H-JEPA: End-to-End Learning of Hierarchical World Models for Visual Planning

Anonymous authors
Paper under double-blind review

Abstract

Long-horizon planning with latent world models requires reasoning across timescales and levels of abstraction. Existing task-agnostic JEPA world models predict and plan at a single timescale or with multiple horizons in one shared latent space. We introduce **H-JEPA**, an end-to-end recipe for training a hierarchy of action-conditioned JEPAs in which each level predicts farther ahead in its own learned latent space. Planning proceeds top-down: the top level optimizes progress toward the goal, and each level's predictions become subgoals for the planner below it. When factors in the data evolve at separated timescales, higher levels discard fast, unpredictable detail and retain slower task-relevant state. Across four simulated navigation and manipulation environments, hierarchical planning improves over a flat JEPA; on Visual AntMaze, a three-level hierarchy raises success from 18% to 73% using less planner compute. Ablations attribute these gains to both temporal decomposition and higher-level goal representations. With inverse-dynamics supervision, the approach extends to diverse real-robot videos from DROID, where hierarchy improves offline planning fidelity at lower planner compute.

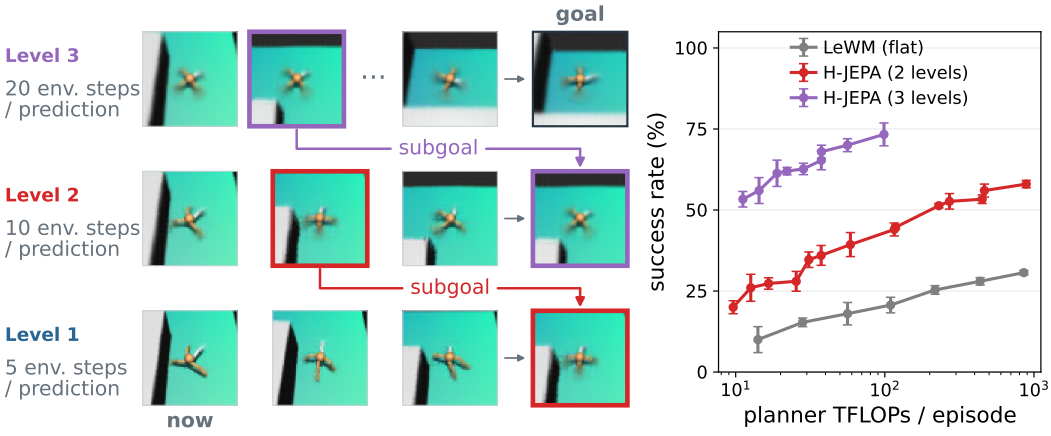

Figure 1: **Hierarchical abstractions support planning at multiple timescales.** *Left:* visualization of hierarchical planning in Visual AntMaze. Level 3 plans toward the final goal in the most abstract latent space. Its first predicted state becomes a subgoal for the level-2 planner; this decomposition recurses to the level 1 planner, which outputs raw actions. Each lower-level planner encodes its predictions into the upper-level latent space before evaluating subgoal costs. The first and last columns decode each level's start and goal latents, respectively; the middle columns decode its planned predictions. Higher-level latents retain the ant's position and maze layout while losing leg-pose detail. See §2.2 for the formal planning procedure. *Right:* measured planning success versus planner compute on AntMaze. Curves show Pareto fronts for flat LeWM [52] and two-/three-level H-JEPA; error bars are standard error (SE) over three paired model and planning seeds (§4.1).

1 Introduction

World models learn the dynamics of an environment from experience so that an agent can predict, understand, and plan [29, 30, 31]. Joint-Embedding Predictive Architectures (JEPAs) learn such

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.

models by predicting future *latent* states rather than pixels [48, 1, 9], avoiding both reward and pixel reconstruction [5, 58], and recent action-conditioned, task-agnostic JEPA world models plan zero-shot in the learned latent space [92, 2, 72, 52, 77]. Most of them, however, are *flat*: a single latent space, with a predictor at a single timescale, covers every use of the model. A single timescale and a single level of abstraction limit such a model in two ways. The first is temporal. Reaching far into the future means rolling the predictor out over many fine-grained steps: error compounds with every step, and the action search space grows exponentially with the horizon. The second is representational. A latent that carries every fast-varying detail may model low-level dynamics well, but it is a poor goal-matching objective whenever the goal is more abstract than the state (a location to reach rather than a pose to match): the surplus detail adds goal-irrelevant variation to the planning cost, so latent distance no longer tracks progress toward the goal (§4.2).

A hierarchical world model architecture in which each higher level models a longer timescale and retains in its latent only what is predictable over it offers a division of labor: higher levels, operating on slow features over long strides, serve long-range prediction and high-level planning, while lower levels, operating on fast features over short strides, serve short-range prediction and low-level control. The brain has such hierarchical structure: cortical areas form a hierarchy of intrinsic timescales, with slower-varying representations higher up [37, 43, 55, 3]. Prior hierarchical world models fall short of this on one axis or another (§M, Table 13). Hierarchical latent video models reconstruct pixels and are not used for control [67, 44, 54], and pre-existing hierarchical model-based reinforcement learning operates in the reward-driven, task-specific setting [32, 36, 27]. HWM [91] is a task-agnostic JEPA world model that plans hierarchically, yet its predictors over different horizons share a single latent space: it decomposes the horizon, but cannot match objectives in more abstract spaces.

We introduce **H-JEPA**, a hierarchical JEPA architecture in which each level predicts the future within its own latent space, at a coarser temporal stride and a higher degree of abstraction than the level below, with no reward and no reconstruction training objectives. Fig. 1 illustrates the learned hierarchy alongside its planning compute–success tradeoff. We make four contributions:

1. We introduce an end-to-end training method for hierarchical JEPA world models (§2).
2. We show that higher levels learn selective abstractions: they discard fast features that are unpredictable over their horizons while retaining slow, predictable features (§3). This selectivity is stronger when entities evolve at more widely separated timescales.
3. We show that H-JEPA's hierarchical planning outperforms single-level planning at a fraction of the test-time compute (§4.1). We identify two complementary mechanisms: temporal decomposition breaks long-horizon tasks into easier subproblems, while hierarchical latent spaces let higher-level planners score progress at the goal's own level of abstraction (§4.2).
4. With an inverse-dynamics term, the method extends to DROID, a real robot-manipulation corpus whose scene, lighting and objects change every episode (§4.3).

2 METHOD

2.1 HIERARCHICAL PREDICTIVE LEARNING

H-JEPA learns a hierarchy of latent predictive models from trajectories of observations and actions. The hierarchy is temporal: level 1 operates on the finest observation stream, while each higher level consumes the latent states produced by the level below over a coarser stride. This gives a stack of JEPA world models in which each level has its own observation encoder, action encoder, and latent predictor. All levels predict future latent states from past latent states and actions.

Let o_t denote an observation and a_t the action block for the transition from o_t to o_{t+1}. The first level encodes the observation, optionally with the proprioceptive state, into a latent state $z_t^{(1)} = E^{(1)}(o_t)$, and the action block into $a_t^{(1)} = A^{(1)}(a_t)$. Higher levels are built compositionally: both state and action encoders pool temporal windows of latents from the level below.

Each level $\ell > 1$ has two temporal hyperparameters. The *stride* s_ℓ is the subsampling factor: upper-level time t maps to lower-level time $t \cdot s_\ell$, so consecutive upper-level states are s_ℓ lower-level steps apart. The *window size* w_ℓ is the number of lower-level steps each upper-level state summarizes. The two are independent: s_ℓ sets how far each window advances and w_ℓ how far it spans.

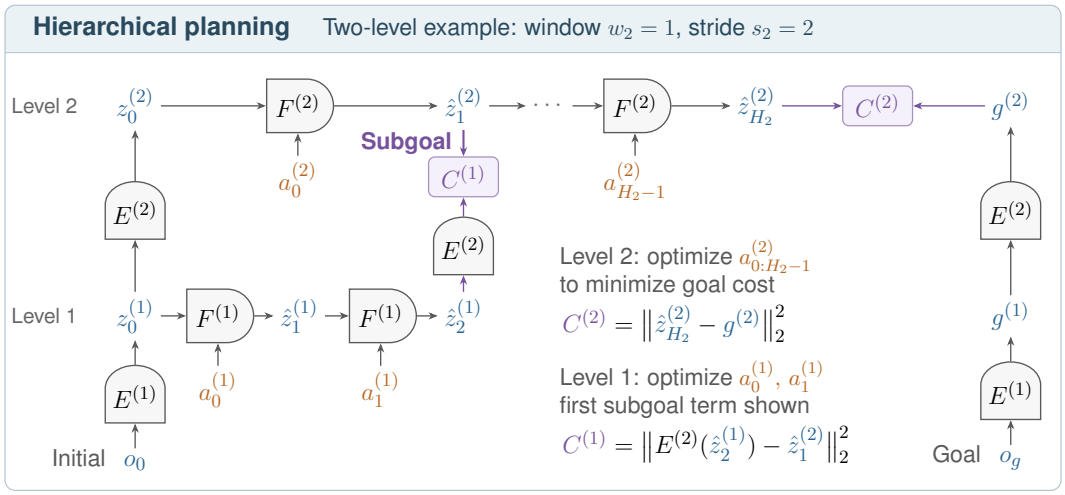

Figure 2: **Planning runs top-down through the hierarchy, and the lower level is steered by subgoals the upper level predicted.** Two-level instance (L=2) of §2.2. The current observation o_0 and the goal observation o_g are encoded bottom-up into initial states $z_0^{(1)}, z_0^{(2)}$ and goal states $g^{(1)}, g^{(2)}$. Level 2 optimizes macro-actions by unrolling $F^{(2)}$ and matching its final prediction to $g^{(2)}$. Its predicted states $\hat{z}_i^{(2)}$ become subgoals for level 1, which optimizes primitive actions and lifts its predicted states through $E^{(2)}$ to compare them in the level-2 latent space (Eq. (6)). Shown for closed-loop planning with $K_2 = 1$, $s_2 = 2$ and $w_2 = 1$: level 1 matches its encoded two-step prediction to the first optimized level-2 prediction. An action prefix is executed before replanning.

Where a subscript $a{:}b$ selects the $b{-}a$ indices $a, \ldots, b{-}1$, the level-ℓ state and action embeddings are

$$z_t^{(\ell)} = E^{(\ell)}\left(z^{(\ell-1)}_{t\cdot s_\ell\,:\,t\cdot s_\ell + w_\ell}\right), \qquad a_t^{(\ell)} = A^{(\ell)}\left(a^{(\ell-1)}_{t\cdot s_\ell + w_\ell - 1\,:\,(t+1)\cdot s_\ell + w_\ell - 1}\right). \tag{1}$$

The state encoder $E^{(\ell)}$ pools the w_ℓ-step lower-level window into one abstract state. The action encoder $A^{(\ell)}$ instead aggregates s_ℓ lower-level action embeddings, independent of w_ℓ, so $a_t^{(\ell)}$ covers the full transition from the state anchoring $z_t^{(\ell)}$ toward the next upper-level state $z_{t+1}^{(\ell)}$. All experiments in this paper use $w_\ell{=}1$, so upper-level state encoders are pointwise. Fig. 8 in §C illustrates a two-level architecture with stride $s_2 = 2$ and window size $w_2 = 2$ to show temporal pooling.

Each level is trained as a JEPA in its own latent space. Given a context of c_ℓ latent states $(z^{(\ell)}_{t-c_\ell+1}, \ldots, z^{(\ell)}_t)$ and the associated action embeddings, a predictor $F^{(\ell)}$ predicts future latent states. One-step training uses the teacher-forced latent prediction loss

$$\mathcal{L}^{(\ell)}_{\text{pred}} = \frac{1}{c_\ell}\sum_{\tau=1}^{c_\ell}\left\|F^{(\ell)}(z^{(\ell)}_{<\tau}, a^{(\ell)}_{<\tau}) - z^{(\ell)}_{\tau+1}\right\|_2^2, \tag{2}$$

The per-level loss also includes SIGReg [6], a sketched normality regularizer that prevents collapse by encouraging an isotropic Gaussian embedding distribution. More details are in §K. Let $Z^{(\ell)} \in \mathbb{R}^{n\times D}$ collect n level-ℓ embeddings of width D, flattened over batch and time. the overall level-ℓ objective is

$$\mathcal{L}^{(\ell)} = \mathcal{L}^{(\ell)}_{\text{pred}} + \lambda_\ell \text{SIGReg}\left(Z^{(\ell)}\right). \tag{3}$$

We train H-JEPA end-to-end, jointly optimizing all levels with their per-level losses. A stagewise variant instead trains level 1 alone, then freezes it and trains level 2 on its latents. We use it only for analyses (§G, Fig. 19 and Fig. 17); every headline result is end-to-end.

2.2 Hierarchical planning over learned abstractions

H-JEPA plans top-down through the learned hierarchy (Fig. 2). The current and goal observations are encoded at every level ℓ, yielding initial latent states $\{z_0^{(\ell)}\}_{\ell=1}^L$ and goal latent states $\{g^{(\ell)}\}_{\ell=1}^L$, where L is the number of levels. The top level proposes a coarse plan to the goal. Each lower level

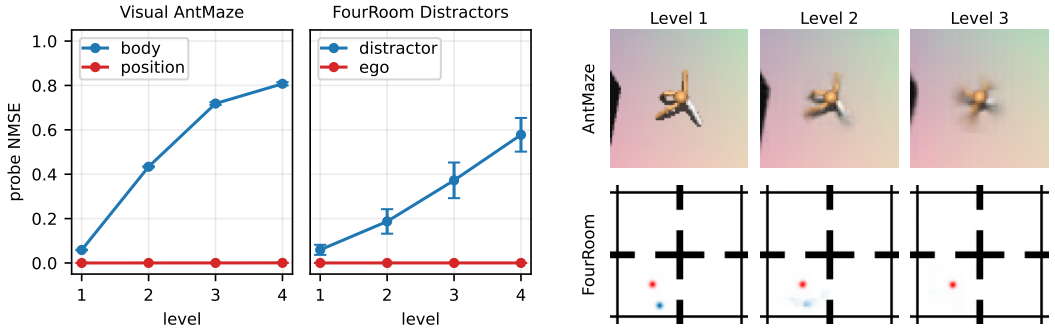

Figure 3: **Selective abstraction across hierarchy levels.** *Left*: MLP-probe NMSE (mean ± standard error (SE) over three training seeds) at successive levels of four-level H-JEPAs on Visual AntMaze and FourRoom Distractors. Body state or distractor position becomes less recoverable at higher levels while agent position remains recoverable. *Right*: one observation decoded from levels 1–3; decoders serve only for visualization. AntMaze's leg pose becomes less distinct; FourRoom's blue distractor fades while the red agent remains visible.

plans toward subgoals supplied by the predicted trajectory of the level above, refining the plan into finer-scale transitions until level 1 produces actions that can be executed in the environment. The top-level planner optimizes macro-actions to minimize the distance between its final predicted state and the goal:

$$a^{(L),*}_{0:H_L-1} = \arg\min_{a^{(L)}_{0:H_L-1}} \left\| \hat{z}^{(L)}_{H_L} - g^{(L)} \right\|_2^2 . \tag{4}$$

At each level ℓ, the predicted states $(\hat{z}^{(\ell)}_1, \ldots, \hat{z}^{(\ell)}_{H_\ell}) = \mathrm{Roll}^{(\ell)}(z^{(\ell)}_0, a^{(\ell)}_{0:H_\ell-1})$ are obtained by autoregressively applying predictor $F^{(\ell)}$ from $z^{(\ell)}_0$ under the candidate action sequence, with horizon H_ℓ measured in level-ℓ steps. A superscript $*$ marks the optimized actions and their predicted states.

For each lower level $\ell < L$, the optimized rollout from the level above, $(\hat{z}^{(\ell+1),*}_1, \ldots, \hat{z}^{(\ell+1),*}_{H_{\ell+1}})$ supplies subgoals. The lower level optimizes its actions to match these subgoals with its own predicted states, encoded into the upper-level latent space by $E^{(\ell+1)}$.

