM preserves the original prediction pathway of an action-conditioned latent world model and adds an auxiliary world head h_w during training. The world head produces $\hat{z}^{\,w}$, while the complementary action-driven component is defined as $\hat{z}^{\,a} = \hat{z} - \hat{z}^{\,w}$. The auxiliary objectives in Section 3.2 shape the semantics of the two components. The world head is removed at inference, leaving the original prediction pipeline unchanged.

3 DWM: World/Action Disentanglement for Latent World Models

We build on the additive identity $\Delta z_t = \Delta z_t^{\text{world}} + \Delta z_t^{\text{action}}$ (Section 2.2) as an inductive bias, and reallocate the transition supervision of a latent world model so that its two sources are learned along separate prediction paths (Section 3.1), each shaped by its own auxiliary objective (Section 3.2). This construction uses the components that action-conditioned latent world models typically share—an encoder φ, an action-conditioned predictor g that produces a rollout state r_t, and a readout of the predicted next latent—so it can be integrated with such models without altering their architecture or inference.

3.1 Decomposing the Predicted Transition

To instantiate the decomposition in Eq. 5 without changing the original world-model prediction pathway, DWM preserves the prediction of the full next latent and introduces an auxiliary branch to isolate the world-driven component. As illustrated in Figure 2, given the action-conditioned predictor state r_t, the original prediction head h_p continues to produce the full prediction $\hat{z} = h_p(r_t)$ and is supervised by the standard world-model objective $\mathcal{L}_{\text{WM}}$. In parallel, we attach a lightweight world head h_w to the same predictor state and obtain $\hat{z}^{\,w} = h_w(r_t)$. We then define the remaining action-driven component as the residual between the full prediction and the world component:

$$\hat{z}^{\,w} = h_w(r_t), \qquad \hat{z}^{\,a} = \hat{z} - \hat{z}^{\,w}, \qquad \hat{z} = \hat{z}^{\,w} + \hat{z}^{\,a}. \tag{6}$$

This construction converts the additive transition decomposition in Eq. 5 into an explicit decomposition of the model prediction. Rather than learning two independent predictors, DWM retains the original prediction as the complete next-state estimate and uses the auxiliary branch to determine how this prediction is partitioned into world-driven and action-driven components. This residual formulation also guarantees that the two components always reconstruct the original prediction exactly.

Importantly, the decomposition is used only to shape representation learning during training. At inference, the auxiliary world head and the decomposition are discarded, and the original prediction head h_p remains the sole output of the world model. DWM therefore introduces no additional inference-time computation and can be applied without modifying the deployment pipeline of the underlying latent world model.

3.2 Learning Complementary World and Action Components

The parameterization above alone does not guarantee that the two components correspond to world-driven and action-driven changes. We therefore introduce two complementary objectives to identify

the desired decomposition. The original world-model loss $\mathcal{L}_{\mathrm{WM}}$ is retained to supervise the full prediction $\hat{z}$, while the additional losses constrain $\hat{z}^{\,w}$ to be action-invariant and separate it from the action-driven residual:

$$\mathcal{L} = \mathcal{L}_{\mathrm{WM}} + \lambda_{\mathrm{wc}}\mathcal{L}_{\mathrm{wc}} + \lambda_{\mathrm{orth}}\mathcal{L}_{\mathrm{orth}}. \tag{7}$$

World-contrastive objective. To make $\hat{z}^{\,w}$ invariant to the current action while retaining state-specific information, we construct two views of the same context using different actions. For each context i in a mini-batch of size B, we keep the observation history fixed and replace the current action $a_t^{(i)}$ with a perturbed action $\tilde{a}_t^{(i)}$, obtained in practice by randomly permuting actions within the batch. Passing the two action-conditioned predictor states through the shared world head produces $\hat{z}^{\,w,(i)}$ and $\tilde{\hat{z}}^{\,w,(i)}$. After ℓ_2 normalization, denoted by $u^{(i)}$ and $\tilde{u}^{(i)}$, we treat the two representations of the same context as a positive pair and representations from different contexts as negatives:

$$\mathcal{L}_{\mathrm{wc}} = -\frac{1}{2B}\sum_{i=1}^{B}\left[\log\frac{e^{u^{(i)\top}\tilde{u}^{(i)}/\tau}}{\sum_j e^{u^{(i)\top}\tilde{u}^{(j)}/\tau}} + \log\frac{e^{\tilde{u}^{(i)\top}u^{(i)}/\tau}}{\sum_j e^{\tilde{u}^{(i)\top}u^{(j)}/\tau}}\right]. \tag{8}$$

This objective encourages the world component to remain unchanged under different actions, while the contrastive negatives prevent the representation from collapsing across states.

