

LeWAM: A JEPA World Action Model with Diffusion-Steering-Based MPC

Shashank Hegde

Alexander Popov

Elie Aljalbout

Nikolai Smolyanskiy

NVIDIA

Abstract: World action models (WAMs) predict actions and future observations, typically from a reconstruction-based representation that carries noisy, redundant information which can complicate downstream predictions. We introduce LeWAM, a bidirectional transformer for forward, backward, inverse dynamics and policy prediction, on a decoder-free JEPA latent trained end-to-end through all four modes. We see the following benefits: 1) Alignment: linear probes read robot and object state from LeWAM’s latent better than from a regular Le World Model (a forward-only JEPA world model), while the latent ignores visual distractors as well as LeWM does and far better than a reconstruction-based WAM. 2) Acting: Closed-loop evaluations of LeWAM match a regular flow-matching policy trained on the same encoder at matched size, while also providing a world model. 3) Planning: Sampling raw actions when planning with WAMs lets MPC exploit dynamics-model inaccuracies; planning in the noise space of the policy head instead improves the closed-loop performance of these WAMs.

Keywords: World-Action-Models, MPC, JEPA

1 Introduction

Robotic control increasingly relies on predictive models to anticipate environmental dynamics. Traditional world models learn a latent space through image reconstruction, subsequently training policies conditioned on these representations [1, 2]. A recent evolution of this paradigm, termed world action models (WAM), explicitly couples future-state prediction with action generation to ground control directly in learned dynamics [3, 4, 5]. While these generative approaches successfully integrate vision and action, they predominantly rely on pixel-level reconstruction to shape their underlying representations [3]. Consequently, their latent spaces are forced to encode every visual detail a decoder might render, regardless of its relevance to the task [3]. In visually complex environments, this architectural choice inherently wastes model capacity on noisy, task-irrelevant information, complicating downstream predictive tasks [6, 7].

To circumvent the limitations of reconstruction, alternative frameworks such as the joint-embedding predictive architecture model dynamics exclusively in latent space [8, 9, 10]. By discarding the pixel decoder entirely, these architectures inherently resist task-irrelevant appearance details [11, 12]. Recent instantiations demonstrate stable, end-to-end training from pixels using regularization techniques that prevent representational collapse without relying on heuristics [13, 14]. Despite these representational advantages, existing frameworks in this category typically separate the dynamics model from the policy, lacking the unified multi-modal structure found in modern WAMs [13].

Bridging this gap, we propose LeWAM, a unified, reconstruction-free world action model. Building upon foundational decoder-free predictors, LeWAM introduces a bidirectional transformer capable of simultaneous forward dynamics, backward dynamics, inverse dynamics, and policy prediction [13]. The predictor is conditioned on actions, while the encoder is trained end-to-end directly from task pixels under a strict regularizer. Because the architecture eschews pixel decoders, tokenizers,

Table 1: World models and world action models as published: what shapes the latent and what one network does. LeWM has no policy head. Dreamer [2]; TD-MPC2 [24]; DINO-WM and V-JEPA 2-AC [30, 31]; LeWM [13]; UWM [3]; Cosmos Policy [4].

	decoder- free	action loss to encoder	policy head	forward dyn.	inverse dyn.	backward dyn.	multimodal head	one backbone
Dreamer	–	–	✓	✓	–	–	–	–
TD-MPC2	✓	–	✓	✓	–	–	–	–
DINO-WM / V-JEPA 2-AC	✓	–	–	✓	–	–	–	–
LeWM	✓	–	–	✓	–	–	–	–
UWM	–	–	✓	✓	✓	–	✓	✓
Cosmos Policy	–	–	✓	✓	✓	–	✓	✓
LeWAM (ours)	✓	✓	✓	✓	✓	✓	✓	✓

and pixel-space losses, its representations are shaped entirely by the dual objectives of accurate dynamics prediction and robust action generation [15, 16].

Integrating a generative policy within a world model naturally invites test-time intervention through planning [17, 18]. However, executing sampling-based model predictive control over raw actions often fails in practice [19]. When optimizing over action space, the planner tends to exploit inevitable inaccuracies within the learned dynamics model, leading to plans that optimize for hallucinated futures and subsequently fail during deployment [20]. To resolve this, we shift the decision variable of the sampling-based planner from the action space to the deterministic noise space of the flow-matching policy [21, 22, 23]. This diffusion steering approach searches strictly within the world model’s imagination, evaluating candidates decoded at imagined states [24]. By projecting proposals onto the sampler’s typical set, the search remains constrained to actions the policy head was explicitly trained to produce.

We evaluate our approach on human-generated demonstration datasets augmented with visual distractors [25, 26]. Our investigation yields the following primary contributions:

- **Alignment:** We demonstrate representational alignment; the latent space captures dynamic robot and object states as effectively as reconstruction-based baselines, while successfully ignoring visual distractors.
- **Acting:** We validate robust acting capabilities; the integrated flow-matching policy matches the performance of standard behavioral cloning policies while simultaneously providing a world model for dynamics prediction.
- **Planning:** We establish a stable planning paradigm; steering the generative process by sampling within the policy’s noise space prevents the exploitation of dynamics errors and significantly improves closed-loop performance during inference [27, 28, 29].

2 Related work

World models and planning in latent space. World Models such as Dreamer [32, 1, 2] learn a recurrent latent space primarily via reconstruction and then train policies conditioned on latents in this space. The main shortcoming of reconstruction-based world modeling is in the capacity wasted on noisy and task-irrelevant information. Many approaches have been proposed to address this problem. For instance, DreamerPro and bisimulation methods [11, 12] drop the decoder to resist task-irrelevant appearance [6, 7]. TD-MPC2 [33, 24] plans by MPPI over a decoder-free latent and value. JEPA world models [30, 10, 31] plan by CEM-MPC on a goal-latent distance, and LeWM [13] trains its encoder end-to-end under SIGReg [14]. LeWAM adds the action head and multi-mode skeleton to LeWM’s latent. We present a clear comparison to these methods in Table 1. Note that the head on LeWM’s frozen latent we present in our experiments is our control, not an intrinsic part of LeWM.