We first choose the number of subgoals to match, $1 \le K_{\ell+1} \le H_{\ell+1}$, then set the lower-level horizon to cover the corresponding encoder windows. With upper-level stride $s_{\ell+1}$ and window size $w_{\ell+1}$, this requires $H_\ell = K_{\ell+1}s_{\ell+1} + w_{\ell+1} - 1$. The encoded predictions are

$$\tilde{z}^{(\ell+1)}_i = E^{(\ell+1)}\left(\hat{z}^{(\ell)}_{is_{\ell+1}:\, is_{\ell+1}+w_{\ell+1}}\right), \qquad i = 1, \ldots, K_{\ell+1}. \tag{5}$$

Each $\tilde{z}^{(\ell+1)}_i$ is aligned with its upper-level subgoal in time and latent space. The lower-level solves

$$a^{(\ell),*}_{0:H_\ell-1} = \arg\min_{a^{(\ell)}_{0:H_\ell-1}} \left[\left\| \tilde{z}^{(\ell+1)}_{K_{\ell+1}} - \hat{z}^{(\ell+1),*}_{K_{\ell+1}} \right\|_2^2 + \beta \sum_{i=1}^{K_{\ell+1}-1} \left\| \tilde{z}^{(\ell+1)}_i - \hat{z}^{(\ell+1),*}_i \right\|_2^2 \right], \tag{6}$$

where $\beta \ge 0$ weights each intermediate subgoal relative to the terminal target. Closed-loop planning uses $K_{\ell+1} = 1$, matching only the first predicted subgoal; open-loop planning uses $K_{\ell+1} = H_{\ell+1}$, with $\beta = 0$ matching only the terminal prediction and $\beta > 0$ also matching intermediate subgoals.

We optimize action sequences at every level by gradient descent. Level 1 returns primitive actions for execution in the environment. In closed-loop control, we execute an action prefix, encode the new observation, and repeat hierarchical planning until the evaluation budget of environment steps is exhausted. §E gives execution schedules and solver settings.

3 Learning Hierarchical Abstractions

Environments. We use FourRoomDistractors and Visual AntMaze for navigation, and Push-T and OGBench Cube for manipulation. FourRoomDistractors pairs a controlled agent with a distractor that moves independently and smoothly between occasional random teleports, while Visual AntMaze

requires a quadruped to navigate a maze. Push-T involves pushing a T-shaped block, and OGBench Cube involves picking up and placing a cube. §B describes the environments and datasets in detail. In §4.3, we extend the evaluation to DROID, a dataset of real robot manipulation trajectories.

Setup. We train H-JEPAs up to four levels end-to-end, with gradients from the per-level objectives propagating through the levels. The level-1 JEPA is LeWM [52]: a ViT-Tiny encoder that summarizes each image in a global representation through its `[CLS]` token, and a causal transformer predictor. Trained on its own, this single-level model is the flat baseline we compare against throughout. Higher-level JEPAs use two-layer MLP encoders and causal transformer predictors, with stride $s_\ell{=}2$ and window $w_\ell{=}1$, so a single prediction step at each higher level spans twice as many original environment timesteps. Within each environment, all levels share the same SIGReg coefficient. §C provides architecture and training details.

Evaluation protocol. We use frozen two-layer MLP probes on detached representations, reporting variance-normalized MSE (NMSE), averaged with equal weight over an entity's dimensions. A value near 1 is the error of predicting the marginal mean, while a value near 0 indicates accurate recovery. Quantitative results in this section are mean $\pm$ SE over three training seeds; pixel decodings are qualitative examples from post-hoc decoders. §D details probe and decoder training.

Higher levels discard fast-varying features while retaining slow ones. In Fig. 3, AntMaze's body state and FourRoom's distractor position become less recoverable with depth, while the agent's position remains accurately recoverable throughout both hierarchies. The discarded detail is consistent with each level's prediction horizon: the FourRoom distractor's motion is locally predictable between teleports, but longer horizons are more likely to cross a random teleport, making its future position harder to predict. In AntMaze, the ant's joint configurations oscillate many times faster than its global position drifts (§F.1), so they are harder to predict over long horizons. These trends are consistent with JEPA's joint optimization of the encoder and predictor: prediction-error gradients flow through the encoder, encouraging each level to retain features it can predict at its own timescale and abstract away those it cannot. In these environments, H-JEPA thus learns a hierarchy in which lower levels retain both fast- and slow-varying features, while higher levels progressively discard high-frequency detail and preserve the slow features.

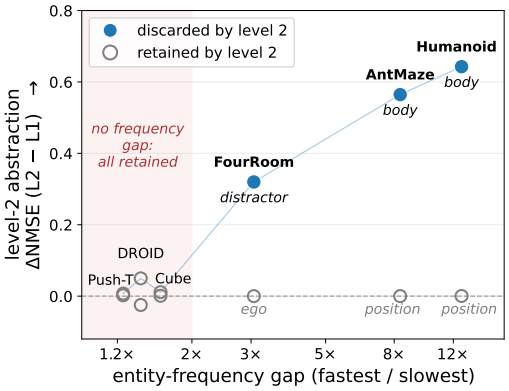

Figure 4: **Larger temporal-frequency gaps correlate with stronger selective abstraction.** The x-axis measures temporal separation in each training dataset: the ratio of the characteristic frequencies (spectral centroids) of the fastest- and slowest-varying state components, shown on a log scale. The y-axis shows the change in per-entity probing NMSE from level 1 to level 2; positive values indicate poorer recoverability at level 2. Across environments, larger frequency gaps coincide with greater increases in fast-entity probing error (filled markers), while slow-entity error remains near its level-1 value (open markers). Environments with gaps below 2× (shaded region) show little selective abstraction. §F.2 reports per-entity results.

Selective abstraction does not emerge in every environment. We relate the change in per-entity probing error from level 1 to level 2 to the training dataset's *entity frequency gap*: the ratio of the characteristic frequencies (spectral centroids) of the fastest- and slowest-varying state components, which define the fast and slow entities, respectively. §F.1 illustrates this separation in an AntMaze trajectory. Across evaluated environments, larger gaps accompany greater loss of fast-entity information at level 2, while slow-entity recoverability remains near its level-1 value (Fig. 4). This separation appears in AntMaze, Humanoid, and FourRoom; the manipulation datasets have smaller gaps and show little selective abstraction. See §F.2 for gap measurements.

4 Planning with Hierarchical Abstractions

Planning top-down through the learned hierarchy improves long-horizon control. We first compare end-to-end H-JEPA with flat LeWM [52] across test-time compute budgets and hierarchy depths,

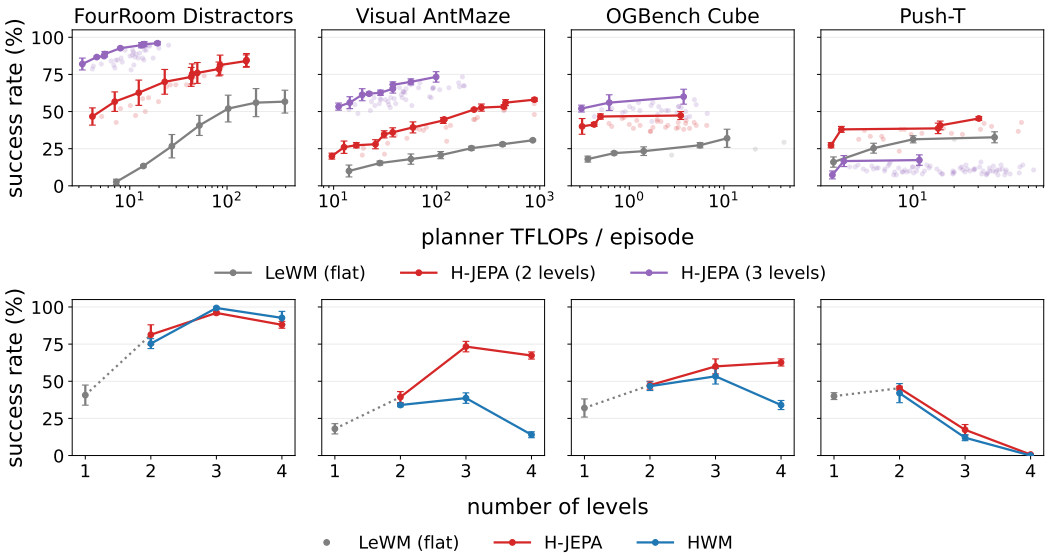

Figure 5: **Gradient-based planning across compute budgets and hierarchy depths**, comparing LeWM [52], H-JEPA, and the hierarchical world model baseline HWM [91]. Error bars in both panels show one standard error (SE) over three seeds. *Top:* success versus planner FLOPs per episode, sweeping action-sample counts with other planner settings fixed. FLOPs include forward and backward passes. Bold curves are Pareto fronts. *Bottom:* success versus hierarchy depth on a separate set of 50 tasks per environment. LeWM and H-JEPA settings are selected on the top-row sweep tasks under a 100-TFLOP budget. See §E for protocols and Fig. 7c for the DROID results.

and with HWM [91], whose temporal hierarchy is confined to one latent space. We then examine how learned abstractions reshape the planning cost surface and how temporal decomposition breaks long-horizon control into easier subproblems. Both evaluations use simulated environments with a fixed camera and background. We then scale the method to diverse scenes, on real teleoperated robot video from DROID. Task construction and planner settings for all evaluations are in §E.

4.1 Planning performance across compute budgets and depths

H-JEPA scales LeWM in depth: **LeWM** is the single-level planner, and **H-JEPA** adds levels above it, planning across distinct latent spaces with the top-down procedure of §2.2. We test up to four levels, first examining how planning success scales with test-time compute (Fig. 5, top). The sweep varies the number of trajectories optimized by gradient descent at each planning level, with other planner settings and tasks fixed within the sweep. The depth comparisons and cost ablations use a separate evaluation task set (§E). On FourRoom, AntMaze and Cube, each additional hierarchy level, up to three levels, shifts the Pareto frontier up and to the left, achieving higher success rates with less planner compute. Push-T is the exception: two-level H-JEPA matches or exceeds LeWM at every budget, but the three-level model performs poorly even with maximum compute. The likely cause is training data. Clips must lie within one episode, and Push-T episodes are short relative to the minimum number of clips required to train a three level model, so it sees only 58% of LeWM's transitions per epoch (Table 3); the other environments retain 75–88% at three levels.

Distinct representations within a temporal hierarchy. We compare H-JEPA depth for depth with **HWM** [91], a hierarchical planner confined to one latent space, asking whether distinct latent spaces add to the benefit of temporal hierarchy alone. For a controlled comparison, HWM (blue curves in Fig. 5, bottom) uses the same implementation, data and training settings as H-JEPA at each depth, except that encoders above level 1 are identity maps. All levels therefore predict and plan in the same latent space. The improvement of H-JEPA over HWM is most pronounced on AntMaze, where three-level H-JEPA reaches nearly twice HWM's success. On Cube, H-JEPA also scores higher at higher depths, although more moderately relative to AntMaze. On FourRoom and Push-T, both hierarchies perform similarly. Consistent with earlier findings, both methods deteriorate on Push-T at three levels and beyond, likely because short episodes limit usable training data: three- and four-level H-JEPA see only 58% and 14% of LeWM's training transitions per epoch, respectively (Table 3).

	Model depth		
Cost / planner	2 levels ↑	3 levels ↑	4 levels ↑
Native L1	23.3 ± 3.3	16.7 ± 1.8	10.0 ± 0.0
L2 projection	$\mathbf{31.3 \pm 2.7}$	20.7 ± 4.8	20.7 ± 1.8
L3 projection	–	$\mathbf{22.0 \pm 2.3}$	21.3 ± 4.7
L4 projection	–	–	$\mathbf{26.7 \pm 2.9}$
Hierarchical	39.3 ± 3.7	73.3 ± 3.5	63.3 ± 7.7

Table 1: **Changing only the cost on AntMaze.** Success (%, mean ± SE, three seeds). Within each column, the first four rows share level-1 dynamics and planner settings; only the cost space changes. Hierarchical planning is a separate reference.

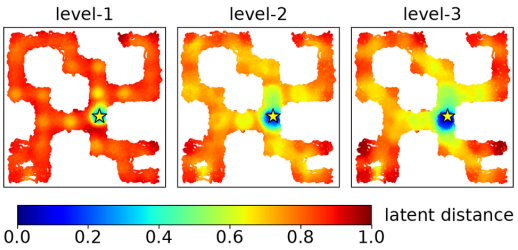

Figure 6: **Planning cost geometry on AntMaze.** Distance to anchor $\star$ in each latent space of one three-level H-JEPA (blue near, red far; one training seed). Higher levels show more graded costs along corridors and a broader low-cost region around the anchor.

HWM achieves higher mean success than LeWM in all four environments, supporting the benefit of temporal decomposition even without a representational hierarchy. H-JEPA provides further gains in some environments, particularly substantial on AntMaze, suggesting that hierarchical representations offer benefits beyond temporal decomposition alone. We next examine the mechanisms through which temporal decomposition and hierarchical representations each improve planning.

4.2 How hierarchy improves planning

Hierarchical planning changes two things at once: it decomposes the search in time, and it scores candidate futures in a more abstract latent space. We isolate the second factor, asking how much an objective in a higher, more abstract level helps on its own, independent of temporal decomposition. We focus on Visual AntMaze, where H-JEPA's substantial gains over HWM suggest benefits from hierarchical representations beyond temporal decomposition alone (Fig. 5, bottom).

Planning costs across the hierarchy. In Visual AntMaze, we visualize the cost geometry in each level's latent space by mapping the latent distance from every state to a fixed anchor (Fig. 6). A navigation environment such as this gives an intuitive reference for a useful cost: latent distance to the anchor should correlate with the length of the shortest traversable path to it, accounting for the maze walls. In the three-level model shown, level-1 distances are nearly constant away from the anchor, providing little signal to guide a gradient planner until it is already close. At higher levels, costs rise smoothly along corridors from the low-cost anchor toward more distant states. These results suggest that training higher levels to predict over longer horizons can both promote abstraction and shape the latent cost geometry to provide a more informative planning signal toward distant goals.

Changing only the cost. To attribute planning gains to the cost geometry, we perform flat planning with each model's level-1 world model, keeping rollout dynamics fixed while varying only the latent space used to compute the goal cost. The native L1 cost scores a rollout by the level-1 latent distance $\|\hat{z}^{(1)} - g^{(1)}\|$ between the prediction and the goal. The *level-k projection* maps both the prediction and the goal through the upper encoders $E^{(2)} \circ \cdots \circ E^{(k)}$ of the same H-JEPA model before measuring the distance; with window $w_\ell{=}1$ these encoders are pointwise, so a single level-1 latent projects to every level. Each column of Table 1 corresponds to a separately trained H-JEPA model with two, three or four levels, evaluated on the tasks of Fig. 5 (bottom). Within each column, the cost-space rows share the same level-1 world model and planner settings, varying only the latent space used to measure the goal cost. Each model's full hierarchical planner serves as a reference.

For every tested multilevel H-JEPA, at least one upper-level cost improves planning success over the native level-1 cost (Table 1). Gains are particularly large when planning in the native space performs poorly. The best projected level-1 planner also exceeds the separately trained LeWM baseline in mean success at every tested depth on AntMaze (LeWM: $18.0 \pm 3.5\%$; Fig. 5, bottom). These gains require only level-1 planning: the upper-level encoders define the cost, but no upper-level solvers are called to generate subgoals, as they are in hierarchical planning. Thus learning a hierarchy of representations can benefit planning simply by providing a more abstract space in which to measure distance to the goal, even before using the hierarchy to decompose the planning problem. We extend this ablation to FourRoom, Push-T and Cube in §H (Table 9). On FourRoom, abstract goal costs

improve flat planning even though H-JEPA and HWM perform similarly. On Push-T and Cube, where we observe no selective abstraction, projected costs provide no clear benefit.