Orthogonality objective. The world-contrastive objective encourages the world component to be invariant to the current action, but action invariance alone does not guarantee a clean decomposition. Since the full prediction is represented as the sum of the world component and the residual, the two components may still encode overlapping information about the same state transition. In particular, the residual may redundantly retain directions already captured by the world head, making the separation ambiguous. We therefore introduce an orthogonality constraint, $L_{orth} = |cos(\hat{z}^w, \hat{z}^a)|$, which discourages shared directions between the two components. Combined with the world-contrastive objective, this encourages $\hat{z}^w$ to capture the action-invariant part of the transition, while pushing the residual $\hat{z}^a$ to explain complementary variation needed for the action-conditioned prediction. The original world-model loss simultaneously ensures that their sum remains predictive of the full next latent. Together, $\mathcal{L}_{\mathrm{wc}}$ identifies the action-invariant part of the transition, while $\mathcal{L}_{\mathrm{orth}}$ encourages the residual to capture complementary action-dependent variation. The original $\mathcal{L}_{\mathrm{WM}}$ simultaneously preserves the predictive quality of their sum.

4 Experiments

We begin by describing the benchmarks and experimental setup (Section 4.1), and then evaluate DWM on the three W-variants, comparing it against representative latent world models and disentanglement methods (Section 4.2). To understand where these gains come from, we analyze the learned world/action decomposition (Section 4.3) and evaluate multi-step prediction and generalization to unseen gravity directions (Section 4.4). We further test DWM under multiple interacting world effects on Ball-in-Cup (Section 4.5), and finally show that DWM generalizes across distinct latent world models (Section 4.6).

4.1 Benchmarks and Experimental Setup

To evaluate world/action disentanglement under controlled changes in persistent autonomous motion, we pair each of three standard tasks—PushT, Reacher, and TwoRoom—with a *W-variant* that introduces a persistent action-invariant source of motion (gravity-driven sliding, vertical-plane gravity, or constant environmental drift, respectively). Each W-variant keeps the same goal as its flat counterpart but adds a source of motion that persists even when the agent is idle; the flat counterpart serves as a comparability control. We further evaluate on Ball-in-Cup as a more demanding simulated task with multiple interacting world effects. Tossing and catching the ball require predicting motion shaped by gravity, string tension, and ball–cup contact, whose combined influence changes throughout the trajectory. Full physical parameterizations, dataset sizes, and OOD splits are deferred to Appendix B (Table 4).

Model and optimization. We evaluate DWM on action-conditioned latent world models built from an encoder φ and a Transformer predictor g (Section 2.1), using a context length $h = 3$ and

observations subsampled at a stride of 5 frames. DWM attaches the world head h_w (Section 3.1) in parallel to the pred head h_p as a two-layer MLP with BatchNorm, leaving the encoder and predictor otherwise unchanged. All trainable modules are optimized jointly with AdamW, and the number of training epochs is fixed per task family before evaluation, so that a base model and its DWM variant are compared under an identical training and planning budget. We instantiate this recipe on two representative models—LeWM and DINO-WM—whose full architecture, optimization, and hardware settings are given in Appendix C.

Evaluation metrics.

- **Prediction quality**: one-step `pred_loss` between $\hat{z}_{t+1}$ and z_{t+1}, reported both offline and under autoregressive rollout. Latent-space MSE for PushT-W and TwoRoom-W; qpos error (rad) for Reacher-W.
- **Disentanglement diagnostics**: response of the world and pred heads under controlled action perturbations, quantifying whether the world head is action-invariant while the pred head remains action-sensitive.
- **OOD gravity**: on PushT-W, cosine similarity of $\hat{z}^{\,a}$ across held-out gravities, and CEM success at gravities distinct from training.
- **Planning**: CEM success rate on a fixed set of start–goal pairs, averaged over 3 planner seeds; per-task protocols (goal offset, evaluation budget, CEM sample and refinement counts, success criterion) are in Appendix B.