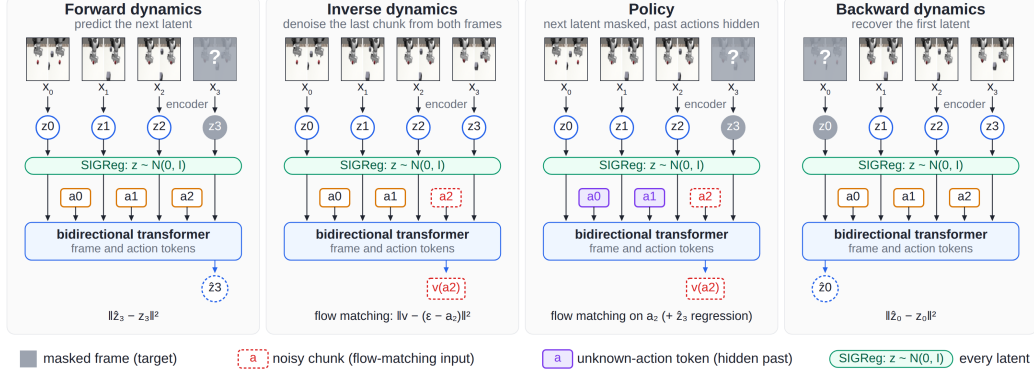


Figure 1: The four training modes on one window: the mode is set by what is masked at the transformer’s input. Nothing is decoded to pixels.

World action models. One recent trend in robotics is to couple future-state prediction with action generation, grounding robot control in learned dynamics in an architecture referred to as world action models. Dreamitate adapts a video generator to task-specific human demonstrations and tracks tools in its predictions to obtain executable trajectories [34]. Unified World Models (UWM) makes this connection explicit through independently controlled video and action diffusion, allowing one model to serve as a policy, forward model, or inverse model [3]. Other methods such as DreamZero exploit pretrained video priors to generalize to unseen tasks and motions [5]. How such priors guide action varies, for instance, Cosmos Policy incorporates actions, future states, and values into the video model’s native diffusion process, enabling both control and planning [4].

Steering a generative policy. Frozen diffusion and flow-matching policies [25, 15, 16, 35] offer several points of intervention for test-time control. Guidance steers the sampling process itself [17], whereas candidate ranking selects among policy samples using a learned value [36, 4] or world-model rollouts [37, 38]. Rather than merely selecting a proposal, MPC can refine it through MPPI in *action* space, potentially leaving the policy’s support [20, 39], while Diffusion-ES alternates simulator-based scoring with re-noising and denoising [18]. A different route is to optimize the noise that generates the action. DSRL learns a policy over this noise through environment interaction, without a world model or planning [21, 22]. Golden Ticket instead searches offline with CEM, using real task-reward rollouts to find one task-specific noise vector that is reused at every step [23]. DS builds on this formulation rather than introducing a new optimizer: both its decision variable and DS-CEM’s update match Golden Ticket’s. The distinction lies in when and where search happens, and how candidates are evaluated. At every decision, DS searches within world-model imagination, decoding action chunks at imagined states and scoring them by goal-latent distance, without search-time environment interaction or a learned value function. This formulation accommodates different sampling-based updates such as CEM [27, 19], MPPI [28], and CMA-ES [29].

3 LeWAM

We use a multi-mode skeleton following UWM and Cosmos Policy. This model operates like the predictor in LeWM (Figure 1).

Inputs and notation. A training window contains four frames, $o_{t-2:t+1}$, at a frameskip of five, and the three 5-step action chunks between them, $a_{t-2:t} \in \mathbb{R}^A$. The chunk dimension A is five steps times the per-step action dimension ($A = 10$ on PushT, 35 on robomimic, 70 on Transport), z-scored per dimension. The first three frames form the context, with latents $\mathbf{z}_t = (z_{t-2}, z_{t-1}, z_t)$; the fourth frame is the prediction target, and the final chunk a_t is the chunk the policy must produce.

Modes. LeWAM masks specific inputs to serve four training modes: (i) *Forward*: the final frame’s latent is replaced by a mask token, all actions are clean, and the backbone regresses z_{t+1} via MSE against the live encoder’s output; (ii) *Backward*: masks the first frame’s latent, regressing it from the later frames and all actions; (iii) *Inverse*: keeps all frames clean and replaces the target chunk a_t with a noisy flow-matching sample whose velocity the backbone predicts, inferring the action that bridges past and future states; and (iv) *Policy*: masks the final frame and replaces a_t with a noisy flow-matching sample, so both losses apply and the model must deduce the action from the frames while predicting the future latent. In this mode the past action tokens ($a_{t-2:t-1}$) are replaced by a learned unknown-action token, so the policy cannot copy the demonstrator’s recent motion.

During training, every mode is executed on every batch: the encoder processes the inputs once, the backbone runs independently for each mode, and the losses are summed. The Forward and Backward modes, alongside SIGReg [14], impose a JEPA structure on the latent space, while the Inverse and Policy modes make the space more task-aligned.

Action Generation. Action generation employs rectified flow [16]. During training, we sample $\sigma \sim U(0, 1)$ and construct a noisy action $a_\sigma = (1 - \sigma)a + \sigma\epsilon$, training the backbone to predict the velocity target $\epsilon - a$. At test time, the action chunk is integrated from standard noise $\epsilon \sim \mathcal{N}(0, I)$ using eight Euler steps. This integration forms a deterministic mapping, denoted $a = g(\epsilon; \mathbf{z}_t)$.

Notably, unlike reconstruction-based WAMs, our architecture operates without a pixel decoder, tokenizer, or any pixel- or token-space loss. As a result, the latent representations are shaped entirely by the dual objectives of dynamics prediction and action generation.