Temporal decomposition. H-JEPA and HWM both decompose long-horizon planning into shorter subgoal problems. On held-out expert trajectories in all four simulated environments, re-anchored level-1 rollouts have more consistently decreasing costs to nearby subgoals than the full-horizon cost to the final goal. This diagnostic supports a benefit from temporal decomposition even with HWM's shared latent space, consistent with its planning gains over LeWM (Fig. 5, bottom). §I provides the trajectory illustration, monotonicity definition, and quantitative comparison; these measurements characterize costs along expert trajectories rather than the success of optimized plans.

4.3 Scaling to diverse scenes: DROID

We now extend H-JEPA to natural scenes that change across episodes. Once the scene varies across episodes, the background becomes the easily predictable part of the signal compared to the robot, so a purely predictive objective is pushed to encode the scene and discard the agent. To address this, we add an Inverse-Dynamics Modeling (IDM) loss term to the objective.

Environments and evaluation. In this section we use DROID [42], a dataset of real teleoperated manipulation videos whose scene, lighting, and manipulated objects all change between episodes. Details about data are in §B. DROID has no simulator, so we score plans offline with the *Fréchet fidelity*, which measures how closely a plan's three-dimensional end-effector path follows the expert's. Let p and $\tilde{p}$ be the cumulative xyz paths of the planned and the expert delta-pose sequences, let d_F be the discrete Fréchet distance in meters, and let $\mathbf{0}$ be the zero-action path:

$$\text{fidelity} = \frac{d_F(\mathbf{0}, \tilde{p}) - d_F(p, \tilde{p})}{d_F(\mathbf{0}, \tilde{p})}. \tag{7}$$

Thus, 0% is satisfied by the arm not moving, 100% is the expert path, and negative values are worse than doing nothing. Scoring the whole path, rather than its endpoint, is why it is relevant for the non-greedy pick-and-place tasks we evaluate. Planners and budgets are in §E, DROID clips and protocol in §J.1, and calibration of this metric against success in simulation, where it ranks graded task progress, in §J.2.

Inverse dynamics loss avoids slow-feature collapse. Against a non-stationary background a purely predictive objective admits a slow-feature solution in which the encoder keeps the scene identity, which is predictable over the horizon, and discards the moving agent, which is only predictable with actions. A simple construction in §L shows that this solution is always feasible when the regularizer sees only the marginal law of the embeddings, a failure mode related to slow-feature analysis [86]. To avoid such collapse, a loss has to act on the joint distribution of representations across time. As in previous work on JEPA world models [76, 72, 71], we therefore add a third term to the level objective of §2.1: an inverse-dynamics model that must regress, from a pair of consecutive state representations, the action relating them. With d_a the action dimension and $\mathrm{sg}[\cdot]$ the stop-gradient, the level-ℓ IDM head $I^{(\ell)}$ gives the loss term

$$\mathcal{L}^{(\ell)}_{\text{idm}} = \frac{1}{T-1} \sum_{t=1}^{T-1} \frac{1}{d_a} \left\| I^{(\ell)}\left(z_t^{(\ell)},\, z_{t+1}^{(\ell)}\right) - \mathrm{sg}\left[a_t^{(\ell)}\right] \right\|_2^2, \tag{8}$$

gets added to the level objective of Eq. (3). The term follows that prior work but, to our knowledge, has not been paired with SIGReg, so we cross-validate both weights jointly (level-1 panel of Fig. 19). At $\ell > 1$ the regressed target is the pooled macro-action $a_t^{(\ell)}$ of Eq. (1), not a primitive action, so it is detached and the gradient reaches the encoder only through the two states.

Hierarchical planning results. Without the loss term of Eq. (8) the model collapses and performance is zero, as in Fig. 7b and Fig. 18. OGBench Cube escapes it because its scene barely varies: latent variance is 75% *between* episodes on DROID but only 28% on Cube (§L.2), so diverse scenes pull the encoder's capacity toward scene identity. Fig. 7b exhibits that adding IDM makes LeWM a strong baseline, and that a second level adds performance on top of it. Both the representational lever and the temporal lever of §4.2 are active on DROID, as shown by the clear advantage of H-JEPA over

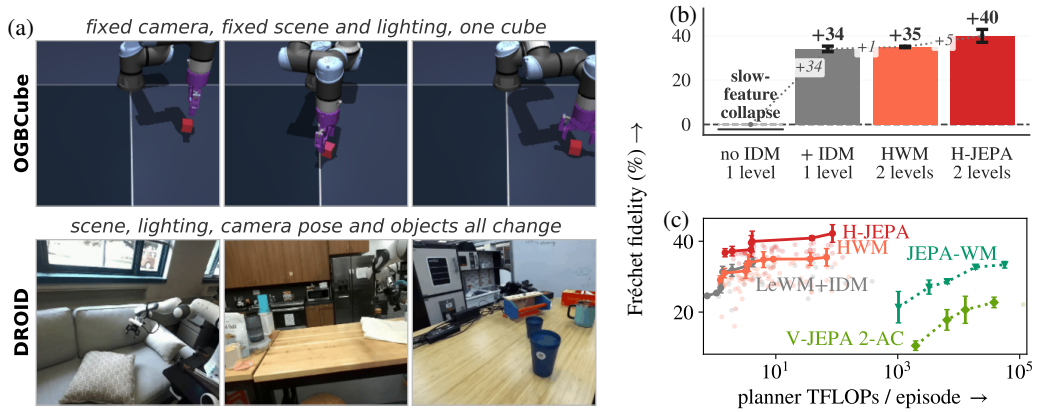

Figure 7: **Scene diversity makes the inverse-dynamics loss necessary, and a second level adds performance at a lower planner budget.** *(a)* First frames of three Cube and three DROID clips, showing how much the scene varies between episodes in each. *(b)* DROID at 5 fps, Fréchet fidelity (%), showing the slow-feature-collapsed LeWM, LeWM+IDM, then two hierarchies: HWM, whose level 2 is an identity encoder, and H-JEPA, whose level 2 is learned. *(c)* The one-level and two-level models of *(b)* swept over planner budgets, Fréchet fidelity against measured planner TFLOPs per episode on a log axis, the bold line being the Pareto front. Two frozen-encoder world models are shown for reference: they are public checkpoints, evaluated at their native 1 fps on the 15 clips whose window fits at that rate, so their intervals span three planner seeds. All other intervals are SE over three train seeds.

HWM. Finally, Fig. 7c shows, like in the single-scene datasets of §4.1, that H-JEPA reaches higher performance at a lower planning budget than the flat model. The frozen-encoder baselines, shown for reference, need orders of magnitude more compute for a lower fidelity.

5 Related Work

Action-conditioned JEPA world models plan in learned latent spaces without pixel reconstruction or reward supervision. DINO-WM [92] learns dynamics over frozen pretrained features, V-JEPA 2 [2] adds action-conditioned prediction after video pretraining, and PLDM [72] and LeWM [52] train the encoder and predictor jointly from pixels. H-JEPA extends this end-to-end setting to a hierarchy of latent spaces and prediction timescales, using LeJEPA's SIGReg regularizer [6] at every level.

Hierarchical video models learn representations at several timescales but reconstruct observations and do not plan [67, 44, 54], while pre-existing hierarchical model-based control methods operate in the reward-driven, task-specific setting [32, 36, 27]. Closest to our setting, HWM [91] trains latent world models at multiple temporal horizons from reward-free offline data and plans hierarchically, but within a shared latent space. H-JEPA additionally learns a distinct representation at each level, so that higher-level objectives can discard detail that lower-level dynamics retain. §M gives the extended survey and Table 13 compares hierarchical control methods.

6 Conclusion

We presented H-JEPA, an end-to-end method for training a hierarchy of action-conditioned JEPAs, each predicting in its own latent space over progressively longer horizons. Higher levels discard fast features that are unpredictable over their horizon and retain slow, predictable ones. Hierarchical planning with H-JEPA outperforms flat planning at lower planner compute by scoring goals in a more abstract latent space and decomposing long tasks into shorter subgoal problems. An inverse-dynamics term extends H-JEPA to the diverse scenes of real robot video in DROID. These DROID results are offline and open-loop; closed-loop evaluation on a real robot remains to be done.

Open directions include learning hierarchies from action-free video and other high-dimensional temporal streams; aligning upper-level latents with language so that goals can be specified in words rather than as target observations; and exploring whether variable-duration segments in place of fixed strides for training upper level world models can yield better representations or planning performance.

AI USE STATEMENT

We used generative AI tools, principally large language models accessed through an agentic coding assistant. Among the uses ICLR requires authors to disclose, we used them to implement methods, i.e. the training, planning and evaluation code; to clean and reformat datasets, including the pipeline that re-encodes the DROID corpus; to support qualitative analysis of data; to provide ingredients of proofs and assist in writing them; to propose and refine hypotheses; and to give feedback on our experiments. We did not use them to generate synthetic datasets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to design our methodology or experiments, or to interpret results, and translation does not apply to this work. Among the recommended disclosures, we also used them to create and edit software code, including the scripts that submit cluster jobs; to create and modify figures; to suggest experimental parameters; to brainstorm; to search for and summarize existing literature; and to draft parts of this paper and edit it for readability.

We did not use generative AI to produce experimental results or to fabricate or impute any reported data point. We have reviewed all AI-assisted work: the authors read and made sure the code was correct, every figure and table in this paper is produced by a generator that parses real run logs and gates its values against pinned expected numbers, and the authors checked every proof. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work trains and evaluates world models on public robot-manipulation datasets and simulated environments. It involves no human subjects, no personally identifying data and no user study. The real-robot corpus, DROID [42], is a public teleoperation dataset released by its authors under CC-BY 4.0, which permits redistributing an adapted version given attribution and a statement of the changes made; we re-encode it for training and release that derivative on those terms, and §J.4 describes the pipeline. All planning results are obtained in simulation or offline against held-out recorded episodes, and no policy from this work was run on physical hardware, so the work carries no direct risk of physical harm.

REPRODUCIBILITY STATEMENT

Environments, datasets and splits are described in §B; the architecture, training objective and optimization settings in §C, with the exact hyperparameters of every headline run in §C.2. Probing and decoding protocols are in §D. Planning evaluation is specified in §E for the simulated environments and in §J.1 for DROID, including the planner configuration pinned to each reported number, and §J.2 defines the Fréchet fidelity metric and states what it does and does not measure. The DROID re-encoding pipeline is described in §J.4. Unless stated otherwise, reported values are means over three training seeds; results that rest on a single seed are marked as such in the text.

REFERENCES

[1] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. arXiv:2301.08243.

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

[3] Christopher Baldassano, Janice Chen, Asieh Zadbood, Jonathan W. Pillow, Uri Hasson, and Kenneth A. Norman. Discovering event structure in continuous narrative perception and memory. *Neuron*, 95(3):709–721, 2017.

[4] Federico Baldassarre, Marc Szafraniec, Basile Terver, Vasil Khalidov, Francisco Massa, Yann LeCun, Patrick Labatut, Maximilian Seitzer, and Piotr Bojanowski. Back to the features: Dino as a foundation for video world models, 2025. URL https://arxiv.org/abs/2507.19468.

[5] Randall Balestriero and Yann LeCun. Learning by reconstruction produces uninformative features for perception, 2024. URL https://arxiv.org/abs/2402.11337.

[6] Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. https://arxiv.org/abs/2511.08544, 2025.

[7] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15791–15801, June 2025.

[8] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2105.04906.

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

[10] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. OpenAI technical report, https://openai.com/index/video-generation-models-as-world-simulators/, 2024.

[11] Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.15391.

[12] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. arXiv:2104.14294.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020. arXiv:2002.05709.

[14] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. arXiv:2011.10566.

[15] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023. arXiv:2303.04137.

[16] Andy Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3):181–204, 2013.

[17] Harald Cramér and Herman Wold. Some theorems on distribution functions. *Journal of the London Mathematical Society*, 1(4):290–294, 1936.

[18] Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, 1992.

[19] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2302.00111.

[20] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[21] Yonathan Efroni, Dipendra Kumar Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Provable rl with exogenous distractors via multistep inverse dynamics. *ArXiv*, abs/2110.08847, 2021. URL `https://api.semanticscholar.org/CorpusID:239016034`.

[22] T. W. Epps and Lawrence B. Pulley. A test for normality based on the empirical characteristic function. *Biometrika*, 70(3):723–726, 1983.

[23] Kuan Fang, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Dynamics learning with cascaded variational inference for multi-step manipulation. In *Conference on Robot Learning (CoRL)*, 2019. arXiv:1910.13395.

[24] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793, 2017. arXiv:1610.00696.

[25] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann LeCun. Rankme: assessing the downstream performance of pretrained self-supervised representations by their rank. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[26] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2006.07733.

[27] Christian Gumbsch, Noor Sajid, Georg Martius, and Martin V. Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *International Conference on Learning Representations (ICLR)*, 2024.

[28] Nico Gürtler and Georg Martius. Long-horizon planning with predictable skills. In *Reinforcement Learning Conference (RLC)*, 2025.

[29] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. arXiv:1803.10122 ("World Models").

[30] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, 2019. arXiv:1811.04551.

[31] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1912.01603.

[32] Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2206.04114.

[33] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023. Published in Nature 640:647–653, 2025.

[34] Nicklas Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022. arXiv:2203.04955.

[35] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.16828.

[36] Nicklas Hansen, Jyothir S V, Vlad Sobal, Yann LeCun, Xiaolong Wang, and Hao Su. Hierarchical world models as visual whole-body humanoid controllers. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2405.18418.

[37] Uri Hasson, Eunice Yang, Ignacio Vallines, David J. Heeger, and Nava Rubin. A hierarchy of temporal receptive windows in human cortex. *Journal of Neuroscience*, 28(10):2539–2550, 2008.

[38] Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling. *arXiv preprint arXiv:2507.07955*, 2025.

[39] Tomoshi Iiyama, Masahiro Suzuki, and Yutaka Matsuo. SUNTA: Hierarchical video prediction with surprise-based chunking. *arXiv preprint arXiv:2607.02087*, 2026.

[40] Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor world models: Perception for action via inverse dynamics, 2026. URL https://arxiv.org/abs/2606.20104.

[41] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022. arXiv:2205.09991.

[42] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024.

[43] Stefan J. Kiebel, Jean Daunizeau, and Karl J. Friston. A hierarchy of time-scales and the brain. *PLoS Computational Biology*, 4(11):e1000209, 2008.

[44] Taesup Kim, Sungjin Ahn, and Yoshua Bengio. Variational temporal abstraction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. arXiv:1910.00775.

[45] Markus Kögel, Mohamed Ibrahim, Christian Kallies, and Rolf Findeisen. Safe hierarchical model predictive control and planning for autonomous systems. *International Journal of Robust and Nonlinear Control*, 2023. arXiv:2203.14269.

[46] Yilun Kuang, Yash Dagade, Quentin Le Lidec, Lucas Maes, Randall Balestriero, and Yann LeCun. LpWM: A case for sparse representations in world models. *arXiv preprint arXiv:2608.22764*, 2026.

[47] Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Rajiv Didolkar, Dipendra Misra, Dylan J Foster, Lekan P Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=TNocbXm5MZ.

[48] Yann LeCun. A path towards autonomous machine intelligence. *OpenReview*, 2022. Version 0.9.2, https://openreview.net/forum?id=BZ5a1r-kVsf.

[49] Alexander Levine, Peter Stone, and Amy Zhang. Multistep inverse is not all you need. *Reinforcement Learning Journal*, 2:884–925, 2024.

[50] Tianyu Li, Roberto Calandra, Deepak Pathak, Yuandong Tian, Franziska Meier, and Akshara Rai. Planning in learned latent action spaces for generalizable legged locomotion. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):2682–2689, 2021. arXiv:2008.11867.

[51] Etai Littwin, Omid Saremi, Madhu Advani, Vimal Thilak, Preetum Nakkiran, Chen Huang, and Joshua M. Susskind. How JEPA avoids noisy features: The implicit bias of deep linear self distillation networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ez7w0Ss4g9.