4.2 Main Results: Planning Under World Effects

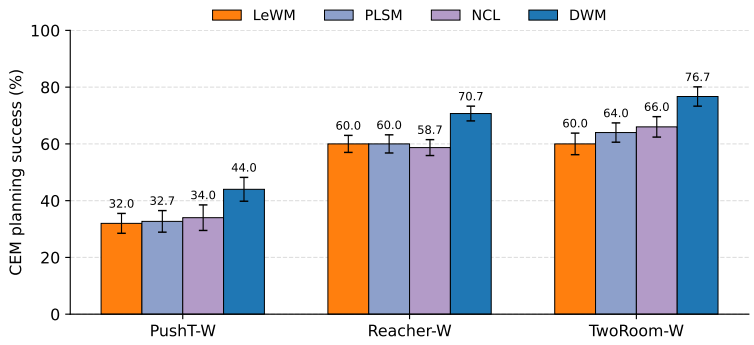

Figure 3: CEM planning success of LeWM, PLSM, NCL, and DWM on the three W-variants. Bars report the mean over 3 CEM planner seeds with n=50 start–goal pairs per seed; error bars denote the across-seed standard deviation.

We compare DWM against LeWM and two representative disentanglement objectives adapted on top of it: PLSM (Saanum et al., 2024), which minimizes the mutual information between the current latent state and the action-induced latent change, and NCL (Wang et al., 2024), which imposes non-negativity constraints on contrastive representations. Both baselines share LeWM's single next-latent target and neither reallocates supervision between world- and action-driven components. We evaluate all methods under one training recipe (Section 4.1) and one CEM protocol (3 planner seeds, $n = 50$ start–goal pairs per seed; per-task configurations in Appendix B). Figure 3 reports CEM planning success on the three W-variant tasks. LeWM drops from 94.0%, 82.0%, and 94.0% on the flat counterparts (Appendix E) to 32.0%, 60.0%, and 60.0% once a persistent world effect is present, indicating that single-target supervision struggles in this regime; PLSM and NCL remain close to it at 32.7%, 60.0%, 64.0% and 34.0%, 58.7%, 66.0% on PushT-W, Reacher-W, and TwoRoom-W.

DWM supervises an action-invariant world component and its complementary action-driven component, raising CEM planning success on the three tasks from 32.0%, 60.0%, 60.0% to 44.0%, 70.7%, and 76.7%, an average absolute improvement of 13.1 percentage points over the baseline; Appendix D shows that both $\mathcal{L}_{\text{wc}}$ and $\mathcal{L}_{\text{orth}}$ are needed for this gain, and Appendix E confirms that the separation does not degrade performance on the flat PushT, Reacher, and TwoRoom tasks.

4.3 Does DWM Learn the Intended Decomposition?

Building on the planning gains of Section 4.2, we next examine whether the intended world/action decomposition is realized in the learned representation. Specifically, for each validation window we fix the observation history $z_{t-h+1:t}$ and sample K alternative actions $\{\tilde{a}_t^{(k)}\}_{k=1}^{K}$; we then measure the response of each head by the variance and peak-to-peak range of its outputs across these perturbations, which capture the average and worst-case sensitivity, respectively. A successful decomposition should preserve the pred head's sensitivity to the action while rendering the world head invariant to it.

Table 1: Disentanglement diagnostics on the three W-variant tasks. For each validation window the current action a_t is perturbed with K random alternatives while the history is held fixed; we report the per-sample variance and peak-to-peak range of each head across these perturbations, averaged over windows. Since LeWM has no world head, only the pred-head variance column is populated for it.

Task	**Pred-head var** (LeWM)	**Pred-head var** (DWM)	**World-head var** (DWM)	**World/pred var ratio** ↓	**World/pred range ratio** ↓
PushT-W	0.847	0.864	0.0017	**0.0019**	**0.0424**
TwoRoom-W	0.1304	0.0847	$3.76e{-4}$	**0.0045**	**0.0783**
Reacher-W	0.0053	0.0072	$8e{-5}$	**0.0107**	**0.0909**

As summarized in Table 1, the DWM pred head matches the action sensitivity of the LeWM predictor in magnitude across all three W-variants, while the world-head variance is significantly reduced. The world/pred variance and range ratios remain well below 10^{-2} and 10^{-1} across tasks, showing that the world head is close to invariant to the action both on average and in the worst case, while the pred head keeps the action-conditioned transition available for planning.