Encoder and latent. Each frame is independently encoded at a resolution of 224 pixels using a ViT-tiny/14. We use the DINOv2 small architecture [40], but train it entirely from scratch without pretrained weights. The encoder’s [CLS] token passes through a BatchNorm MLP projector to produce a single latent vector, $z_t \in \mathbb{R}^{384}$. This vector serves as the model’s sole representation: it is read by the probes, compared by the planner, and regressed by the dynamics model. Similar to LeWM, this latent is taken directly from the live encoder output, without relying on an exponential moving average (EMA) target network or a stop-gradient operation. To prevent representation collapse, we apply SIGReg [14], which regularizes the model by pushing the batch of latents toward an isotropic Gaussian distribution.

Backbone A bidirectional transformer of depth 8, width 512 and 16 heads reads the three frame latents and three action tokens with modality, frame and σ embeddings, and serves four modes by masking. Figure 2 shows the LeWAM architecture of Section 3: a shared ViT-tiny encoder with SIGReg on its single latent per frame, and one bidirectional transformer that serves the four training modes of Figure 1 by masking.

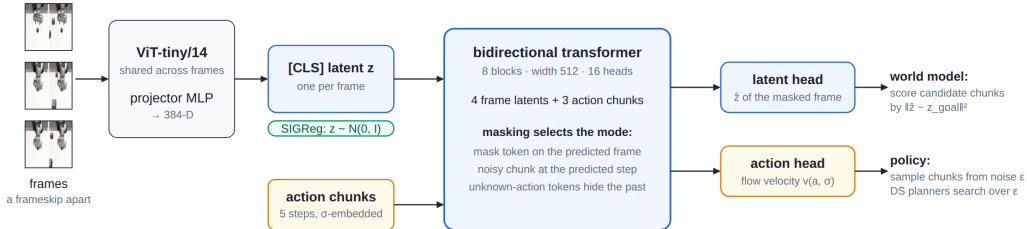


Figure 2: LeWAM: a shared ViT-tiny encoder with SIGReg on its single latent per frame, and one bidirectional transformer that serves the four training modes by masking. No pixel decoding.

4 Diffusion steering: planning in the policy’s noise space

4.1 Goal-conditioned planning with a latent distance

Given context $\mathbf{z}_t$ and a goal latent z_g retrieved from a demonstration, following [30, 13], we score a chunk a by

$$\hat{c}(a) = \|\hat{f}(\mathbf{z}_t, a) - z_g\|_2, \quad (1)$$

with $\hat{f}$ the forward mode, and minimize it over H chunks by a sampling-based planner. The latter proposes S plans, scores each on its rolled-out final latent, updates the distribution, repeats this process I times, and executes the chosen plan’s first chunk before replanning. The update rule defines the planner. Sampling-based MPCs originally search over raw actions $a_{1:H} \in \mathbb{R}^{AH}$.

4.2 Motivation: Failure of Action-Space MPC with LeWM

While action-space sampling-based MPC has been shown to work with LeWM, it fails on the smaller human-sourced datasets studied here (see Figure 5). We hypothesize that the dynamics model is imperfect with less data, and compounds errors during planning rollouts. Gaussian action-sampling MPC on such a model optimizes for hallucinated futures, and the planned actions fail during deployment.

4.3 Diffusion steering

Unlike regular world models, a flow-matching WAM has a policy head whose sampler is a deterministic map $g(\cdot; \mathbf{z})$ from a Gaussian draw $\epsilon \in \mathbb{R}^A$, to a chunk. Diffusion steering (DS) moves the decision variable of any sampling-based planner from the actions to that noise (any diffusion or flow sampler provides the map). The plan is denoted by $\epsilon_{1:H}$. Chunk h is decoded at the state imagined after chunks $1..h-1$,

$$a_h = g(\epsilon_h; \hat{\mathbf{z}}_{h-1}), \quad \hat{\mathbf{z}}_h = \hat{f}(\hat{\mathbf{z}}_{h-1}, a_h), \quad \hat{\mathbf{z}}_h = \text{shift}(\hat{\mathbf{z}}_{h-1}, \hat{\mathbf{z}}_h), \quad \hat{\mathbf{z}}_0 = \mathbf{z}_t, \quad (2)$$

where shift drops the oldest context latent and appends the predicted one, and the plan is scored by (1) on its final latent $\hat{\mathbf{z}}_H$. The search starts at the policy’s prior $\mathcal{N}(0, I)$, so the first iteration of every DS planner is best-of- N proposal ranking with $N = S$ over projected noise [4, 36]. DS-CEM, DS-MPPI and DS-CMA-ES share one budget (64×3 , horizon 5), the only difference between them is in the update rule.

4.4 Projection onto the sampler’s typical set

The head was trained to start from draws of $\mathcal{N}(0, I)$, which concentrate on the sphere of radius $\sqrt{A}$; as a planner narrows its search its proposals drift toward a mean of much smaller norm, which for a rectified-flow head acts like sampling at reduced temperature (mode-seeking). *Projected* DS maps every proposal back onto the sphere, $\epsilon \leftarrow \sqrt{A} \epsilon / \|\epsilon\|$, before decoding. As a consequence, the search is over *directions*, every decoded candidate is one the head was trained to produce, and the temperature effect is given up. The search distribution stays Euclidean; only proposals are projected. The search distribution stays Euclidean; only proposals are projected (ablated in Table 4).

Putting it altogether, we get Algorithm 1.