[52] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. https://arxiv.org/abs/2603.19312, 2026.

[53] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2209.00588.

[54] Ramy Mounir, Sujal Vijayaraghavan, and Sudeep Sarkar. STREAMER: Streaming representation learning and event segmentation in a hierarchical manner. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[55] John D. Murray, Alberto Bernacchia, David J. Freedman, Ranulfo Romo, Jonathan D. Wallis, Xinying Cai, Camillo Padoa-Schioppa, Tatiana Pasternak, Hyojung Seo, Daeyeol Lee, and Xiao-Jing Wang. A hierarchy of intrinsic timescales across primate cortex. *Nature Neuroscience*, 17(12):1661–1663, 2014.

[56] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. arXiv:1805.08296.

[57] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024. arXiv:2406.02523.

[58] Nilaksh, Saurav Jha, Artem Zholus, and Sarath Chandar. Reconstruction or semantics? what makes a latent space useful for robotic world models, 2026. URL https://arxiv.org/abs/2605.06388.

[59] NVIDIA. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[60] NVIDIA, :, Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning, 2025. URL https://arxiv.org/abs/2511.04831.

[61] Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. Byte latent transformer: Patches scale better than tokens. *arXiv preprint arXiv:2412.09871*, 2024.

[62] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.20092.

[63] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2778–2787. PMLR, 2017. arXiv:1705.05363.

[64] Karl Pertsch, Oleh Rybkin, Frederik Ebert, Shenghao Zhou, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2006.13205.

[65] Jean Ponce, Basile Terver, Martial Hebert, and Michael Arbel. Dual perspectives on non-contrastive self-supervised learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=f5MC1G6XhB.

[66] Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.

[67] Vaibhav Saxena, Jimmy Ba, and Danijar Hafner. Clockwork variational autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2102.09532.

[68] Robin Schiewer, Anand Subramoney, and Laurenz Wiskott. Exploring the limits of hierarchical world models in reinforcement learning. *Scientific Reports*, 14, 2024. arXiv:2406.00483; DOI 10.1038/s41598-024-76719-w.

[69] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. arXiv:1911.08265.

[70] Chang Shi, Tim Pearce, Manan Tomar, Siddhartha Sen, and John Langford. Hierarchical latent prediction for language models. *arXiv preprint arXiv:2608.05806*, 2026.

[71] Vlad Sobal, Jyothir S V, Siddhartha Jalagam, Nicolas Carion, Kyunghyun Cho, and Yann LeCun. Joint embedding predictive architectures focus on slow features. `https://arxiv.org/abs/2211.10831`, 2022.

[72] Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *arXiv preprint arXiv:2502.14819*, 2025.

[73] Richard S. Sutton and Andrew G. Barto. An adaptive network that constructs and uses an internal model of its world. *Cognition and Brain Theory*, pp. 217–246, 1981.

[74] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2): 181–211, 1999.

[75] Yunhao Tang, Zhaohan Daniel Guo, Pierre Harvey Richemond, Bernardo Ávila Pires, Yash Chandak, Rémi Munos, Mark Rowland, Mohammad Gheshlaghi Azar, Charline Le Lan, Clare Lyle, András György, Shantanu Thakoor, Will Dabney, Bilal Piot, Daniele Calandriello, and Michal Valko. Understanding self-predictive learning for reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023. arXiv:2212.03319.

[76] Basile Terver, Randall Balestriero, Megi Dervishi, David Fan, Quentin Garrido, Tushar Nagarajan, Koustuv Sinha, Wancong Zhang, Mike Rabbat, Yann LeCun, and Amir Bar. A lightweight library for energy-based joint-embedding predictive architectures. `https://arxiv.org/abs/2602.03604`, 2026.

[77] Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Transactions on Machine Learning Research (TMLR)*, 2026. arXiv:2512.24497.

[78] Edward C. Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948.

[79] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017. arXiv:1703.01161.

[80] Tongzhou Wang, Simon Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22591–22612. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/wang22c.html`.

[81] Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. Adajepa: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026.

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

[83] Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. arXiv:1506.07365.

[84] Björn Weghenkel, Asja Fischer, and Laurenz Wiskott. Graph-based predictable feature analysis. *Machine Learning*, 106:1359 – 1380, 2016. URL https://api.semanticscholar.org/CorpusID:11180992.

[85] Björn Weghenkel and Laurenz Wiskott. Slowness as a proxy for temporal predictability: An empirical comparison. *Neural Computation*, 30(5):1151–1179, 05 2018. ISSN 0899-7667. doi: 10.1162/neco_a_01070. URL https://doi.org/10.1162/neco_a_01070.

[86] Laurenz Wiskott and Terrence J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

[87] Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. iVideoGPT: Interactive videogpts are scalable world models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2405.15223.

[88] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning (CoRL)*, 2022. arXiv:2206.14176.

[89] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.06114.

[90] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning (ICML)*, 2021. arXiv:2103.03230.

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

[92] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2411.04983.

Appendix

A Notation **18**

B Environments and datasets **19**

C Architecture and training **19**

D Probing and decoding **22**

E Planning evaluation **23**

F Temporal-frequency separation and selective abstraction **26**

G Data specialization across levels **27**

H Planning-cost ablations across environments **28**

I Prediction-cost monotonicity **28**

J DROID: corpus, planning evaluation and offline metric **30**

K Anti-collapse: SIGReg, inverse dynamics and detachment **34**

L Slow-feature collapse in joint-embedding predictive objectives **36**

M Extended related work **39**

A NOTATION

Table 2: **Notation reference.** Symbols are introduced in §2 unless the row names another section. Level superscripts are always parenthesized, (ℓ); a bare subscript is a time index and $a{:}b$ is a half-open range. Appendix-local symbols are defined where they are used.

Symbol	Description
Levels and indices	
ℓ, L	Hierarchy level index, $\ell = 1, \dots, L$; level 1 is the finest, $L = 2$ throughout
t	Discrete time index, at the level's own rate
$a{:}b$	Half-open slice: the $b-a$ indices $a, \dots, b-1$
M	Number of random directions drawn by SIGReg (§K)
Observations, latents and actions	
o_t	Image observation at time t
a_t	Primitive action block for the transition $o_t \to o_{t+1}$
$z_t^{(\ell)}$	Level-ℓ latent state; $z_t^{(1)} = E^{(1)}(o_t)$
$\hat{z}^{(\ell)}_{1:H_\ell}$	Action-conditioned autoregressive rollout at level ℓ
$\hat{z}^{(\ell+1),*}$	Optimized rollout produced by the level above; $*$ marks an optimized quantity
$a_t^{(\ell)}$	Level-ℓ action embedding (a macro-action for $\ell > 1$)
$a^{(\ell),*}_{0:H_\ell-1}$	Optimized action sequence returned by the level-ℓ planner
$g^{(\ell)}$	Goal state at level ℓ, obtained by encoding the goal observation
$Z^{(\ell)}$	Minibatch of level-ℓ embeddings, $Z^{(\ell)} \in \mathbb{R}^{n \times D}$: n rows of width D
N, T	Sequences and timesteps of a minibatch of embeddings
D, W	Embedding width, and the number of data-parallel ranks (§K.3)
Learned components	
$E^{(\ell)}$	State encoder at level ℓ; pools a w_ℓ-step lower-level window
$A^{(\ell)}$	Action encoder at level ℓ; pools an s_ℓ-step lower-level window for $\ell > 1$
$F^{(\ell)}$, $I^{(\ell)}$	Latent predictor and inverse-dynamics head at level ℓ
$\mathrm{Roll}^{(\ell)}$	Autoregressive rollout operator induced by $F^{(\ell)}$
Training objective	
$\mathcal{L}^{(\ell)}$, $\mathcal{L}^{(\ell)}_{\mathrm{pred}}$, $\mathcal{L}^{(\ell)}_{\mathrm{idm}}$	Total level-ℓ objective and its prediction and inverse-dynamics terms
$\mathrm{SIGReg}(Z^{(\ell)})$	Sketched normality (anti-collapse) regularizer
u_m, h_m	Unit direction $u_m \in \mathbb{S}^{D-1}$ and its projection $h_m = Z^{(\ell)} u_m$ (§K)
$\mathcal{T}_{\mathrm{EP}}$	Epps–Pulley normality statistic (§K)
Planning	
H_ℓ	Planning horizon at level ℓ, in level-ℓ steps
$C^{(\ell)}(z, z')$	Pairwise cost paid by the level-ℓ planner; squared Euclidean distance $\|z - z'\|_2^2$ within one latent space
$\tilde{z}_i^{(\ell+1)}$	Level-ℓ rollout lifted into level-$(\ell{+}1)$ space by $E^{(\ell+1)}$
$K_{\ell+1}$	Number of subgoal terms in the level-ℓ cost; pinned by the lift, $K_{\ell+1} = \lfloor H_\ell / s_{\ell+1} \rfloor \le H_{\ell+1}$
$g_i^{(\ell+1)}$	The i-th subgoal handed to level ℓ; the last is anchored, $g^{(\ell+1)}_{K_{\ell+1}} = g^{(\ell+1)}$
β	Weight of one subgoal term relative to the terminal term, Eq. (6)
t_j, ξ_j	Replanning times, and primitive commands committed at phase j (the commit schedule, §E)
S, q, R	Planner candidates per level, scored final steps of a rollout, and level-1 blocks executed per replan (§E)
Architecture and loss hyperparameters	
s_ℓ, w_ℓ	Temporal stride and window size from level $\ell-1$ to level ℓ: each level-ℓ state spans w_ℓ lower-level states, taken every s_ℓ
c_ℓ	Context length, in level-ℓ latent states
λ_ℓ, γ_ℓ	SIGReg and inverse-dynamics weights at level ℓ

B Environments and datasets

We evaluate on two navigation environments, FourRoomDistractors and Visual AntMaze, two simulated manipulation environments, Push-T and OGBench Cube, and one realistic manipulation setting, the real-robot DROID corpus. Push-T and Cube follow the LeWM setup [52] in environment and dataset; FourRoomDistractors, Visual AntMaze and their datasets are ours; DROID is a public release. Observations are RGB, 224×224 except Visual AntMaze at 64×64 and DROID at 256×256; actions are continuous.

a) **FourRoomDistractors** is a continuous 2D navigation environment derived from the two-room environment of Sobal et al. [72]. The world is divided into four equal rooms by two walls, each with one door; every door is closed independently with probability 0.5 at reset, so the route between two rooms changes across episodes. The ego agent is a dot moving 2.5 px per step under a 2D velocity command. A single distractor dot of the same speed moves under a smooth random policy that ignores the ego, and teleports to a random free location at Poisson-distributed intervals of mean 35 steps; it neither collides with the ego nor affects the task. We collect $3{,}750$ episodes of 400 steps (1.5M transitions) with a per-episode mixture of two ego policies: a heavily noised shortest-path expert that picks a new goal each time it reaches the current one, and a smooth random-exploration policy. Level 1 is trained on 1.0M of these transitions (Table 4), and 150 disjoint episodes serve as validation data.

b) **Visual AntMaze** is the OGBench `visual-antmaze-medium` environment [62]: an 8-DoF quadruped navigating a maze from a third-person camera over a color-coded floor, so that the agent's location can be read from the image, with an 8D torque command in $[-1, 1]$. The world model additionally receives the 27-dimensional proprioceptive state (joint positions without the global xy, and joint velocities). We collect data with the OGBench protocol, a SAC low-level locomotion expert driven by a commanded direction, in two of its variants. `explore` resamples a random direction every 10 steps under heavy action noise, giving low-quality but high-coverage trajectories: $12{,}500$ episodes of 400 steps (5.0M transitions). `stitch` drives the lightly noised expert along the maze's shortest path toward a goal four cells away, giving short goal-directed segments that must be stitched to solve longer tasks: $31{,}250$ episodes of 400 steps (12.6M transitions), of which 5.5M are used for training level-2. Validation uses $1{,}250$ held-out `stitch` episodes.

c) **Push-T** is the continuous 2D manipulation task of Chi et al. [15] in which a circular agent pushes a T-shaped block to a target configuration; the 2D command is the agent's target position. We use the expert dataset of DINO-WM [92] as released with LeWM: $18{,}685$ training episodes (2.34M frames, 125 steps on average) and 21 validation episodes.

d) **OGBench Cube** is the single-cube variant of the OGBench `cube` manipulation environment [62], in which a robot arm must pick up a cube and place it at a target location; the 5D command sets the end-effector displacement, yaw, and gripper. As in LeWM, the data are $10{,}000$ episodes of 200 steps from the benchmark's scripted data-collection policy, split into $9{,}800$ training and 200 validation episodes.

e) **DROID** [42] is a corpus of real teleoperated manipulation video with no simulator, in which scene, lighting, camera calibration and manipulated objects all change between episodes. The command is the 7-dimensional end-effector command: Cartesian xyz, roll-pitch-yaw and gripper. Actions are normalized per coordinate against dataset statistics before training, so the gripper dimension, one to two orders of magnitude larger than the Cartesian deltas, does not dominate the mean in Eq. (8). We train on a re-encoded copy, $74{,}530$ episodes and $220{,}084$ views with all three camera views at 256×256; §J.4 describes that build and what it costs. Because there is no simulator, plans are scored offline against logged expert trajectories on a frozen 16-clip evaluation set, specified in §J.1.

C Architecture and training

This appendix collects the architecture and training settings of every model in the paper: the LeWM baseline and the two-, three-, and four-level H-JEPA models of the simulated environments (§C.1), then the DROID models (§C.2). Probing and decoding are described in §D, and planning evaluation in §E.

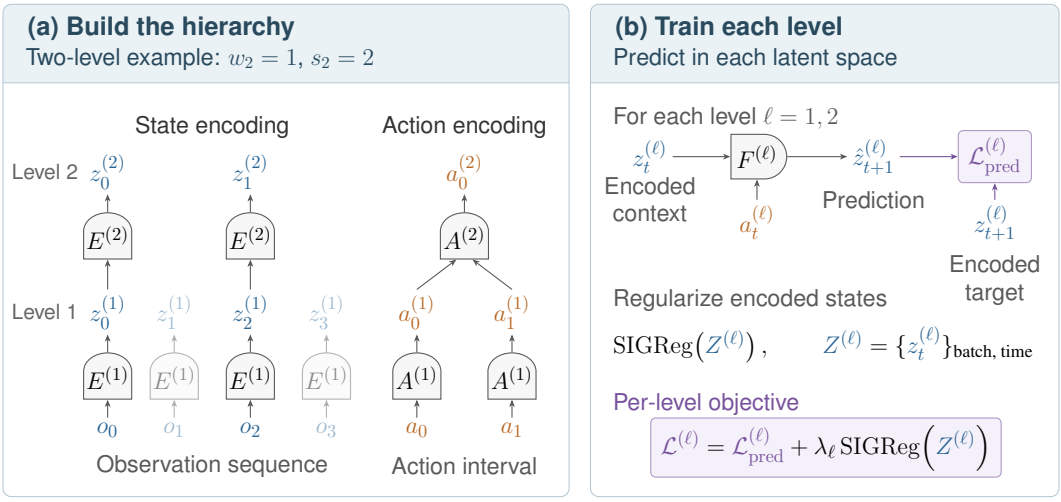

Figure 8: **H-JEPA training across temporal scales.** (a) With w_2=1 and s_2=2, level 2 encodes every other level-1 state pointwise and aggregates the two intervening action embeddings. Faded columns are retained at level 1 but skipped at level 2. All experiments use pointwise state encoding. (b) Each level predicts the next latent state conditioned on encoded actions, with SIGReg regularizing the encoded states against collapse.

Fig. 8 illustrates the hierarchy construction and per-level training objective.

C.1 Single-scene environments

These are the models of Fig. 5 (bottom). LeWM is a single-level JEPA using the architecture and objective of Maes et al. [52]. Each H-JEPA model trains all levels jointly from scratch in one end-to-end run, with upper-level losses backpropagating through the encoders below. We describe shared settings once and list environment-specific choices in Table 4.