4.4 Multi-Step Prediction and OOD Dynamics Generalization

Section 4.3 verified that the decomposition emerges at the representation level. We next ask whether it also manifests in the learned latent dynamics, in two complementary ways: by remaining stable when the source of the world effect is shifted out of distribution, and by improving multi-step prediction along the horizon that CEM actually queries.

OOD gravity generalization. To test whether the learned decomposition further generalizes across world dynamics, we train on PushT-W at gravity $(45, 25)$ and evaluate at held-out settings, comparing $\hat{z}^{\,a}$ across gravities and CEM planning success under the protocol of Section 4.2. Pooled over $(90, 0)$ and $(0, 45)$, the OOD-to-in-distribution $\hat{z}^{\,a}$ cosine still reaches 0.9991 (mean effect MSE 0.0015), indicating that DWM isolates a direction of action effects independent of the gravity value. Control follows suit (Fig. 4): LeWM drops to 30.0% pooled over the two OOD gravities, while DWM retains 44.0% ($+14$ pp), matching the in-domain gap and thereby showing that the planning gains of Section 4.2 come from a gravity-agnostic decomposition rather than overfitting to the training gravity.

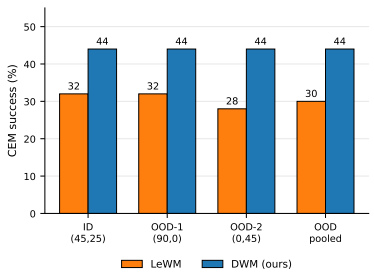

Figure 4: OOD gravity CEM planning success on PushT-W.

Latent prediction and rollout quality. We evaluate the predictor on held-out validation windows of each W-variant with one-step prediction and horizon-20 autoregressive rollout. Error metrics follow Section 4.1: latent MSE for PushT-W and TwoRoom-W, and qpos error (rad) for Reacher-W.

Table 2 shows that DWM reduces the one-step prediction error on the three W-variants by 89.8%, 31.6%, and 29.8%, and the horizon-20 error by 66.7%, 31.7%, and 26.1%; on PushT-W its horizon-20 error (0.344) is even lower than LeWM's one-step error (0.372). The planning gains of Section 4.2 thus rest on more accurate latent dynamics along the horizons that CEM optimizes.

Table 2: Latent prediction quality on the three W-variant tasks. *Offline* is the one-step error on the held-out validation set; *Rollout@20* is the autoregressive error at horizon 20. Error metric: latent-space MSE for PushT-W and TwoRoom-W, qpos error (rad) for Reacher-W.

Task	Metric	LeWM	DWM	Relative change
PushT-W	Offline `pred_loss` ↓	0.372	**0.038**	−89.8%
PushT-W	Rollout@20 `pred_loss` ↓	1.034	**0.344**	−66.7%
TwoRoom-W	Offline `pred_loss` ↓	0.317	**0.217**	−31.6%
TwoRoom-W	Rollout@20 `pred_loss` ↓	0.382	**0.261**	−31.7%
Reacher-W	Offline qpos error (rad) ↓	0.046	**0.033**	−29.8%
Reacher-W	Rollout@20 qpos error (rad) ↓	0.299	**0.221**	−26.1%

4.5 Multiple Interacting World Effects: Ball-in-Cup

We further evaluate DWM on Ball-in-Cup, where gravity, string tension, and ball–cup contact jointly drive the ball's motion, so a single trajectory couples several action-invariant physical processes and probes DWM under multiple interacting world effects. Under matched training and planning budgets, we compare LeWM and DWM using CEM planning success and autoregressive prediction loss, with a per-model success head scoring candidate trajectories (Appendix B).

Table 3: CEM planning success and autoregressive prediction loss on Ball-in-Cup.

Model	CEM success	Rollout `pred_loss` ↓
LeWM	36.8%	0.671
DWM	**42.8%**	**0.089**

Table 3 shows that DWM raises CEM planning success from 36.8% to 42.8% and cuts autoregressive prediction loss from 0.671 to 0.089, confirming that the world/action decomposition helps under multiple interacting world effects.

4.6 Generalization Across Distinct Latent World Models

To test whether DWM's benefits generalize beyond LeWM, we integrate the same world/action decomposition into DINO-WM (Zhou et al., 2024), another representative action-conditioned latent world model that operates on frozen DINOv2 patch features rather than a scratch-trained encoder. Under matched training and CEM budgets, we compare planning performance with and without DWM on the three W-variants (Fig. 5). Applied to DINO-WM, DWM raises planning success on all three tasks by 10.2 percentage points on average. Together with the LeWM results, world/action separation plugs cleanly into existing action-conditioned latent world models and delivers consistent improvements across them.