5 Experiments and Results

5.1 Tasks and data.

PushT [41, 25]: a circular agent pushes a T-shaped block to a fixed target pose; actions are absolute target positions at 10Hz in chunks of five. We use the 206 human demonstrations of Chi et al. [25] and paint two distractors that are independent of the actions: a dot on a Lissajous curve and a bar sliding along the top edge. *robomimic* [42]: Lift, Can, Square, ToolHang and Transport,

Algorithm 1 Diffusion steering of a sampling-based planner $\mathcal{P}$ (one decision)

Input: context $\mathbf{z}_t$, goal z_g , horizon H , samples S , iterations I , policy map g , forward mode $\hat{f}$, planner $\mathcal{P}$ with PROPOSE, UPDATE, SELECT (CEM: Gaussian refit to K elites, best elite; MPPI: softmax reweighting, weighted mean; CMA-ES: covariance and step-size adaptation, best sample)
 $\theta \leftarrow$ the policy’s prior $\mathcal{N}(0, I)$ over $\epsilon_{1:H}$
for $i = 1$ **to** I **do**
 $\epsilon_{1:H}^{(1:S)} \leftarrow \text{PROPOSE}(\theta)$
 $\epsilon_h^{(s)} \leftarrow \sqrt{A} \epsilon_h^{(s)} / \|\epsilon_h^{(s)}\|$ for every s, h (projection onto the sampler’s typical set)
 for each s **do**
 $\hat{\mathbf{z}} \leftarrow \mathbf{z}_t$
 for $h = 1$ **to** H **do**
 $a_h^{(s)} \leftarrow g(\epsilon_h^{(s)}; \hat{\mathbf{z}})$; $\hat{\mathbf{z}} \leftarrow \hat{f}(\hat{\mathbf{z}}, a_h^{(s)})$; $\hat{\mathbf{z}} \leftarrow \text{shift}(\hat{\mathbf{z}}, \hat{\mathbf{z}})$
 end for
 $c^{(s)} \leftarrow \|\hat{\mathbf{z}} - z_g\|$
 end for
 $\theta \leftarrow \text{UPDATE}(\theta, \epsilon^{(1:S)}, c^{(1:S)})$
end for
return the first chunk of $\text{SELECT}(\theta, \epsilon^{(1:S)}, c^{(1:S)})$, decoded at $\mathbf{z}_t$

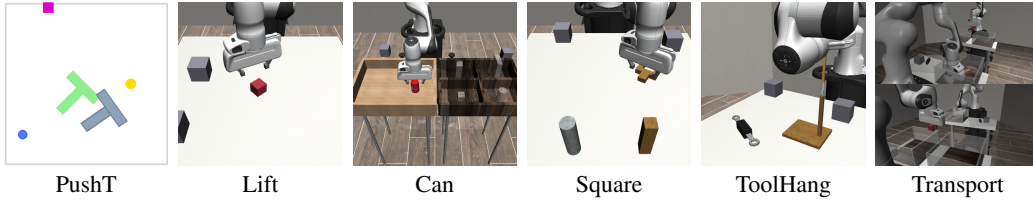


Figure 3: The six tasks as the models see them, with their distractors: painted dot and bar on PushT, two wobbling boxes on the robomimic tasks (cameras and resolutions in Section B).

using the benchmark’s 200 proficient-human demonstrations replayed in robosuite. We add two grey distractor boxes, larger than the task objects, that wobble in every scene without affecting the dynamics.

Every model is trained on these frames alone (Figure 3): pixels in, actions out, with no proprioception or state. Training runs for 300 epochs and five seeds on a 90% split of the data; evaluation uses the remaining 10% and runs in the same simulator with distractors present.

5.2 Baselines.

We train three baselines and a reference. All are our reimplementations that retain their authors’ original architectures but share one training protocol (pixels only, 300 epochs, five seeds, identical optimization and learning-rate schedules), sized to roughly match LeWAM’s 41.0M trainable parameters. *Dreamer* [1]: an RSSM trained by image reconstruction with an MLP action head. *LeWM* [13]: has no action head, so we freeze it and train LeWAM’s rectified-flow head on its latents with the same loss and schedule as LeWAM. *RecWAM*: the Cosmos Policy architecture [4] at LeWAM’s scale, a flow-matching transformer over the tokens of a VAE pretrained on each dataset’s frames, denoising the next frame and action chunk jointly. *Flow-matching policy*: LeWAM’s encoder and head trained through the policy loss only, so no world model.

5.3 Latent Space analysis

For each model and task we fit a ridge regression from the latent to agent, object, gripper and distractor state, with cross-validation to remove any decoder confounds, and evaluate it on a hold-

Table 2: Linear probe metrics across all tasks, 5 seeds (Negative R^2 clipped to 0).