Data. Within each environment, all four models use the same training data: 1.0M transitions from the one-distractor FourRoom dataset; a 10.05M-transition, equal mixture of AntMaze `explore` and `stitch`; the Push-T expert set; or the Cube scripted set (§B). Frames use ImageNet normalization without augmentation. A level-1 step spans 5 environment steps, with the intervening actions concatenated. Every upper level has stride $s_\ell = 2$ and window $w_\ell = 1$. Each sample is one clip long enough to supply $T = 4$ states at the top level; each lower level takes a random four-state crop of its own stream. The causal predictor receives three states and learns their teacher-forced one-step successors. If n_ℓ is the number of level-ℓ frames needed in the clip, $n_L = 4$ and $n_{\ell-1} = (n_\ell - 1)s_\ell + w_\ell$, giving the minimum clip lengths in Table 3. Batches contain 128 clips. A clip must lie within one episode, so the number of clips an epoch yields decreases with the minimum clip length (Table 3): little on FourRoom and AntMaze, whose episodes are long, and sharply on Push-T, whose episodes are short relative to the three- and four-level minimum clip lengths. H-JEPA training is epoch-matched across depths, so a deeper model also takes correspondingly fewer gradient steps.

Training dynamics. Fig. 9 tracks the unweighted prediction and SIGReg terms of every level of the four-level models over training. Neither term degenerates at any level: no prediction loss falls toward zero and every SIGReg statistic settles at a finite value, so joint end-to-end training keeps all levels non-collapsed at the deepest setting we train. Both terms are ordered by level in every environment, each higher level converging to a larger prediction loss and a larger SIGReg value, consistent with the fewer clips and coarser targets an upper level sees (Table 3). Upper levels also start improving later than level 1, most visibly on AntMaze, where the abstract predictors hold a plateau over the first epochs before descending. On Push-T and Cube the level-3 and level-4 prediction curves are the noisiest and flatten earliest, in line with the short-episode clip deficit of Table 3.

Architecture. The level-1 encoder is a ViT-Tiny whose `[CLS]` token passes through a one-hidden-layer MLP projector with Batch Normalization. On AntMaze, its 256-dimensional pixel projection

Table 3: **Temporal layout of one training sample and the clips it leaves per epoch.** The minimum clip length is the shortest episode segment that yields one training sample, in environment steps, including both endpoint observations. All H-JEPA models use stride two and window one at every level above the first. A clip must lie within one episode (the loader reserves $5\,n_1$ environment steps per clip), so the number of clips an epoch yields falls as the minimum clip length grows; parentheses give each count as a percentage of the LeWM count in the same environment. Each clip supplies three teacher-forced transitions per level.

	LeWM	Two-level H-JEPA	Three-level H-JEPA	Four-level H-JEPA
Level-1 frames per clip	4	7	13	25
Minimum clip length (env. steps)	16	31	61	121
Training clips per epoch				
FourRoom	952,708	915,298 (96%)	840,478 (88%)	690,838 (73%)
AntMaze	9,575,000	9,200,000 (96%)	8,450,000 (88%)	6,950,000 (73%)
Push-T	1,981,721	1,701,446 (86%)	1,143,623 (58%)	285,931 (14%)
Cube	1,783,600	1,636,600 (92%)	1,342,600 (75%)	754,600 (42%)

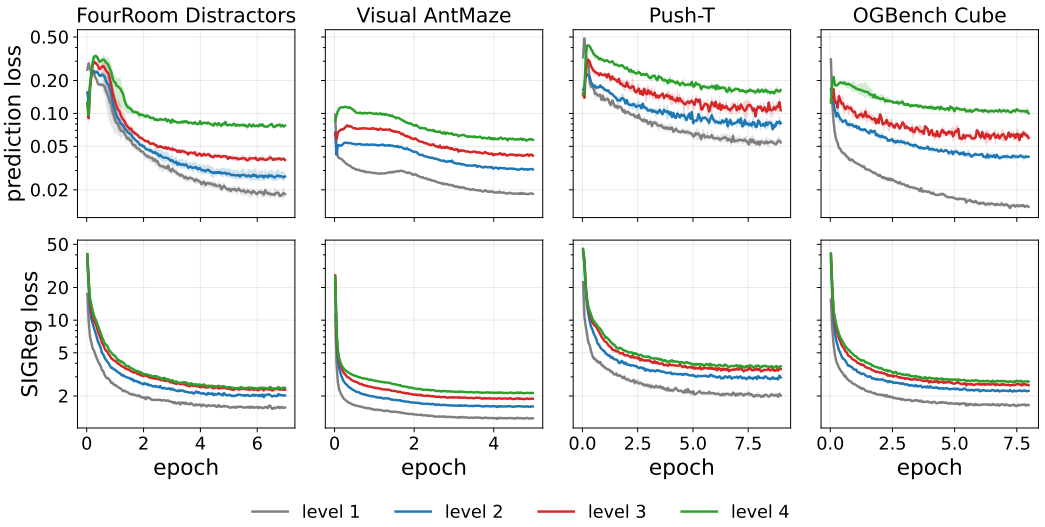

Figure 9: **Per-level training losses of the four-level H-JEPA models.** Unweighted prediction loss (top) and SIGReg statistic (bottom) of each level over training, mean over three seeds with a min/max band, on a log scale. Every level of a model uses the same SIGReg weight (0.18 on FourRoom, 0.09 elsewhere), so the weighted curves differ from these by a constant factor per environment. These are the statistics of the small-environment depth-family models and are not on the scale of the DROID values quoted in §L.

is concatenated with a 128-dimensional embedding of the proprioceptive state from a two-layer MLP. The level-1 predictor is a causal transformer with 4 layers, 8 heads, width 192, feed-forward width 1024, action conditioning through adaptive LayerNorm, and an MLP projector of the same form as the encoder's. Each upper level uses a two-layer MLP encoder with hidden width 384, applied to the lower-level latent; a one-layer transformer pools the two intervening lower-level action embeddings into a 4-dimensional macro-action. Its causal predictor has the same structure as level 1, with dimensions given in Table 4. Within an environment, latent dimensions and upper-level architectures are shared across all H-JEPA depths.

Training. We minimize the sum of $\mathcal{L}_{\text{pred}} + \lambda_\ell$ SIGReg over levels (§2.1), with $M = 1024$ random projections and no stop-gradient, moving-average targets, or pretrained weights. All models use AdamW with weight decay 10^{-3}, linear warm-up followed by cosine decay, gradient clipping at 1.0, and `bfloat16`. Peak learning rates are 5×10^{-5} for level 1 and 10^{-5} for every upper level. The SIGReg coefficient is uniform across levels within each H-JEPA model. Training uses three seeds (42, 43, 44). The two-, three-, and four-level H-JEPA runs share an epoch budget within each environment; longer clips change the number of valid samples, so optimizer-step counts and training FLOPs are not matched across depths.

Table 4: **Per-environment architecture and training settings.** Upper-level predictor dimensions and H-JEPA epoch counts apply to every depth. H-JEPA models are initialized independently of the LeWM baselines.

	FourRoom	AntMaze	Push-T	Cube
Frame size / patch	224 / 14	64 / 4	224 / 14	224 / 14
Command / proprio dim.	2 / n/a	8 / 27	2 / n/a	5 / n/a
Latent dim. d	192	384	192	192
Upper predictor layers / heads	4 / 8	5 / 12	6 / 16	6 / 16
Upper predictor width / FF width	192 / 1024	224 / 1536	192 / 2048	192 / 2048
SIGReg λ, LeWM / H-JEPA	0.72 / 0.18	0.09 / 0.09	0.09 / 0.09	0.09 / 0.09
H-JEPA epochs	7	5	9	8

Implementation and hardware. We use the `stable-worldmodel` and `stable-pretraining` libraries used by LeWM [52], and train on H200 GPUs.

C.2 DROID

To avoid slow-feature collapse at level $\ell > 1$, where the macro-actions a level consumes are embeddings produced by $A^{(\ell)}$ rather than raw actions, we apply SIGReg to the action embeddings as well, with weight 0.02; level 1 carries no such term. Our DROID models see pixels and actions only, no proprioception.

The data pipeline and the learning rates also differ from those of the single-scene environments (§C.1), which balance the levels: there, each lower level takes a random four-state crop of its own stream, so every level learns from three teacher-forced transitions per clip, and every upper level trains at a fifth of the level-1 learning rate. On DROID, a training clip supplies eight level-1 states, and level 2 reads three of them, subsampled at stride 3 from the same clip, with no crop of its own. Both levels train with one-step teacher forcing, as in the single-scene environments, so level 2 receives two prediction targets per clip against seven at level 1, and both levels share the peak learning rate of Table 5.

Table 5 gives the training settings of the headline two-level run. The architecture and loss settings describe the model rather than how it is scheduled, so they apply equally to the end-to-end and stagewise procedures (§2); the optimization settings are those of the end-to-end headline runs. Level 1 consumes raw actions, which is why the action regularizer appears only in the level-2 block.

Level 1 is common with the one-level LeWM+IDM model: the level-1 and optimization blocks of Table 5 describe it too. Frames are center-cropped to a square, resized to 256×256 and normalized with ImageNet statistics, with no augmentation beyond the random choice of clip. Level 2 has stride $s_2 = 3$ and window $w_2 = 1$: its encoder is applied to every third level-1 state on its own. Its macro-action encoder concatenates the raw 7-dimensional commands of the three level-1 steps it spans and maps them to an 8-dimensional macro-action through an MLP with one hidden layer of width 64 and a final LayerNorm; level 1 takes its commands directly. At both levels the predictor is a causal transformer with 6 heads of width 64, feed-forward width 1536 and dropout 0.1, conditioned on actions through adaptive LayerNorm. Every SIGReg term, the action one included, uses $M = 1024$ random projections.

D PROBING AND DECODING

All probe and decoder results are obtained *post hoc*: after training, the world model is frozen and every frame of a held-out probing set is encoded through the hierarchy without striding, giving one detached latent per level for every frame. Probers and decoders are then fitted on these latents and measure what each latent encodes.

Probers. Each probed entity has a two-layer MLP (hidden width 512, LayerNorm, GELU) per level, mapping the latent to the z-scored entity vector under a mean-squared-error loss. We report the variance-normalized error (NMSE) on the evaluation split: the per-dimension de-normalized squared error divided by the entity's variance and averaged equally over dimensions (§3).

Table 5: **Training hyperparameters on DROID.** Predictors are causal transformers, given as width/depth. Encoder output grids are per frame, one pooled vector here. The grouping row gives the shape of the tensor SIGReg tests: (T, N, D) tests normality within each timestep, over the N sequences of the batch. The architecture and loss settings do not depend on the training schedule, whether end-to-end or stagewise (§2); the optimization block does, and reports the end-to-end headline runs. The peak learning rate is the same at both levels, AdamW runs at $\beta = (0.9, 0.999)$, and the number of iterations per epoch is capped by the length of the dataloader, which is what gives 292 at this batch size. The rollout row gives the number of predicted steps the loss is unrolled over at each level, as in prior work [77].

	DROID
Global	
Stop-gradient on latent targets	None
Level-1 to level-2 stride	3
Rollout steps L1 / L2	(2,1)
Level 1	
Encoder	ViT-S/16, pooled
Encoder output grid	$1\times1\times384$
Predictor	384/12
SIGReg grouping	(T, N, D)
SIGReg weight λ_1	0.08
IDM weight γ_1	100
Level 2	
Encoder	latent MLP 768
Encoder output grid	$1\times1\times384$
Predictor	384/12
SIGReg grouping	(T, N, D)
SIGReg weight λ_2	0.16
IDM weight γ_2	50
Action SIGReg weight	0.02
Optimization	
Optimizer	AdamW
Total batch size	256
Peak learning rate	5×10^{-4}
Warm-up, fraction of steps	0.1, linear
Decay	cosine to 10^{-5}
Weight decay	10^{-4}
Gradient clipping, per level	1.0
Precision	`bfloat16`
Epochs	100
Iterations per epoch	292
Training seeds	1, 1000, 10000

Decoder. Following LeWM, a lightweight transformer decoder reconstructs the frame from a single latent, for visualization only. The latent is projected to width 256 and serves as key and value for a set of learnable query tokens, one per encoder patch (16×16 queries at both resolutions); four cross-attention blocks with 8 heads and residual MLPs update the queries, and a linear layer maps each to its pixel patch. The loss is the mean-squared error to the ImageNet-normalized frame.

Training. Probers and decoders are trained jointly with AdamW (weight decay 10^{-4}), batch size 128, for 25 epochs (60 on FourRoomDistractors); probers use learning rate 10^{-3} and decoders 10^{-4} (10^{-3} on Visual AntMaze). Table 6 lists the probing data and the probed entities per environment. Entities are grouped by their temporal frequency into the slow entity that level 2 retains and the fast entity it discards (§3); Push-T and Cube form a single frequency band, so all of their entities are retained and none is marked fast.

E Planning evaluation

Evaluation tasks. FourRoomDistractors and Push-T use pairs of held-out expert frames 75 environment steps apart; the FourRoom start and goal lie two rooms apart. Cube uses held-out 20-step

Table 6: **Probing data and probed entities per environment.** Transition counts are for fitting (train) and scoring (eval). Visual AntMaze is scored on two evaluation sets drawn from the `explore` and `stitch` distributions, with metrics averaged over both. On FourRoomDistractors and Visual AntMaze, whose state separates into slow and fast entities (Fig. 12), the entities are labeled accordingly. Entity dimensions in parentheses.

Environment	transitions (train / eval)	Probed entities
FourRoomDistractors	160k / 40k	**slow:** ego xy (2) **fast:** distractor xy (2)
Visual AntMaze	500k / 2×25k	**slow:** global xy (2) **fast:** body state (27)
Push-T	250k / 2.5k	agent xy (2), block xy (2), block orientation (2), agent velocity (2)
OGBench Cube	250k / 40k	cube position (3), cube orientation (6), effector position (3), effector yaw (2), gripper (2), arm joint positions (6), arm joint velocities (6)

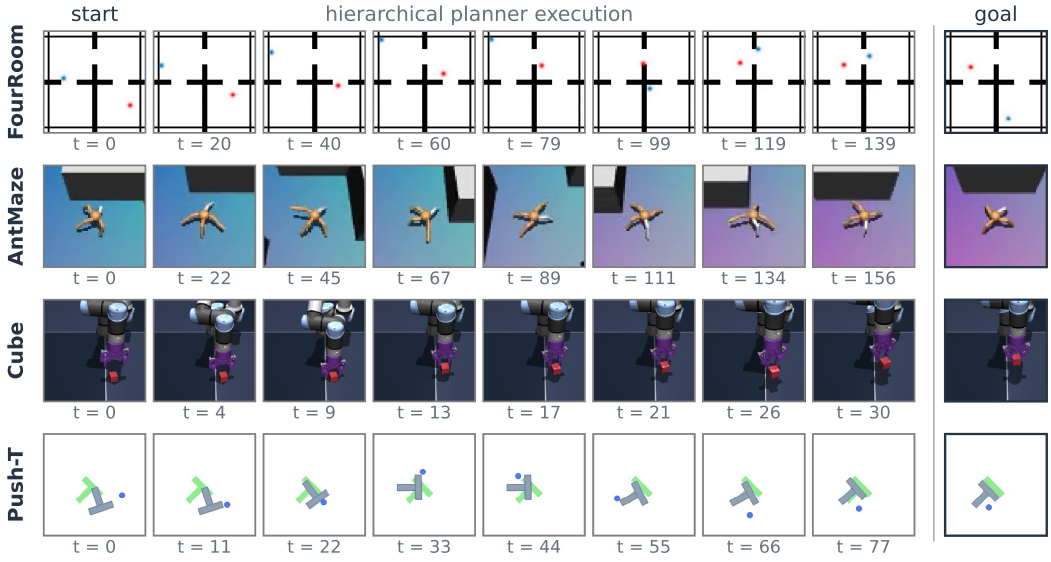

Figure 10: **Successful hierarchical planner executions.** Selected H-JEPA executions on one evaluation task per environment, using three-level models for FourRoom and AntMaze and two-level models for Cube and Push-T. These checkpoints also appear in Fig. 5 (bottom); the recordings use their original evaluation settings rather than the compute-selected sample counts. Each row shows eight observations sampled uniformly in environment steps, followed by the goal observation supplied to the planner. Rows end at first recorded success, except Cube, which shows the full 30-step execution to include the completed lift. Labels give elapsed environment steps.

segments centered on a grasp, so the task requires manipulating the object. Visual AntMaze uses expert rollouts between maze cells at grid distance three (82 steps on average), with neutral-pose start and goal states. The planner receives goal images and, on AntMaze, proprioception; success is determined by the environment's goal test within the episode budget in Table 7 (on AntMaze, within 1.0 of the goal xy). Fig. 10 shows one successful H-JEPA execution per environment. We use separate task sets for the compute sweeps in Fig. 5 (top) and the final evaluations in Fig. 5 (bottom) and Tables 1 and 9. The final evaluation set contains 50 start–goal tasks per environment, generated by the same process and with the same difficulty settings as the compute-sweep tasks, but using different task-generation seeds. Planner settings are fixed before evaluation on these new tasks. All methods and model depths share the same tasks within each set.