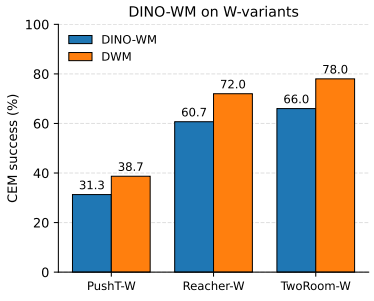

Figure 5: Results on DINO-WM.

5 Conclusion

Joint-embedding latent world models are trained against a single next-latent target, fusing action-driven and action-invariant changes into one supervision signal and leaving the model unable to attribute transitions to their causes—a supervision-level limitation that undermines prediction and planning under realistic world effects. To address this, we introduce DWM, a world/action-decomposed training framework that supervises the two components separately through a world head and an orthogonality constraint, delivering consistent improvements across multiple more realistic benchmarks and multiple representative action-conditioned latent world models.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript formatting and language editing.

ETHICS STATEMENT

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

REPRODUCIBILITY STATEMENT

Section 3 describes the model decomposition and training objectives. Section 4.1 specifies the architecture and optimization settings, and Section 4.1 defines the evaluation metrics. Section 4.6 describes the integration of the same DWM implementation with DINO-WM and the comparison protocol. Appendix B provides the task definitions, physical parameters, dataset sizes, and planning configurations used in the reported experiments; Appendix C gives the per-model architectures, optimization schedules, and hardware setup; Appendix D reports the ablations on $\mathcal{L}_{\mathrm{wc}}$ and $\mathcal{L}_{\mathrm{orth}}$.

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. *arXiv preprint arXiv:2301.08243*, 2023.

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

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

Tianqi Du, Qi Zhang, Yifei Wang, and Yisen Wang. Beyond the next step: Variable-length latent world models for long-horizon planning. *arXiv preprint arXiv:2606.21775*, 2026.

Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. Focus: object-centric world models for robotic manipulation. *Frontiers in Neurorobotics*, 19:1585386, 2025.

Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi Jaakkola. Learning task informed abstractions. In *ICML*, 2021.

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

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

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

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *ICLR*, 2021.

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

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *ICLR*, 2024.

Xiaodi Huang, Ziyi Ding, Jingtian Wan, Yuchen Liu, Yuan Zhang, Xiao-Ping Zhang, Jiayu Chen, Zhang Zhang, and Tao Huang. Traj-lewm: Path-aware world-model planning via latent trajectory cost. *arXiv preprint arXiv:2608.14125*, 2026.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *ICLR*, 2020.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020.

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

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *ICLR*, 2023.

Minting Pan, Xiangming Zhu, Yunbo Wang, and Xiaokang Yang. Iso-dream: Isolating and leveraging noncontrollable visual dynamics in world models. In *NeurIPS*, 2022.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. In *ICLR*, 2023.

Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

Tankred Saanum, Peter Dayan, and Eric Schulz. Simplifying latent dynamics with softly state-invariant world models. In *NeurIPS*, 2024.

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.

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.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Tongzhou Wang, Simon S. Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised mdps: Learning world models better than the world itself. In *ICML*, 2022.

Yifei Wang, Qi Zhang, Yaoyu Guo, and Yisen Wang. Non-negative contrastive learning. In *ICLR*, 2024.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. AdaJEPA: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 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. *arXiv preprint arXiv:2411.04983*, 2024.

A Related Work

Latent world models for planning. Latent world models learn compact predictive representations for imagination, planning, and policy learning (Hafner et al., 2021; 2023; Hansen et al., 2022; 2024; Maes et al., 2026; Micheli et al., 2023; Robine et al., 2023). Among them, joint-embedding world models have attracted growing attention for directly regressing future visual representations without reconstructing raw signals, yielding compact and semantically meaningful latents for downstream control. Representative works include LeWM (Maes et al., 2026) and DINO-WM (Zhou et al., 2024), whose transition prediction loss matches the combined next-state representation without explicitly allocating supervision between action-driven and action-invariant changes. Our method is complementary to prior advances in planners (Rubinstein, 1997; Chua et al., 2018; Hansen et al., 2024), backbones (Bardes et al., 2024; Caron et al., 2021), and policy objectives (Hafner et al., 2020; 2023). Instead, it modifies the supervision of a shared latent predictor by introducing two projections with complementary objectives, explicitly separating action-driven and action-invariant dynamics during training.