Model	$R^2 = 1 - \text{MSE}/\text{Var}$		Pearson r		MSE	
	Task-state $\uparrow$	Distractors $\downarrow$	Task-state $\uparrow$	Distractors $\downarrow$	Task-state $\downarrow$	Distractors $\uparrow$
<i>PushT</i>						
Dreamer	0.56 ± 0.01	0.03 ± 0.03	0.78 ± 0.00	0.30 ± 0.01	0.39 ± 0.01	0.96 ± 0.03
LeWM + diff. head	0.54 ± 0.05	0.00 ± 0.02	0.77 ± 0.03	0.06 ± 0.04	0.40 ± 0.05	1.07 ± 0.02
RecWAM	0.64 ± 0.02	0.79 ± 0.03	0.82 ± 0.02	0.89 ± 0.02	0.32 ± 0.02	0.21 ± 0.03
LeWAM (ours)	0.81 ± 0.01	0.00 ± 0.02	0.92 ± 0.00	0.06 ± 0.02	0.16 ± 0.01	1.19 ± 0.02
<i>Lift</i>						
Dreamer	0.46 ± 0.01	0.81 ± 0.02	0.68 ± 0.00	0.90 ± 0.01	0.60 ± 0.01	0.19 ± 0.02
LeWM + diff. head	0.26 ± 0.02	0.00 ± 0.02	0.52 ± 0.02	0.01 ± 0.03	0.81 ± 0.02	1.08 ± 0.02
RecWAM	0.56 ± 0.01	0.86 ± 0.01	0.72 ± 0.01	0.93 ± 0.00	0.48 ± 0.01	0.14 ± 0.01
LeWAM (ours)	0.42 ± 0.02	0.00 ± 0.05	0.67 ± 0.01	0.16 ± 0.04	0.63 ± 0.02	1.20 ± 0.05
<i>Can</i>						
Dreamer	0.79 ± 0.01	0.01 ± 0.03	0.89 ± 0.01	0.17 ± 0.05	0.21 ± 0.01	0.99 ± 0.03
LeWM + diff. head	0.64 ± 0.01	0.00 ± 0.02	0.81 ± 0.00	0.02 ± 0.02	0.37 ± 0.01	1.04 ± 0.02
RecWAM	0.84 ± 0.01	0.61 ± 0.02	0.92 ± 0.01	0.78 ± 0.01	0.16 ± 0.01	0.39 ± 0.02
LeWAM (ours)	0.84 ± 0.00	0.00 ± 0.02	0.92 ± 0.00	0.20 ± 0.03	0.17 ± 0.01	1.07 ± 0.02
<i>Square</i>						
Dreamer	0.70 ± 0.01	0.07 ± 0.02	0.84 ± 0.01	0.27 ± 0.03	0.28 ± 0.01	0.93 ± 0.02
LeWM + diff. head	0.72 ± 0.01	0.00 ± 0.00	0.85 ± 0.01	-0.01 ± 0.01	0.27 ± 0.01	1.02 ± 0.00
RecWAM	0.87 ± 0.01	0.61 ± 0.01	0.93 ± 0.00	0.77 ± 0.01	0.13 ± 0.01	0.40 ± 0.01
LeWAM (ours)	0.90 ± 0.01	0.00 ± 0.02	0.95 ± 0.00	0.03 ± 0.01	0.10 ± 0.01	1.08 ± 0.02
<i>ToolHang</i>						
Dreamer	0.76 ± 0.00	0.23 ± 0.03	0.87 ± 0.00	0.40 ± 0.00	0.24 ± 0.00	0.77 ± 0.03
LeWM + diff. head	0.82 ± 0.02	0.00 ± 0.00	0.91 ± 0.01	0.01 ± 0.01	0.18 ± 0.02	1.00 ± 0.00
RecWAM	0.88 ± 0.00	0.75 ± 0.01	0.94 ± 0.00	0.86 ± 0.01	0.12 ± 0.00	0.25 ± 0.01
LeWAM (ours)	0.93 ± 0.00	0.00 ± 0.00	0.96 ± 0.00	-0.02 ± 0.01	0.07 ± 0.00	1.01 ± 0.00
<i>Transport</i>						
Dreamer	0.80 ± 0.03	0.00 ± 0.00	0.89 ± 0.02	0.03 ± 0.02	0.21 ± 0.03	1.01 ± 0.00
LeWM + diff. head	0.72 ± 0.01	0.00 ± 0.00	0.84 ± 0.01	0.01 ± 0.01	0.29 ± 0.01	1.01 ± 0.00
RecWAM	0.86 ± 0.01	0.76 ± 0.01	0.92 ± 0.01	0.87 ± 0.01	0.15 ± 0.01	0.24 ± 0.01
LeWAM (ours)	0.88 ± 0.00	0.00 ± 0.00	0.94 ± 0.00	-0.02 ± 0.01	0.12 ± 0.00	1.02 ± 0.00
<i>Average over tasks</i>						
Dreamer	0.68 ± 0.00	0.19 ± 0.01	0.83 ± 0.00	0.34 ± 0.01	0.32 ± 0.00	0.81 ± 0.01
LeWM + diff. head	0.62 ± 0.01	0.00 ± 0.01	0.78 ± 0.01	0.03 ± 0.01	0.39 ± 0.01	1.04 ± 0.01
RecWAM	0.78 ± 0.01	0.73 ± 0.01	0.87 ± 0.00	0.85 ± 0.00	0.22 ± 0.00	0.27 ± 0.01
LeWAM (ours)	0.80 ± 0.00	0.00 ± 0.01	0.89 ± 0.00	0.03 ± 0.01	0.21 ± 0.00	1.10 ± 0.01

out set. We track MSE, Pearson’s r and explained variance R^2 , both on the state variables that matter for the task and on those of the task-irrelevant distractors. Table 2 shows that, on average, LeWAM encodes task-relevant state better than LeWM and even the reconstruction-based models, and ignores distractors as well as LeWM.

To visually verify the quality of the latent space and the world model, we train a plain feed-forward convolutional decoder on a frozen pre trained LeWAM model. Note that gradients from the decoder do not change the latent space as this is done after LeWAM training. We show the rollouts from PushT and Robomimic Lift in Figure 4 tasks. For PushT we can see that the location of T block is consistently identified, but the rotation is blurry. This seems to be true for both the input context frames as well as the imagined frames. A point to note here is that the distractors are almost completely missing in the imagined rollouts. This further strengthens the linear probe results. In the Lift task, we see similar results, where the distractor blocks are either missing or wrongly predicted by the decoder. We believe the presence of the distractor block (but still in the wrong location) is an artifact of the decoder hallucinating.