Shared solver. Every active level optimizes candidate action sequences by differentiating through its latent rollout, using AdamW for at most 90 iterations with early stopping when the best cost stalls. Candidates are initialized from a Gaussian prior, fitted to the training macro-actions at upper levels, and the lowest-cost candidate is returned. The latent-distance cost scores the last two predicted

states at level 1 and the last state at upper levels. H-JEPA plans top-down, passing the first predicted latent as a subgoal for the level below, whose rollout is encoded into the upper latent space for comparison (§2.2). On Push-T and Cube, level-1 actions are also pooled and matched to the first level-2 macro-action with weight 0.2; upper-level optimization adds Gaussian action noise of standard deviation 0.01. Both terms are disabled on FourRoom and AntMaze.

Replanning and horizons. All models execute two level-1 action blocks (10 environment steps) before replanning from scratch, truncating the final commit at the episode budget. Level 1 advances 5 environment steps per predicted step, and each upper level doubles that interval. For LeWM and the top H-JEPA level, the horizon decreases from the configured H in Table 7 to one across the episode's replanning calls. Each lower level holds horizon two until the level above reaches one, then decreases from its own configured H to one over the remaining calls. An upper level at horizon one is skipped. FourRoom and AntMaze retain the goal cost in the highest skipped level's latent space; Push-T and Cube use the active level's own goal embedding.

Reported settings. Table 7 gives the operating points in the depth comparison. On the compute-sweep tasks (Fig. 5, top), LeWM and H-JEPA select the candidate allocation with the highest observed mean success rate under 100 mean planner TFLOPs per episode, breaking ties by lower compute. Learning rates and horizons were fixed from earlier experiments, before the compute sweeps and final evaluation. The compute sweeps vary only candidate trajectory counts, with all other planner settings held fixed. These settings are then evaluated without further tuning on the separate 50-task set used for Fig. 5 (bottom) and Tables 1 and 9, which report success on this final evaluation set. Results report mean and SE across three training seeds: the selected evaluations pair model and planner seeds 42, 43, and 44.

Table 7: **Planning settings for the simulated depth comparison.** S: candidate sequences; η: AdamW learning rate; H: configured horizon in predicted steps of that level, used by the schedule described above. Tuples list levels from lowest to highest, except the H-JEPA learning-rate row, which lists level one and the rate shared by all upper levels. Episode budgets are in environment steps. Cube's horizon-one upper solvers are skipped, so their listed candidate counts are unused.

	FourRoom	AntMaze	Push-T	Cube
Episode budget	210	350	90	30
η, LeWM	0.1	0.3	0.05	0.05
η, H-JEPA (L1, upper)	$(0.08, 0.1)$	$(0.06, 0.1)$	$(0.1, 0.075)$	$(0.1, 0.075)$
H, LeWM	23	23	15	4
H, two-level H-JEPA	$(2, 12)$	$(4, 9)$	$(2, 8)$	$(2, 2)$
H, three-level H-JEPA	$(2, 2, 6)$	$(4, 2, 5)$	$(2, 2, 4)$	$(2, 2, 1)$
H, four-level H-JEPA	$(2, 2, 2, 3)$	$(3, 2, 2, 3)$	$(2, 2, 2, 2)$	$(2, 2, 1, 1)$
S, LeWM	200	100	800	800
S, two-level H-JEPA	$(100, 800)$	$(25, 200)$	$(50, 400)$	$(400, 100)$
S, three-level H-JEPA	$(50, 200, 200)$	$(100, 100, 400)$	$(25, 50, 200)$	$(25, 800, 800)$
S, four-level H-JEPA	$(50, 200, 200, 400)$	$(50, 200, 25, 50)$	$(200, 800, 800, 800)$	$(200, 50, 800, 800)$

Plans are executed under a *commit schedule*: the agent replans at environment times $t_0 = 0 < t_1 < t_2 < \cdots$; at phase j it encodes the observation at t_j, runs the top-down solve of Eqs. (4)–(6) from that state, executes the first $\xi_j = t_{j+1} - t_j \leq H_1$ actions of $a^{(1),*}_{0:H_1-1}$, and discards the rest. The schedule need not be uniform, and the last phase is truncated by whatever remains of the episode budget. Committing a single command per phase, $\xi_j \equiv 1$, is fully closed-loop control; committing an entire plan in one phase, $\xi_0 = H_1$, is open-loop single-shot execution, which the DROID protocol of §J.1 uses. Table 8 gives its settings.

DROID is offline and open-loop: one plan covers the whole window, so the commit width equals the horizon and nothing is replanned. It is scored on the 16 clips of §J.1 with three planner seeds and three training seeds.

Table 8: **Planning settings on DROID.** S: candidate sequences; η: AdamW learning rate; H: configured horizon in predicted steps of that level. Tuples list levels from lowest to highest. Every level runs AdamW for at most 90 iterations with early stopping when the best cost stalls; the cross-level action cost is 0 and the subgoal weight β of Eq. (6) is 0.5, for both two-level arms. Planning is open-loop: the whole level-1 plan is executed, so the commit width equals the level-1 horizon. The listed η and S are the selected configuration for each bar in Fig. 7b; the compute-Pareto of Fig. 7c sweeps S and η.

	DROID, 5 fps
Episode budget	36
η, LeWM+IDM	0.01
η, HWM (L1, L2)	$(0.01, 0.1)$
η, H-JEPA (L1, L2)	$(0.03, 0.01)$
H, LeWM+IDM	36
H, HWM	$(36, 12)$
H, H-JEPA	$(36, 12)$
S, LeWM+IDM	16
S, HWM	$(16, 16)$
S, H-JEPA	$(4, 4)$

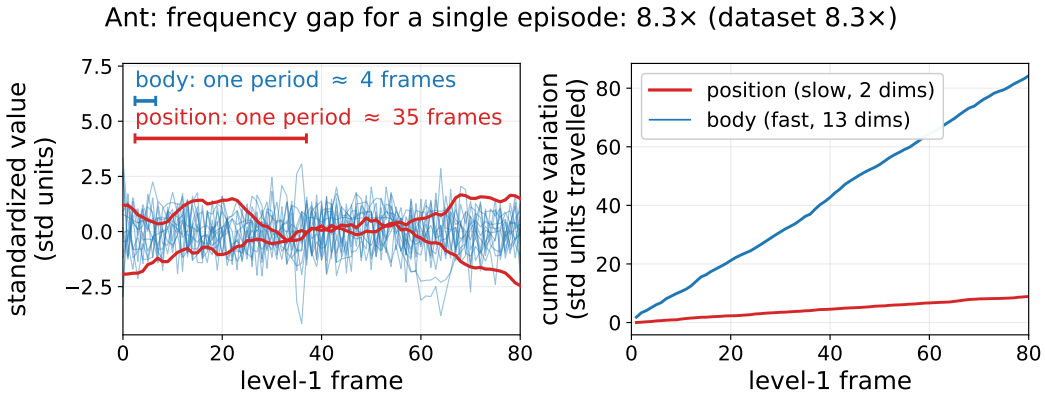

Figure 11: **Temporal separation within one AntMaze episode.** *Left:* normalized global position (red) and 13 body-state dimensions (blue) over 81 level-1 frames. Brackets mark inverse spectral centroids: about 35 frames for position versus 4 for body state (8.3× gap). *Right:* cumulative absolute normalized increments, averaged across dimensions; body state accumulates roughly ten times more variation than position. Velocity dimensions are omitted for legibility; including them leaves the frequency gap unchanged.

F Temporal-Frequency Separation and Selective Abstraction

F.1 Frequency Separation within a Trajectory

The entity-frequency gap of §3 is a dataset-level statistic: one spectral centroid per entity, pooled over episodes, and their fast/slow ratio. Fig. 11 shows the raw behaviour that statistic summarizes on a single AntMaze training episode, so the reader can verify the gap by eye rather than trust the aggregate. The episode is drawn from the same dataset and decimated to the same level-1 frame rate (frameskip 5) that the centroids in Fig. 12 are computed from; it is the episode among the first 64 whose own fast/slow centroid ratio is closest to the dataset value, so it is representative rather than the most striking case (per-episode ratios have median 6.4× and interquartile range 4.9–7.8×).

The entities are AntMaze's probe targets: the slow entity is the global position (xy, two dimensions) and the fast entity is the body state. For legibility we plot the body's 13 position-like dimensions (torso height, root orientation, and the eight hip and ankle joint angles) and omit the 14 joint- and root-velocity dimensions, which oscillate at least as fast and would fill the panel; the episode's gap is 8.3× with either choice. Because the entities have different units, every dimension is normalized to zero mean and unit variance over the episode, making the spectral centroid amplitude-invariant.

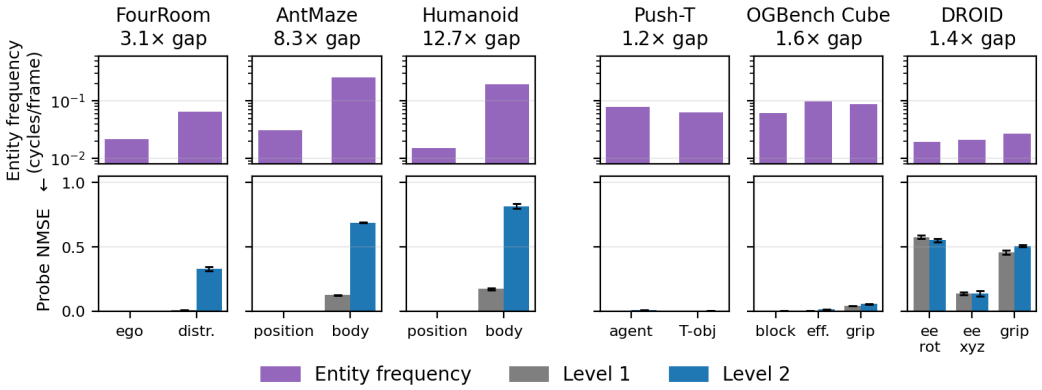

Figure 12: **Per-entity view of selective abstraction.** The left group shows FourRoom, AntMaze and Humanoid, the right group Push-T, OGBench Cube and DROID. *Top:* entity spectral centroids in cycles per level-1 frame (log scale); *bottom:* paired level-1 and level-2 probe macro NMSE, under a probe head matched across the two levels within an environment and differing between them. Error bars show one standard error (SE) over three training seeds. All entities in an environment use the same level-2 configuration.

The trajectory illustrates the slow position changes and fast body-state variation summarized by the dataset-level frequency gap. Frequency separation alone does not establish a difference in predictability, since fast periodic motion can remain predictable. The per-entity probes below show how this separation relates empirically to content retained at level 2 across environments.

F.2 Per-entity probes across environments

For each state entity, we compute the spectral centroid of its amplitude-normalized trajectory stream at the level-1 sampling rate, pooling over episodes. The entity-frequency gap is the ratio of the largest to smallest entity centroid. Fig. 12 expands Fig. 4 with each entity's centroid and paired level-1 and level-2 probe NMSE, computed from two-level H-JEPA models. Humanoid, AntMaze, and FourRoom have gaps of 3.1–12.7×: body-state or distractor decoding worsens at level 2 while agent position remains near its level-1 floor. Push-T, OGBench Cube and DROID have gaps of 1.2–1.6× and show little selective abstraction, with comparable entity recoverability at both levels. The relevant comparison is the within-entity change between levels: absolute NMSE differs across environments, particularly for DROID's natural video.

G Data specialization across levels

We investigate whether training different hierarchy levels on distinct types of data improves planning. We use stagewise training for this analysis, freezing each lower level before training the next so that each level can be trained on its own dataset.

Experimental setup. We use a two-level hierarchy for ease of analysis, with Visual AntMaze's two OGBench dataset types, `explore` and `stitch`, and their `union`. For stagewise training, we vary the dataset independently at each level; for end-to-end joint training, both levels use the same dataset. Fig. 13 compares the resulting two-level hierarchical planners with flat level-1 planners trained on each of the three datasets.

Matched data: a second level helps only when the data is temporally structured. Along the diagonal, adding an abstract level improves long-horizon planning on `stitch` and `union` (end-to-end hierarchical planning SR rises to 21 from the single-level 11 on `stitch` and to 54 from 19 on `union`) but not on `explore`, where neither two-level model (`stagewise` 11, end-to-end 11) beats the single-level baseline (18). We read this as the level-2 model being unable to form a useful temporal abstraction from purely exploratory rollouts, whose trajectories lack the temporal correlation the abstract level relies on; hierarchy then buys nothing.

Mismatched data: the two levels prefer different data. The pattern in the `stagewise` panel is that the low-level model benefits from broad, weakly-correlated exploration while the abstract level

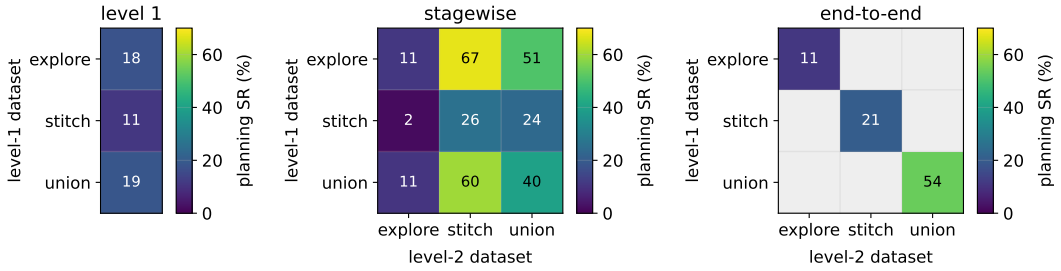

Figure 13: **Planning with different training datasets on Visual AntMaze.** Planning success rate (GD online, mean over three seeds), using `explore`, `stitch`, or their `union` for training. *Left:* flat level-1 planning, with one model trained on each of the three datasets. *Middle:* two-level hierarchical planning after stagewise training, covering all nine combinations of level-1 (rows) and level-2 (columns) datasets. *Right:* two-level hierarchical planning after end-to-end joint training, with both levels trained on the same dataset in each run, giving only diagonal entries. Grey cells were not run. The end-to-end runs are flops-matched to the stagewise ones.

benefits from more temporally coherent, goal-directed trajectories. The best configuration anywhere in Fig. 13 is exactly this pairing (`explore` at level 1 and `stitch` at level 2) reaching 67, well above every matched-dataset cell of either procedure, including end-to-end on `union`/`union` (54) and stagewise on `union`/`union` (40), while the reverse assignment (`stitch` at level 1, `explore` at level 2) collapses to 2. It is the *pairing* (exploratory data low, coherent data high) not either dataset on its own, that drives the gain. This is a practical case for stagewise training: freezing the lower-level model lets us specialize the upper level's data without changing the lower level's learned dynamics, a choice the default joint procedure does not offer.