Decomposing latent transitions in world models. Most prior work disentangles *what* a latent state represents rather than *what causes its transition*: object-centric models factorize states into entities or spatial components (Kipf et al., 2020; Ferraro et al., 2025; Locatello et al., 2020), while task-oriented methods split task-relevant from irrelevant factors (Wang et al., 2022; Fu et al., 2021). Closer to our goal, Iso-Dream (Pan et al., 2022) learns separate forward branches for controllable and noncontrollable dynamics, but this architectural split modifies the base world model and imposes a hard partition that is ill-suited when world and action effects act on the same object. NCL (Wang et al., 2024) enforces non-negativity for feature-level disentanglement, and PLSM (Saanum et al., 2024) regularizes state-dependence of latent changes; neither, however, explicitly separates world from action effects, limiting their benefit when world effects are present. In contrast, DWM introduces an auxiliary world head with complementary training objectives to explicitly separate the two effects, improving action modeling and downstream planning while integrating easily with existing latent world models and preserving the original inference pipeline.

B Task and Dataset Details

This appendix provides the full physical parameterization, dataset scale, and out-of-distribution (OOD) splits for the tasks introduced in Section 4.1. Table 4 summarizes the resulting benchmark suite; the paragraphs below describe each pair in turn, together with Ball-in-Cup as a more demanding simulated task with multiple interacting world effects.

Table 4: Control tasks used in this paper, grouped into three flat/W-variant pairs ("-W" for *world effect*, and *flat* for the counterpart without the added persistent world effect) plus Ball-in-Cup as a more demanding simulated task with multiple interacting world effects. Each W-variant introduces a persistent action-invariant world effect; the flat counterpart shares the same goal but removes that effect. Tasks in the same block share a goal (block pushing, qpos matching, 2D target reaching, catching the ball in the cup).

Task	**Action-invariant world effect**	**Physical setting**
PushT	None	Horizontal table
PushT-W	Gravity-driven sliding	Gravity $(45, 25)$, damping 0.92
Reacher	None	Standard DMC Reacher
Reacher-W	Vertical-plane gravity	$g{=}-9.81 \sin 60^\circ$ along z
TwoRoom	None	$\Delta p{=}a \cdot s$
TwoRoom-W	Constant drift bias	$\Delta p{=}a \cdot s + b$, $b{=}(4, 2)$
Ball-in-Cup	Multiple interacting effects	Standard DMC Ball-in-Cup

PushT / PushT-W. Standard PushT (Chi et al., 2023) is a horizontal-table pushing task: the block does not drift under zero action, so action-invariant world effects are weak. Its W-variant counterpart keeps the same pushing objective but includes gravity $(45, 25)$ in the `pymunk` space (with damping 0.92), so the block continues to slide even when the agent is idle. We use standard PushT as the flat comparability control, and train PushT-W on `tilted_pusht_train_4x` with 3200 episodes $\times$ 200 steps under a random policy; for OOD stress tests, we additionally evaluate at gravity $(90, 0)$ and $(0, 45)$ (Section 4.4). CEM planning uses a goal offset of 10 frames, an evaluation budget of 50 environment steps, and a CEM configuration of $300 \times 30 \times \text{top30}$ (300 samples, 30 refinement iterations, top-30 elite set); an attempt is counted as a success if the block matches the goal configuration at the final step of the budget.

Reacher / Reacher-W. Standard DMC Reacher (Tassa et al., 2018; Tunyasuvunakool et al., 2020) operates on a horizontal plane, where action-invariant world effects are negligible. Reacher-W is a customized DMC Reacher in which the shoulder and wrist hinge axes are rotated from $[0, 0, 1]$ to $[0, 1, 0]$ so that the arm moves in the vertical x–z plane while keeping the same qpos-matching objective. Under the standard gravity vector $g = (0, 0, -9.81 \sin 60^\circ) \approx (0, 0, -8.50)$ m/s^2, gravity therefore exerts a non-zero torque on the shoulder joint and drives passive motion of the arm even under a null action, whereas in the original horizontal geometry the same gravity vector is parallel to the shoulder axis and produces no such torque. Here the world effect refers to this gravity-driven passive motion under a null action, not an additional external force channel. Both variants use $10{,}000$ episodes $\times$ 200 steps, with a held-out validation set of 500 episodes $\times$ 200 steps generated under the same physics and policy mix. CEM planning uses a goal offset of 25 frames, an evaluation budget of 200 environment steps, and a CEM configuration of $300 \times 10 \times \text{top30}$; an attempt succeeds if the joint configuration reaches within 0.15 rad of the goal qpos at any step within the budget.