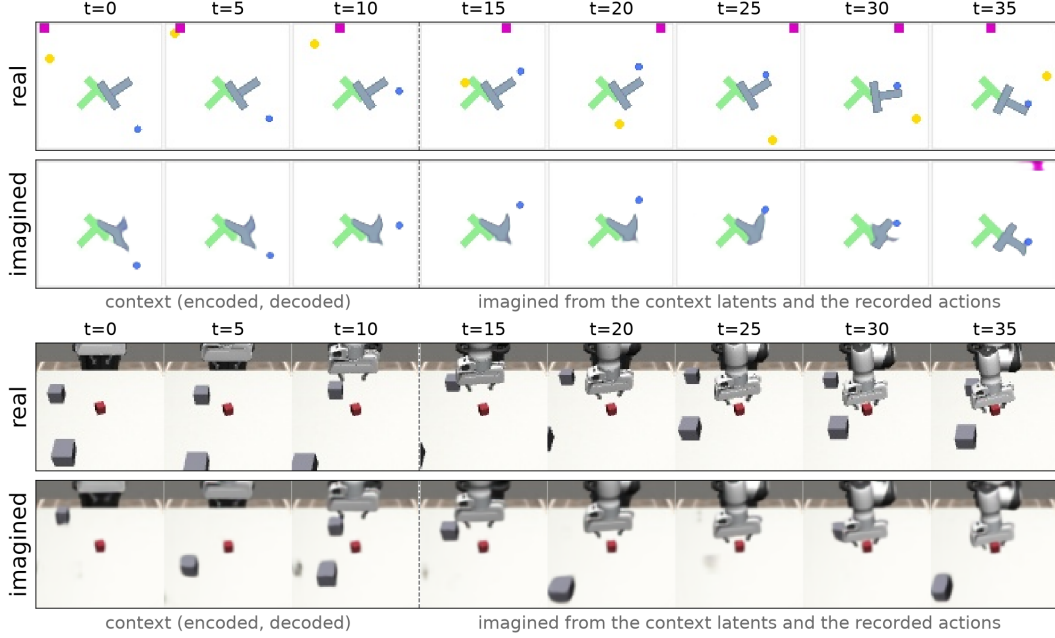


Figure 4: Imagined rollouts with LeWAM using ground truth actions. The top image is for PushT and the bottom one is for the Lift Robomimic task. The first 3 frames are context frames that is passed as input to the model. The 5 frames that follow are imagined frames by rolling out the LeWAM’s forward mode using ground truth actions.

5.4 Closed-loop evaluation.

We evaluate each model’s policy for 250 rollouts ($5 \text{ training} \times 50 \text{ evaluation seeds}$) on all six tasks with visual distractors. For each world model we also run MPC in its latent space, sampling both in action space (CEM, MPPI, CMA-ES) and in the policy’s noise space (DS-CEM, DS-MPPI, DS-CMA-ES); the goal is the latent 25 steps after the demonstration frame nearest the current latent. A BC-only flow-matching policy serves as reference.

Figure 5 shows that RecWAM’s policy is slightly below the reference, which we believe reflects capacity spent on pixel reconstruction, while Dreamer and LeWM + diffusion head perform very poorly. LeWAM’s policy maintains the reference’s performance. Sampling in noise space gives both LeWAM and RecWAM a performance gain at inference time, whereas action-space MPC collapses every model to 3–8% success, confirming that the planner exploits dynamics errors (Section 4.2).

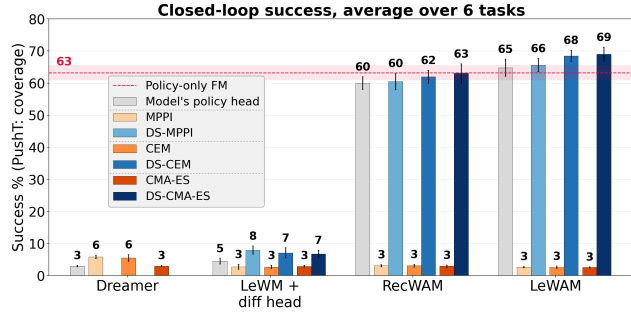


Figure 5: Closed-loop success of all models and planners, over 5 training seeds $\times$ 50 evaluation seeds (250 rollouts).

5.5 Ablation

To analyze the importance of each mode in LeWAM, we perform an ablation by switching off each mode during training and evaluate it on three robomimic tasks (Lift, Can and Square). From Table 3,

Table 3: LeWAM mode ablation. One mode removed at a time from the four-mode model (41M, object-distractor datasets, 300 epochs). R^2 : ridge probe from the frozen latent to task state, computed as $1 - \text{MSE}/\text{Var}$ on z-scored targets. Policy / DS-CEM: closed-loop success (%), 50 episodes per cell. The no-policy variant has no policy head, so its DS-CEM column steers the inverse-trained flow head.

Task	Variant	R^2 state $\uparrow$	Policy $\uparrow$	DS-CEM $\uparrow$
Lift	All four modes	0.423 ± 0.019	94.0 ± 4.2	97.6 ± 0.8
	No forward	0.323 ± 0.045	91.6 ± 2.3	95.6 ± 2.9
	No inverse	0.386 ± 0.031	92.0 ± 5.5	95.6 ± 2.9
	No policy	0.359 ± 0.019	–	0.8 ± 1.6
	No backward	0.370 ± 0.056	88.0 ± 4.6	93.6 ± 2.3
Can	All four modes	0.839 ± 0.004	82.8 ± 6.1	78.8 ± 4.5
	No forward	0.781 ± 0.018	77.6 ± 8.3	74.4 ± 7.7
	No inverse	0.817 ± 0.013	79.6 ± 8.0	73.2 ± 8.2
	No policy	0.761 ± 0.010	–	0.0 ± 0.0
	No backward	0.801 ± 0.015	77.6 ± 3.9	77.6 ± 2.7
Square	All four modes	0.896 ± 0.007	80.4 ± 1.5	87.2 ± 4.7
	No forward	0.876 ± 0.003	71.2 ± 8.4	73.6 ± 1.5
	No inverse	0.885 ± 0.003	82.0 ± 3.8	81.6 ± 3.2
	No policy	0.858 ± 0.015	–	0.0 ± 0.0
	No backward	0.872 ± 0.007	74.0 ± 7.7	82.8 ± 3.7

Table 4: Projection ablation. DS-CEM closed-loop success (%) with LeWAM, with and without projecting proposals onto the $\sqrt{A}$ shell before decoding (Section 4.4). mean $\pm$ std over seeds 0–4, 50 episodes per cell.

Task	With $\sqrt{A}$ shell	Without shell
Lift	97.6 ± 0.8	93.0 ± 3.0
Can	78.8 ± 4.5	23.0 ± 8.0
Square	87.2 ± 4.7	75.0 ± 3.0

we can see that all modes add value to both forming an informative latent space as well as closed-loop performance on average over tasks. State information in the latent space, policy rollout and planning all seem to improve with all four modes. Forward and backward consistently help; inverse is neutral on Square.

We also ablate the projection step of Section 4.4, keeping the budget and goals of Section 4.3. Removing it lowers DS-CEM success on all three tasks (Table 4), by about 5 points on Lift, 12 on Square and 56 on Can.