H Planning-cost ablations across environments

Table 9 extends the AntMaze cost ablation of Table 1 to FourRoomDistractors, OGBench Cube and Push-T, on the tasks of Fig. 5 (bottom). Within each model, we hold the level-1 rollout dynamics and planner settings fixed and vary only the latent space used to score the goal cost.

FourRoom: gains from projected costs. As on AntMaze, at least one upper-level cost improves mean success over the native level-1 cost at every tested depth. The best projection also exceeds the separately trained LeWM baseline at three and four levels. The deepest cost is not always best: the four-level model has its highest mean success with the level-2 projection, although the projected-cost error bars overlap.

Cube and Push-T: no clear gain from projected costs. These environments do not exhibit the selective abstraction observed on AntMaze and FourRoom. On Cube, the level-2 projection lowers success in the two-level model. At three and four levels it gives similar success to the native cost, while the level-3 and level-4 projections roughly halve success. On Push-T, projected and native costs give similar success at two and three levels; the four-level model has near-zero success under every cost and with hierarchical planning. These results show that an upper-level cost need not improve planning; they do not establish that selective abstraction alone causes the gains in the other environments.

I Prediction-cost monotonicity

Table 10 expands the comparison in §4.2. It reports goal-cost monotonicity over full held-out trajectories and subgoal-cost monotonicity within shorter segments, using the two-level HWM and H-JEPA models in the four simulated benchmarks. These measurements characterize costs along expert trajectories rather than the success of optimized plans.

The temporal contribution is the replacement of one long optimization problem by a sequence of shorter subgoal problems. We examine this decomposition using H-JEPA and HWM planners, both of which decompose the problem into subgoals. For simplicity, we use two-level models for this comparison. Rather than optimize a single trajectory all the way to a distant goal (a horizon over which the goal cost need not descend cleanly, and can plateau or re-ascend before the goal), the level-1

Table 9: **Planning-cost ablations across environments.** Success (%, mean ± SE, three seeds), using the same protocol as Table 1. Within each H-JEPA row, only the cost space varies across the four level-1 planning columns; LeWM and hierarchical H-JEPA are references. For FourRoom, bold marks the highest mean among cost spaces. FourRoom gains from projected costs, while Cube and Push-T show no clear benefit.

		Level-1 planning: vary cost space				Hierarchical planner
Environment	Depth	Native L1 ↑	L2 proj. ↑	L3 proj. ↑	L4 proj. ↑	H-JEPA ↑
FourRoom Distractors	LeWM (flat)	40.7 ± 6.8				
	2 levels	9.3 ± 0.7	$\mathbf{38.7 \pm 6.6}$			81.3 ± 6.7
	3 levels	43.3 ± 8.7	65.3 ± 5.5	$\mathbf{71.3 \pm 4.1}$		96.0 ± 1.1
	4 levels	68.7 ± 4.7	$\mathbf{72.0 \pm 3.1}$	66.0 ± 3.1	70.7 ± 2.7	80.0 ± 4.0
OGBench Cube	LeWM (flat)	32.0 ± 6.1				
	2 levels	24.0 ± 1.2	13.3 ± 0.7			47.3 ± 2.7
	3 levels	24.7 ± 1.3	28.0 ± 4.2	14.0 ± 3.5		60.0 ± 5.0
	4 levels	26.7 ± 2.7	25.3 ± 2.4	12.0 ± 2.0	12.0 ± 2.3	55.3 ± 4.4
Push-T	LeWM (flat)	40.0 ± 2.3				
	2 levels	36.7 ± 1.8	38.0 ± 3.1			45.3 ± 1.3
	3 levels	30.7 ± 1.8	28.0 ± 4.0	24.7 ± 4.1		17.3 ± 3.5
	4 levels	1.3 ± 0.7	1.3 ± 0.7	0.7 ± 0.7	1.3 ± 1.3	0.7 ± 0.7

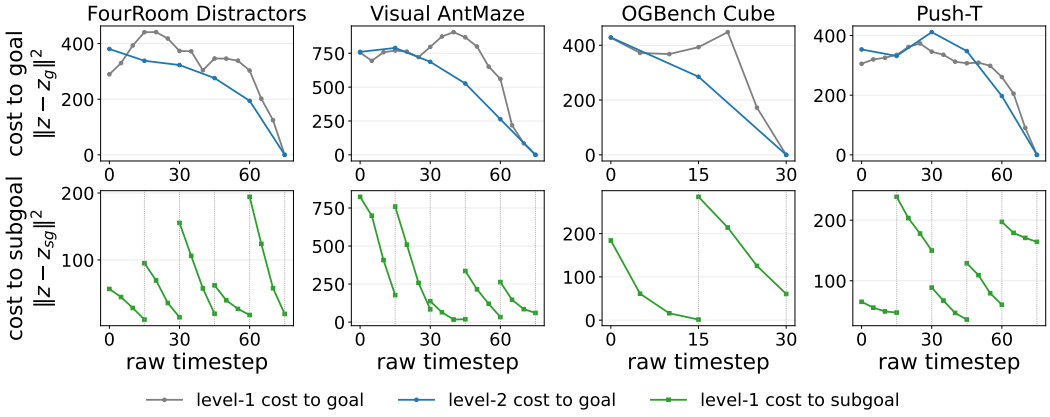

Figure 14: *Top:* **Goal cost in level-1 and level-2 space.** For a held-out expert trajectory we encode each frame at both levels, and compute the distance between each timestep's latent $z_t^{(\ell)}$ and the goal (final-timestep) latent, $\|z_t^{(\ell)} - g^{(\ell)}\|_2^2$. At level 2 we subsample at the training stride ($= 3$). *Bottom:* **Subgoal tracking cost.** The level-2 subgoals are the true level-2 latents at the subsampled timesteps. Within each subgoal segment we re-anchor the level-1 rollout from the encoded latent at the segment start, unroll it to the next subgoal, project it into level-2 space, and measure its distance to that subgoal. Each segment is a bounded-horizon descent toward the next subgoal, like a receding-horizon planner, and the sawtooth pattern shows how hierarchical planning decomposes the long-horizon task into shorter, easier subtasks with monotonically decreasing subgoal costs.

planner solves a chain of short hops, each a bounded-horizon descent to the next level-2 subgoal and each a better-conditioned subproblem than the original. Fig. 14 shows this on a held-out expert trajectory. Over the full horizon the level-1 goal cost stalls and re-ascends on its way to the goal (top); the level-2 goal cost is smoother, yet even it is not guaranteed to decrease monotonically: on Push-T it climbs to a peak before falling (top right). Re-anchoring the level-1 rollout at each subgoal, however, makes the descent monotone: within every segment the cost falls cleanly toward its subgoal target (bottom). We also quantify this effect across held-out trajectories. We measure monotonicity as the negative Spearman rank correlation between the squared latent-distance cost and time, with 1 indicating a strictly decreasing cost sequence. Across held-out trajectories in all four environments, scoring re-anchored level-1 rollouts against H-JEPA subgoals increases monotonicity from 0.48–0.75 under the full-horizon goal cost to 0.89–1.00 (Table 10). The improvement occurs for both HWM and H-JEPA, supporting a benefit from shorter subgoal problems even without distinct abstract latent spaces and helping explain HWM's gains over LeWM in long-horizon planning (Fig. 5, bottom).

Table 10: **Prediction-cost monotonicity** ($-\mathrm{Spearman}$ of $\|z^{(\ell)} - g^{(\ell)}\|_2^2$ against time; higher is better; mean over three training seeds, held-out episodes). *L2* columns score each model's level-2 encoding against the final goal; *subgoal* scores a level-1 rollout, re-anchored to the true state at each subgoal (bounded horizon), against the level-2 subgoals (per-segment).

		HWM		H-JEPA	
Environment	Level-1 ↑	Level-2 ↑	subgoal ↑	Level-2 ↑	subgoal ↑
Visual AntMaze	0.48	0.52	0.94	**0.68**	**0.94**
Push-T	0.75	0.77	0.85	**0.77**	**0.89**
OGBench Cube	0.65	0.74	0.79	**1.00**	**0.99**
FourRoom Distractors	0.56	0.62	0.92	**0.85**	**1.00**

J DROID: CORPUS, PLANNING EVALUATION AND OFFLINE METRIC

DROID exposes no simulator, at least none that runs without the ray-tracing cores IsaacSim requires [60], so every DROID number in this paper is offline: plans are scored against logged expert trajectories. This section collects the whole apparatus in one place: the evaluation protocol (§J.1), the metric and what it does and does not measure (§J.2), what the plans look like once decoded (§J.3), and the re-encoded corpus every DROID model is trained on (§J.4).

J.1 PLANNING-EVALUATION PROTOCOL

Clips. The headline set is the *16-clip evaluation set*: 16 pick-and-place clips, with good lighting, good visibility of the manipulated objects from the left and right cameras, and big enough manipulated objects. Once chosen, the exact frame indices of each clip's evaluation slice are frozen so that every run scores the same initial and goal frame and the same ground-truth actions. For each of the 16 clips, contact waypoints $[\textit{init}, \textit{grasp}, \textit{pre-release}, \textit{place}]$ are read off the gripper-closure trace (grasp = first frame with grip $> 0.85\times$max), and each clip is a fixed-length strided window *centered on the grasp*. Episodes whose window does not fit inside the video are dropped rather than clamped. Centering on the grasp is what makes our eval procedure a probe of non-greedy behavior: a plan that drives straight to the deposit matches the ground-truth net displacement but misses the descent to the object and the transport of the object to the target location, which is why the metric of §J.2 scores the shape of the whole path. Each window spans 108 recorded frames, $\sim$7.2 s, with the goal as its last frame. A clip is read at the frame rate its model was trained at, so the level-1 horizon is that span divided by the stride: $H_1 = 36$ at 5 fps, the setting this paper trains and evaluates at.

A note on frame rates. DROID is recorded at 15 Hz [42], but its video containers are *tagged* 60 fps, so a loader that derives its temporal stride from the tag samples at a quarter of its configured rate. The frozen-encoder baselines [77, 2] are configured at 4 fps with such a loader, which corresponds to 1 frame per second of recorded time, and we evaluate them at that native rate on the matching manifest (15 clips; one clip's window does not fit at 1 fps).

Leak control. Every DROID model in this paper is trained on one corpus: the re-encoded corpus of §J.4, with the 16 evaluation clips of Fig. 17 removed. Plan-evaluation fidelity is therefore a held-out number throughout. Only those 16 clips are removed, so the auxiliary unroll and light-eval metrics remain train-set metrics.

Task and action space. Actions are 7-dimensional end-effector delta-poses (Cartesian xyz, roll-pitch-yaw, gripper). The goal is specified as a single RGB frame, encoded by the model's own encoder; the planning cost is a Euclidean distance in that latent space. All metrics use the first three (translation) dimensions; the 7-dimensional variant is dominated by gripper nuisance.

J.2 THE FRÉCHET FIDELITY OFFLINE METRIC

Fréchet fidelity is defined in Eq. (7); the paths it compares are $p_{1:T+1}$ and $\tilde{p}_{1:T+1}$, the origin-prepended cumulative xyz paths of the planned and ground-truth delta-pose sequences. Each planner seed in $\{42, 43, 44\}$ is reduced to one scalar by averaging over clips; we report the median over

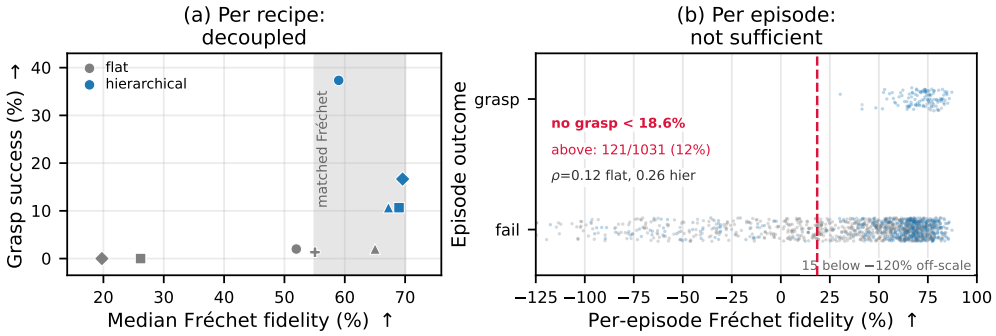

Figure 15: **On OGBench Cube, where the outcome *is* observable, Fréchet fidelity separates the episodes that can succeed from those that cannot, but not the ones that do.** One frozen checkpoint, 9 configuration groups $\times$ 3 planner seeds $\times$ 50 episodes. *(a)* One point per planner configuration; in the shaded band the flat and hierarchical planners have equal offline fidelity and opposite grasp outcomes. *(b)* Every episode: none succeeds below a fidelity threshold, and most above it still fail.

(training seed $\times$ planner seed) cells, because the per-clip mean is in absolute meters and is therefore dominated by large-motion clips.

A path metric is the point. The cumulative xyz action trajectory error (ATE) (used in [77, 7]) compares net displacements, $|\sum \text{planned} - \sum \text{ground truth}|$ per dimension, and is therefore blind to the very detour the grasp-centered protocol above is built to test: a straight-line plan to the deposit scores as well as one that descends to the object and carries it. Within path metrics the choice was made empirically: Fréchet agrees with DTW and with an arc-length-split reach/transport score. We calibrate the metric in two simulated environments where the outcome is observable: OGBench Cube, whose outcome is *binary* (grasp success), and RoboCasa CloseDrawer [57], a kitchen task whose scenes vary between episodes like DROID's and whose outcome is *continuous* (graded drawer-closure progress). There RoboCasa serves as a measuring instrument: we vary the planner on one frozen checkpoint to obtain a spread of outcomes, and read how well the metric tracks them.

Against a binary outcome it is a necessary condition. On OGBench Cube (Fig. 15) no episode succeeds below a floor of Fréchet fidelity, and clearing that floor does not say which episodes will succeed. A grasp either closes or it does not, and no continuous measure of path agreement separates the episodes that complete it from those that come close, so this is what we expect of a path metric rather than a shortcoming of this one.

Against a continuous outcome it ranks progress. Scored against graded drawer closure on RoboCasa CloseDrawer (Fig. 16), the within-configuration rank correlation is positive and moderate, and it survives controlling for executed path length and for per-episode difficulty. This is what the metric is built for: it measures how far along the expert path a plan got.

Together the two set how we read it. It is the end of the path that turns progress into a completed task, so a plan can follow most of the expert path and still not finish. On Cube this shows up as a small offline advantage for the hierarchy alongside a large advantage in grasp success: the Fréchet gap understates the outcome gap. On DROID, where no outcome exists, we therefore read a gap in Fréchet fidelity as evidence that one planner is better, not as a measure of how much better.

J.3 Decoded plans

The Fréchet numbers in §4.3 summarize a plan by one scalar. Because DROID affords no execution, it is worth looking at the plans themselves. Fig. 17 decodes them: the visual decoder is applied to the level-1 latents the planner rolls out under its own committed commands, and to the level-2 macro-plan that supplies its subgoals.