TwoRoom / TwoRoom-W. Standard TwoRoom is a 2D navigation environment with transition $p_{t+1} = p_t + a_t \cdot s$, so the agent does not drift under zero action and action-invariant world effects are absent. TwoRoom-W keeps the same reach-target objective but adds a persistent environmental drift, $p_{t+1} = p_t + a_t \cdot s + b$ with $b = (4, 2)$ pixels/step. We train the W-variant on 2000 episodes $\times$ 200 steps with a random policy under the same drift, yielding $324{,}498$ training windows and a matched validation set of 60×100 transitions. CEM planning on TwoRoom-W adopts a longer-horizon CEM-MPC protocol to accommodate the persistent drift: goal offset 90 frames, evaluation budget of 25 MPC decisions, each planning over a rollout horizon of 5 steps and executing an action block of 5 environment steps before re-planning, with the same $300 \times 10 \times \text{top30}$ CEM budget as Reacher-W; success is declared when the agent reaches the target region within the total budget.

Ball-in-Cup. The DMC Ball-in-Cup task (Tassa et al., 2018; Tunyasuvunakool et al., 2020) requires a movable cup to toss and catch a ball attached to it by a string. The training set contains 800 episodes of 200 steps each. Both world models are trained for 20 epochs with the configuration in Appendix C.

After world-model training, we fit a separate success head to each model's autoregressively predicted latents, supervised by success labels from the corresponding environment trajectories. Each model uses its own success head to rank candidate action sequences during CEM planning. For DWM, this head receives the complete predictions from the pred head; the auxiliary world head is used only during world-model training. Both models use the same CEM budget. Planning success measures whether the ball is caught within the evaluation budget, and autoregressive prediction loss is evaluated over the planning horizon. Planning success is averaged over 100 start states per seed and reported as the mean over seeds $\{42, 43\}$. CEM uses a rollout horizon of 5 frames with a receding horizon of 5 and an action block of 5 environment steps, an evaluation budget of 50 environment steps, and a per-decision budget of $150 \times 10 \times \text{top15}$ samples with residual variance 0.08. Each success head is a two-layer MLP trained for 30 epochs on $8{,}000$ rollout sequences generated by the corresponding world model at the same horizon, with success labels taken from the underlying Ball-in-Cup trajectories; the two models are compared under matched training and planning protocols.

C Implementation Details

Both base models share the DWM additions of Section 3: a world head h_w attached in parallel to the pred head h_p, an action-shuffle world-contrastive loss $\mathcal{L}_{\text{wc}}$, and an orthogonality penalty

$\mathcal{L}_{\text{orth}}$ between the world and action-residual directions, with contrastive temperature $\tau = 0.07$ and loss weights $\lambda_{\text{wc}}, \lambda_{\text{orth}}$ from Eq. 7. At inference the world head is discarded and each base model's original prediction and planning pipeline is used unchanged. Below we give the per-model configuration.

LeWM. The encoder φ is a ViT-tiny (hidden dimension 192, 12 layers, patch size 14, input resolution 224^2) trained from scratch, and the action-conditioned predictor g is a 6-layer Transformer (16 attention heads, head dimension 64, MLP dimension 2048) with AdaLN action conditioning; the context length is $h=3$ and observations are subsampled at a stride of 5 frames. The pred head h_p and world head h_w are each a two-layer MLP $192 \to 2048 \to 192$ with BatchNorm. The encoder, predictor, and both heads are initialized randomly and optimized jointly with AdamW ($\text{lr} = 5 \times 10^{-5}$ across all modules, weight decay 10^{-3}) under a linear-warmup cosine-annealing schedule at the epoch level, in `bfloat16` on NVIDIA H20 GPUs with batch size 128. The number of epochs is fixed per task family before evaluation: 5 for PushT / PushT-W, 20 for Reacher / Reacher-W, 100 for TwoRoom / TwoRoom-W, and 20 for Ball-in-Cup.