6 Conclusion and Future work

This paper presents LeWAM, a decoder-free JEPA-based world action model. We show that LeWAM’s latent space is more informative than LeWM’s, and is void of the visual features that do not affect task dynamics. This property allows LeWAM to better model policy actions. Finally, we show that by sampling in diffusion noise space (an inference-only procedure), we can effectively plan with LeWAM and improve average closed-loop performance across tasks. We believe the two logical next steps are: (1) applying LeWAM to real-world data, which contains an abundance of dynamically inert features; and (2) improving planning algorithms with LeWAM, as these are slow and also require a notion of a goal at inference time.

References

- [1] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.
- [2] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *Nature*, 640:647–653, 2025.
- [3] C. Zhu, R. Yu, S. Feng, B. Burchfiel, P. Shah, and A. Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. In *Robotics: Science and Systems (RSS)*, 2025. arXiv:2504.02792.
- [4] M. J. Kim et al. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.
- [5] S. Ye, Y. Ge, K. Zheng, S. Gao, S. Yu, G. Kurian, S. Indupuru, Y. L. Tan, C. Zhu, J. Xiang, A. Malik, K. Lee, W. Liang, N. Ranawaka, J. Gu, Y. Xu, G. Wang, F. Hu, A. Narayan, J. Bjorck, J. Wang, G. Kim, D. Niu, R. Zheng, Y. Xie, J. Wu, Q. Wang, R. Julian, D. Xu, Y. Du, Y. Chebotar, S. Reed, J. Kautz, Y. Zhu, L. J. Fan, and J. Jang. World action models are zero-shot policies, 2026. URL <https://arxiv.org/abs/2602.15922>.
- [6] A. Zhang, Y. Wu, and J. Pineau. Natural environment benchmarks for reinforcement learning. In *arXiv preprint arXiv:1811.06032*, 2018.
- [7] A. Stone, O. Ramirez, K. Konolige, and R. Jonschkowski. The distracting control suite: A challenging benchmark for reinforcement learning from pixels. In *arXiv preprint arXiv:2101.02722*, 2021.
- [8] Y. LeCun. A path towards autonomous machine intelligence. *OpenReview preprint*, 2022.
- [9] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *CVPR*, 2023.
- [10] A. Bardes, Q. Garrido, J. Ponce, X. Chen, M. Rabbat, Y. LeCun, M. Assran, and N. Ballas. Revisiting feature prediction for learning visual representations from video. In *TMLR*, 2024.
- [11] F. Deng, I. Jang, and S. Ahn. DreamerPro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *ICML*, 2022.
- [12] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. In *ICLR*, 2021.
- [13] L. Maes, Q. Le Lidec, D. Scieur, Y. LeCun, and R. Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.
- [14] R. Balestriero and Y. LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.
- [15] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *ICLR*, 2023.
- [16] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.
- [17] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022.

- [18] B. Yang, H. Su, N. Gkanatsios, T.-W. Ke, A. Jain, J. Schneider, and K. Fragkiadaki. Diffusion-ES: Gradient-free planning with diffusion for autonomous and instruction-guided driving. In *CVPR*, 2024.
- [19] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.
- [20] A. Argenson and G. Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2008.05556.
- [21] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine. Steering your diffusion policy with latent space reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2025. arXiv:2506.15799.
- [22] H. Simsir and O. S. Oguz. Lagrangian perturbation diffusion steering: Latent reinforcement learning for generative policies. *arXiv preprint arXiv:2606.01151*, 2026.
- [23] O. Patil, O. Biza, T. Weng, K. Schmeckpeper, W. Thomason, X. Zhang, K. Sivakumar, R. Walters, N. Gopalan, S. Castro, S. Hart, and E. Rosen. You’ve got a golden ticket: Improving generative robot policies with a single noise vector. *arXiv preprint arXiv:2603.15757*, 2026.
- [24] N. Hansen, H. Su, and X. Wang. TD-MPC2: Scalable, robust world models for continuous control. In *ICLR*, 2024.
- [25] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.
- [26] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.
- [27] R. Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.
- [28] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *ICRA*, 2017.
- [29] N. Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.
- [30] G. Zhou, H. Pan, Y. LeCun, and L. Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.
- [31] M. Assran, A. Bardes, D. Fan, Q. Garrido, R. Howes, M. Komeili, M. Muckley, A. Rizvi, C. Roberts, K. Sinha, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.
- [32] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *NeurIPS*, 2018.
- [33] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. In *ICML*, 2022.
- [34] J. Liang, R. Liu, E. Ozguroglu, S. Sudhakar, A. Dave, P. Tokmakov, S. Song, and C. Vondrick. Dreamitate: Real-world visuomotor policy learning via video generation. In *8th Annual Conference on Robot Learning*, 2024. URL <https://openreview.net/forum?id=InT87E5sr4>.
- [35] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0 : A vision-language-action flow model for general robot control. In *arXiv preprint arXiv:2410.24164*, 2024.

- [36] M. Nakamoto, O. Mees, A. Kumar, and S. Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Conference on Robot Learning (CoRL)*, 2024.
- [37] G. Zhou, S. Swaminathan, R. Vasudeva Raju, J. S. Guntupalli, W. Lehrach, J. Ortiz, A. Dedieu, M. Lázaro-Gredilla, and K. Murphy. Diffusion model predictive control. *arXiv preprint arXiv:2410.05364*, 2024.
- [38] H. Qi, H. Yin, A. Zhu, Y. Du, and H. Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 2025. arXiv:2502.00622.
- [39] C. Hamel. FlowMPC: Improving flow matching policies with world models. *arXiv preprint arXiv:2606.16286*, 2026.
- [40] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. DINOv2: Learning robust visual features without supervision. In *TMLR*, 2024.
- [41] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *CoRL*, 2022.
- [42] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

A Results per Task

While Figure 5 shows closed loop evaluation results averaged across all tasks, we show the per-task closed loop evaluation and MPC results in Figure 6.