DINO-WM. The encoder φ is a frozen pretrained DINOv2 ViT-S/14 (`dinov2_vits14`, patch size 14), producing 384-dimensional patch features, and the predictor g is DINO-WM's 6-layer causal ViT predictor over the P patch tokens per frame, with the same context length and frame stride as above. The world head h_w is a token-wise two-layer MLP $384 \to 2048 \to 384$ with BatchNorm, applied independently to each patch token, while the pred head h_p is the identity on r_t, so the predicted next latent $\hat{z} := r_t$ matches the baseline and only the world head is added. The main loss $\mathcal{L}_{\text{WM}}$ is DINO-WM's original prediction and reconstruction objective, and both auxiliary losses use the same formulation as for LeWM, applied to the visual portion of the token embedding and averaged over (time, patch) locations. The predictor, action/proprio encoders, decoder, and world head are optimized jointly under the base model's schedule with batch size 256 for PushT-W and Reacher-W and 128 for TwoRoom-W on 8 NVIDIA H20 GPUs, with the epoch budget fixed per task family before evaluation. At inference the world head is discarded and the raw predictor output is fed to the CEM planner as in the baseline.

D Ablation Studies

We conduct ablation studies on PushT-W to evaluate the contributions of $\mathcal{L}_{\text{wc}}$ and $\mathcal{L}_{\text{orth}}$ to DWM. We vary each loss weight while holding the other fixed and measure CEM planning success.

Table 5: Loss-weight ablation on PushT-W (CEM planning success, n=50). The weight pair $(0, 0)$ corresponds to the LeWM baseline; other configurations vary one weight while holding the other fixed.

λ_{wc}	0.0	0.1	0.2	0.3	0.3	0.3
λ_{orth}	0.0	0.5	0.5	0.0	0.1	0.5
CEM success	32.0%	36.0%	38.0%	40.0%	42.0%	**44.0%**

Along the world-contrastive axis, reducing λ_{wc} from 0.3 to 0.2 to 0.1 while keeping $\lambda_{\text{orth}} = 0.5$ yields CEM success of $44.0\% \to 38.0\% \to 36.0\%$: as the contrastive signal weakens, the world head deviates from action-invariance and forfeits most of the gain over the LeWM baseline. Along the orthogonality axis, reducing λ_{orth} from 0.5 to 0.1 to 0.0 while keeping $\lambda_{\text{wc}} = 0.3$ gives $44.0\% \to 42.0\% \to 40.0\%$, showing that an action-invariant world head alone does not couple its component into $\hat{z}$ tightly enough to close the remaining gap. DWM attains its best CEM success (44.0%) only when $\mathcal{L}_{\text{wc}}$ and $\mathcal{L}_{\text{orth}}$ are jointly active, confirming that both losses are integral to the framework.

E Planning Success on Flat Counterparts

This appendix reports the behavior of DWM on the flat counterparts of the three W-variant tasks (Section 4.2), where the environment does not inject a persistent action-invariant world effect. All models share the training recipe of Section 4.1 and the CEM protocol of Section 2.1; means and standard deviations are computed over 3 CEM planner seeds with n=50 start–goal pairs per seed.

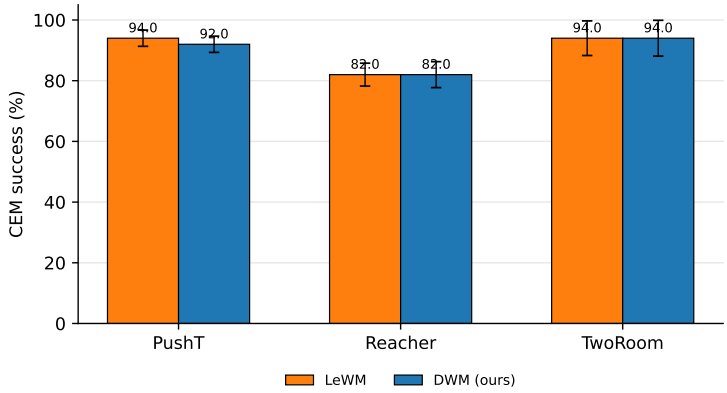

Figure 6: CEM planning success rate on the flat counterparts of PushT, Reacher, and TwoRoom (no persistent action-invariant world effect). DWM stays within noise of the LeWM baseline on all three tasks, indicating that the auxiliary supervision does not harm planning when the world component it targets is absent. Same training recipe and CEM configuration as Section 4.2, averaged over 3 CEM planner seeds (n=50 start–goal pairs per seed); error bars denote across-seed standard deviation.