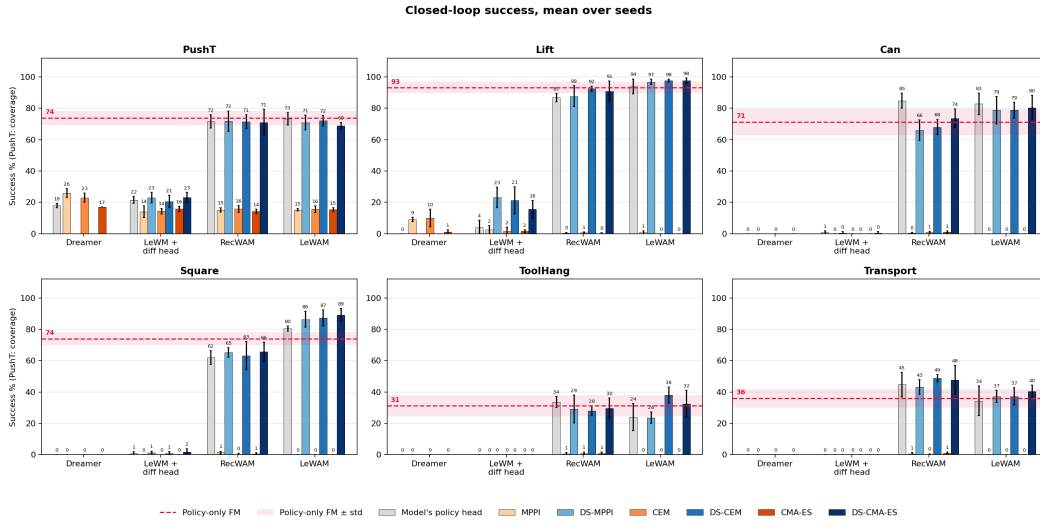


Figure 6: Average closed-loop performance of all models/MPC, for each task, over 5 training seeds, 50 evaluation seeds: 250 unique rollouts.

B Configuration of the tasks

Cameras and resolutions. PushT is rendered at 96 px on a 512 px table and resized to 224 px for the encoder. Lift, Can and Square are rendered at 84 px from the agent-view camera; ToolHang at

168 px from the benchmark’s side camera, which keeps the tool, its frame and the boxes in view; and Transport at 256 px as the two shoulder cameras, one per arm, stacked into a single frame so that the hand-over between the arms is visible. All frames are resized once to 224 px. Evaluation uses 50 seeded simulator resets per seed, with the boxes moving.

Configuration. Table 5 is the single LeWAM configuration used on all tasks. Note: the training window’s context is truncated to a random length (one to three frames, left-aligned, the newest frame always present) instead of masked, so that the first decision of an episode, where the simulator repeats the reset frame, is supervised.

Table 5: LeWAM on the robomimic tasks: one configuration for Lift, Can, Square, ToolHang and Transport.

data	200 demonstrations per task, frames resized once to 224 px, frameskip 5, windows of 3 context frames + 1 target, 5-step action chunks (35-D; 70-D on Transport), z-scored
schedule	300 epochs, batch 128, AdamW (lr 5×10^{-5} , weight decay 10^{-3}), linear warm-up then cosine, bf16, gradient clip 1.0
encoder	ViT-tiny/14 at 224 px, [CLS] token, projector MLP (hidden 2048) to a 384-D latent
regularizer	SIGReg on the frame latent, weight 0.36 ($0.09 \times$ the four modes), 17 knots, 1024 projections
backbone	bidirectional transformer, depth 8, width 512, 16 heads $\times$ 64, MLP 2048
modes	all four on every batch; forward mode weight 2, inverse, policy and backward weight 1
context	random context length (1 to 3 frames, left-aligned truncation); no context-frame masking
policy head	flow matching, 8 Euler steps at inference
evaluation	50 seeded starts per seed, the benchmark horizon per task, boxes moving

C Model sizes

All four models are matched in trainable parameters. LeWAM has 41.0M: a ViT-tiny/14 encoder (width 192, depth 12, 3 heads $\times$ 64) with its projector, and a bidirectional transformer of width 512, depth 8, 16 heads $\times$ 64 and MLP 2048. LeWM’s autoregressive predictor is widened from the released configuration (width 192, depth 6; 18.0M in all) to width 512, 16 heads $\times$ 64, MLP 2048 at depth 6, for 42.2M in all. RecWAM’s flow transformer is widened and deepened to width 512, depth 10, 16 heads $\times$ 64, MLP 2048, for 42.6M trainable parameters, plus a 3.9M VAE tokenizer that is pretrained per dataset and frozen. Dreamer’s RSSM is trained by image reconstruction at 96 px with the balanced KL, and is enlarged from the standard configuration (13.6M) by widening every component: deterministic state 512 to 1536, stochastic state 32 to 64, GRU and MLP hidden 512 to 1280, image embedding 1024 to 1280, for 43.4M.

D Protocol and training hyperparameters

Probes. Ridge regression from the latent to agent location, block location, block angle and the two distractor positions on PushT, and to the end-effector position, the gripper opening, the task object’s position and the two boxes on robomimic; $R^2 = 1 - \text{MSE}/\text{Var}$ on held-out episodes, with five-fold cross-validation over the penalty and a shuffled-target control, the same regression fit to permuted targets, which sits at or below zero for every model and task.

Planning protocol details. Every planner uses the same budget, 64 samples, 3 iterations and 8 elites: the action-space planners, CEM, MPPI and CMA-ES, over action sequences of 5 chunks. For Dreamer, whose MLP head is deterministic, the 64 ranked candidates are the head’s chunk plus 63 Gaussian perturbations of it with standard deviation 0.2 in z-scored action units, scored by Dreamer’s own planning cost, the pixel error of the decoded rollout to the goal frame; the head’s chunk is always a candidate, so ranking cannot fall below the policy, and no DS variant exists for it, since there is no noise-to-action map to search over. Goals are retrieved in each model’s own latent space, by encoding the observation 25 steps ahead of the closest latent from a given demonstration.

The robomimic horizons are 100 steps on Lift, 400 on Can and Square, and 700 on ToolHang and Transport; LeWAM uses the configuration of Table 5 on every robomimic task.