Two things are visible that the scalar does not carry. First, the imagined rollouts move the gripper *down toward the manipulated object* rather than straight at the goal frame, which is the non-greedy behaviour the evaluation clips are chosen to elicit. That is true even of the clip scoring at the zero-action floor: a fidelity of zero means the planned path did not follow the expert's, not that the plan

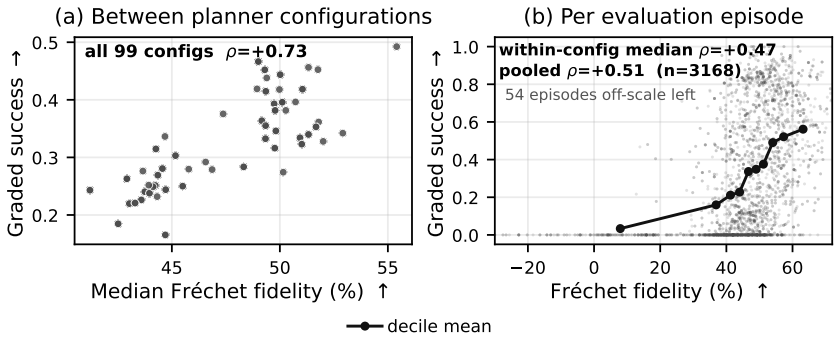

Figure 16: **Against a continuous outcome the same offline metric ranks degree of progress.** RoboCasa CloseDrawer, one frozen checkpoint, 99 planner configurations $\times$ 32 held-out episodes, budget 200 steps; the configurations are varied to spread the outcomes, not tuned for task performance. x is Fréchet fidelity, computed exactly as on DROID; y is graded drawer-closure success, a continuous outcome DROID cannot report. *(a)* One point per planner configuration, median fidelity against mean graded success. *(b)* Every episode, with the mean of each fidelity decile joined. The claim rests on the *within-configuration* rank correlation, which holds episode difficulty and planner settings fixed; the pooled and between-configuration values are inflated by a between-configuration mean shift.

had no idea where the object was. Second, the level-2 macro-plan is a coarse but coherent version of the same trajectory rather than a degenerate one, which is what the coarse cost of Eq. (4), scoring only the final predicted state, requires of it. All 16 clips are shown, ordered best to worst, so both observations can be checked against the full fidelity distribution rather than against a selection. The level-2 macro-plan advances one frame for every three of level 1, so it shares the level-1 strip rather than occupying a row of its own that would be two-thirds empty.

J.4 The re-encoded training corpus

Raw DROID is impractical to train on repeatedly: the loader would touch $\sim$5.5 TB and H.264 decoding, not the GPU, sets the epoch time. Every DROID result in this paper therefore runs on a re-encoded copy: each of the three camera views downscaled to 256×256 (short-side center crop, then scale) with `libx264` at CRF 18, each per-episode trajectory repacked into a flat `.npz`, and the stereo files our loader never opens dropped. Frame count and timestamps are preserved instead of being decimated to the training rate, which keeps the stride phases that decimating would collapse into one. The build is 127.6 GB for 74,530 episodes, a $\sim 45\times$ reduction that decodes $17.9\times$ faster. The re-encode is lossy, so we checked what it costs: trajectories and assembled loader samples come back bitwise identical, the video costs a median PSNR of 33.75 dB, and a flat configuration retrained on the re-encoded corpus but scored on the *original* clips matches raw within noise. We release the corpus, its manifests and the build script. DROID is CC-BY 4.0 [42], which permits redistributing an adapted version with credit and a statement of the change; we mark the re-encode as such.

J.5 Future work

Every model here encodes one camera at a time: both settings pool either the left or the right exterior view and sample one per training window. DROID's wrist camera is never used during training. Precise manipulation is where that costs most, since the gripper and the object it closes on are small from an exterior angle and often self-occluded, while a wrist view sees them directly. Encoding all available views jointly, including the wrist view, rather than sampling among them is the clearest route to better performance, and could be integrated into the method, but we leave this for future work. There could be motivation to extend this method to videos with no action annotation, where the IDM loss would not be an available lever to avoid slow-feature collapse. §L suggests that the mechanism to prevent slow-feature cannot fall back fully on SIGReg, applied in a way that repels the representations of different timesteps.

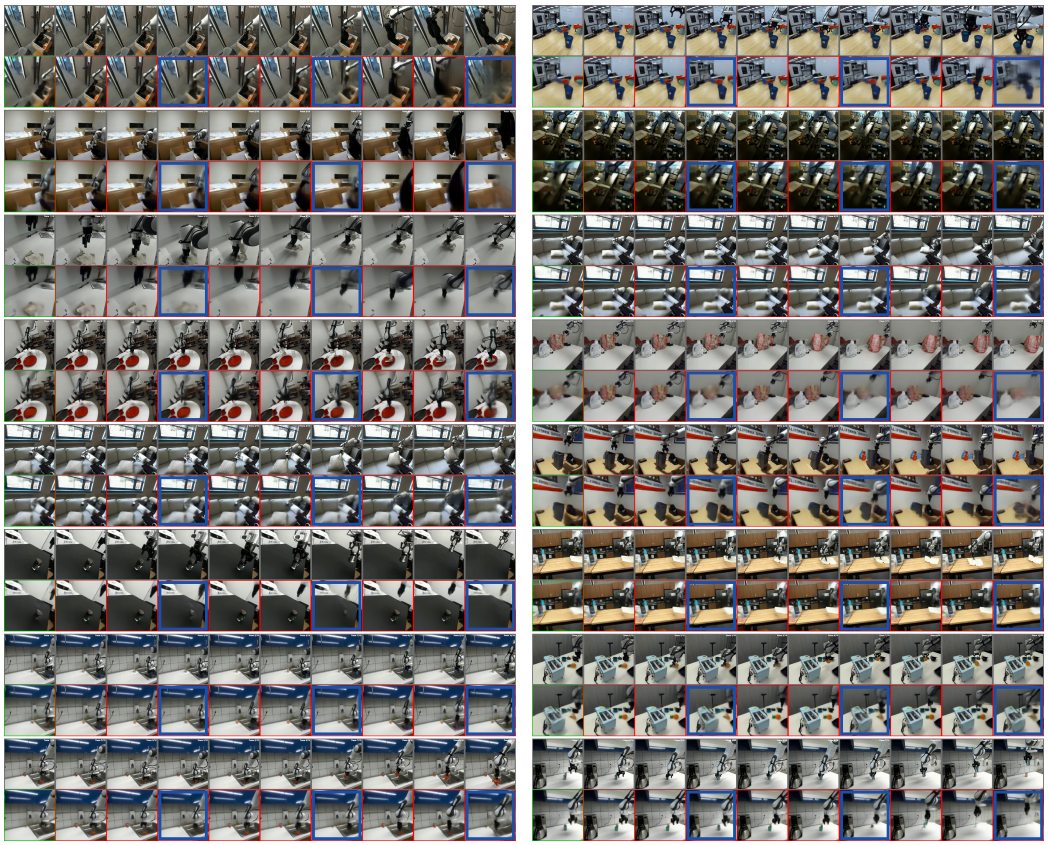

Figure 17: **What the DROID planner imagines, on every clip of the evaluation set.** Two-level stagewise model, held-out, training seed 1; all 16 clips ordered best to worst by Fréchet fidelity, down the left column then the right, $+75\%$ down to $+0\%$ at the zero-action floor. Each clip is two strips: above, the logged expert clip; below, the decoded plan, which is the level-1 rollout under the planner's committed commands with the level-2 macro-plan interleaved into it at every third column (temporal stride 3), ringed in blue. Columns are timesteps. The leftmost is the context frame the planner is given (green); the rest are imagined (red), and blur along the rollout, which is the decoder's fidelity limit and not a planning failure.

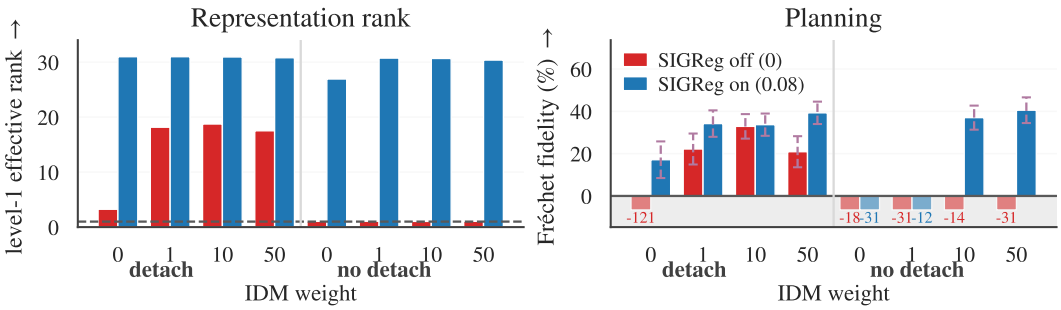

Figure 18: **SIGReg restores representation rank; the inverse-dynamics term is what turns rank into planning fidelity.** Full grid over the level-1 DROID configuration at a reduced two-GPU budget: SIGReg $\in \{0, 0.08\}$ $\times$ IDM weight $\in \{0, 1, 10, 50\}$ $\times$ prediction-target detachment, 16 cells, one training seed each, all read at epoch 100. *Left:* level-1 effective rank [25], max 32; the dashed line is full collapse. *Right:* Fréchet fidelity on the 16-clip set; the grey band holds sub-floor cells, drawn as fixed-depth stubs annotated with their true value. With SIGReg on, rank saturates in every cell including IDM $= 0$, yet those cells plan at or below the floor: rank is necessary and not sufficient. Planning bars are $\pm$ standard error (SE) across the 16 clips; with one run per cell, no bar is a seed interval.

K ANTI-COLLAPSE: SIGREG, INVERSE DYNAMICS AND DETACHMENT

Given $Z^{(\ell)} \in \mathbb{R}^{n \times D}$, a matrix of n level-ℓ embeddings of width D, SIGReg samples M unit directions $u_m \in \mathbb{S}^{D-1}$ and projects the embeddings to one-dimensional samples $h_m = Z^{(\ell)} u_m$. It averages the Epps–Pulley normality statistic [22, 6] across these projections:

$$\mathrm{SIGReg}\Big(Z^{(\ell)}\Big) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{T}_{\mathrm{EP}}(h_m), \tag{9}$$

where $\mathcal{T}_{\mathrm{EP}}$ compares the empirical characteristic function of h_m to that of a standard Gaussian. The statistic, its Gaussian weighting, and the Cramér–Wold argument [17] that reduces matching $\mathcal{N}(0, I_D)$ to matching every one-dimensional projection are those of Balestriero & LeCun [6]; the quadrature is that of the implementation released with LeWM [52], which exploits the even integrand to work on the half-domain $[0, 3]$ with 17 trapezoidal knots, at $\mathcal{O}(n)$ cost in the n samples entering one test. With one embedding per frame, the tensor entering the test is (T, N, D): one normality test per timestep, over the N sequences of the batch. SIGReg is not the only term carrying the anti-collapse burden: §K.1 separates what it, the inverse-dynamics term and prediction-target detachment each contribute, and §K.2 cross-validates the two coefficients that survive that grid. §K.3 covers how the statistic is synchronized across devices.

K.1 WHAT SIGREG, IDM AND DETACHMENT EACH CONTRIBUTE

§4.3 claims an inverse-dynamics term is required on DROID. We test that with a full grid over the three candidates for the anti-collapse burden, the SIGReg weight, the IDM weight and prediction-target detachment, at one training seed per cell (Fig. 18). The grid should be read internally rather than against the headline runs.

Rank against planning fidelity. SIGReg is what restores rank: without it the representation collapses unless IDM and detachment are both present. But visual representations with non-collapsed effective rank do not necessarily mean the JEPA world model will have high Fréchet fidelity after planning. With SIGReg on, rank saturates in every cell, including those with no IDM term at all, and those cells still plan at or below the do-nothing floor. The IDM term is what converts rank into planning fidelity, and it has to be weighted carefully to do so.

When detachment matters. With SIGReg off, detachment is the difference between collapse and a usable representation at every IDM weight. With SIGReg on and a large IDM weight, the cell that omits it is the best planner in the grid. It is a safeguard for a regime we do not train in, hence we do not use this target detachment in our headline setup displayed in table 5.

K.2 Cross-validating the SIGReg and inverse-dynamics coefficients

The grid of §K.1 studies which terms the objective needs, here we investigate how they are coupled. We run two searches over the pair formed by the SIGReg coefficient and the inverse-dynamics weight in Fig. 19. The level-1 search varies one coefficient at a time, which is what the grey cells record. Once settled on the optimal level-1 config, we freeze it and train several level-2 models on top of it, pairing each of the three training seeds of level-2 with the corresponding level-1 frozen model seed.

We first stress that the two coefficients have to be chosen as a pair rather than one after the other. A pooled embedding carries the whole frame, so the inverse-dynamics weight has to be large enough to force the encoder to keep the agent, and the SIGReg coefficient that wins then sits in a correspondingly tight band around whatever that weight is. That band is narrow in absolute terms as well, since one step up from the frozen level-1 coefficient costs the entire result and puts the model to zero performance. A search that varied one coefficient while holding the other at a value the first one does not suit would therefore read both axes as flat.

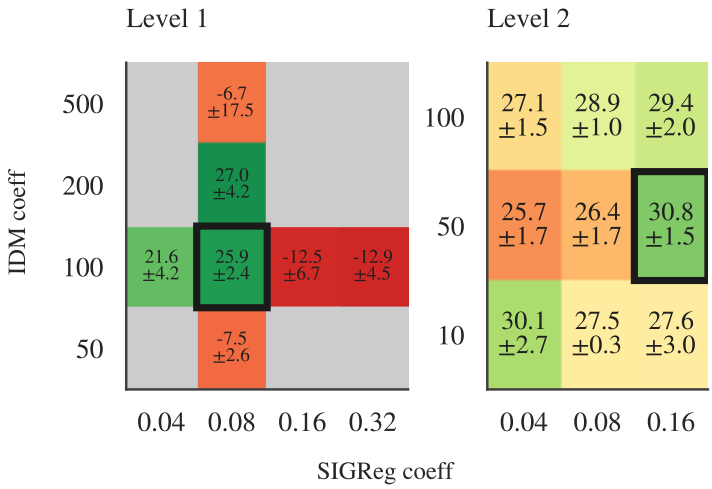

Figure 19: **The SIGReg and inverse-dynamics coefficients are chosen jointly, once per level.** DROID at 5 fps; Fréchet fidelity (%, higher is better), mean and standard error (SE) over three training seeds in every cell. *Left*, the level-1 search on single-level models; *right*, the level-2 search on stagewise models with level 1 frozen. These search runs are not the end-to-end headline models of Fig. 7b, so their values sit below it. Grey cells were never run, because the level-1 search varied one coefficient at a time. The heavy outline marks the coefficients we keep, which are the best cell of each grid. Each panel keeps its own color scale.

K.3 SIGReg under data parallelism

$\mathcal{T}_{\mathrm{EP}}$ is a statistic of the whole batch, not a sum of per-sample terms. Evaluating it on each rank's $n_{\mathrm{loc}} = n/W$ samples and averaging the W results gives the mean of W small-sample statistics, a quantity that shifts with the number of devices. We synchronize one level lower. The empirical characteristic function (ECF) underneath the statistic is a mean over samples, so with equal shards the global ECF is the average of the per-rank ECFs: we `all_reduce` the local real and imaginary ECFs before the quadrature and pass $n_{\mathrm{loc}}W$ into the sample-count prefactor. Two $M \times 17$ tensors cross the wire per call, one entry per direction and quadrature knot, whatever the batch size, and the statistic becomes exactly invariant to sharding (Table 11). The dependence on n that remains is deliberate: the prefactor cancels the $1/n$ variance of the ECF, so on target data the statistic sits at a floor (1.04 to 1.37 over $n = 32$ to 2048) and on collapsed data it grows linearly in n, holding the anti-collapse gradient per sample constant. So λ transfers across batch sizes, but not across choices of what counts as a sample.

Equal shards are assumed throughout, so we drop the last partial batch.

Table 11: **Our distributed SIGReg is exactly invariant to how a fixed total batch is sharded; a single-GPU implementation is not.** One fixed Gaussian matrix ($n = 256$, $D = 32$, $M = 64$ directions) sharded contiguously across W ranks in `float64`. The reference is the SIGReg released with LeWM [52], formula-identical to ours but with its prefactor set by the local batch and no collective; its code documents itself as single-GPU. The data follow the target, so its 14% drift is launch topology, not signal.

		$\mathcal{T}_{\mathrm{EP}}$	
Shards	$\|\nabla\|$ (ours)	ours	single-GPU reference
1×256	0.349154178012	1.372619178543	1.372619
2×128	0.349154178012	1.372619178543	1.180770
4×64	0.349154178012	1.372619178543	1.186233

L SLOW-FEATURE COLLAPSE IN JOINT-EMBEDDING PREDICTIVE OBJECTIVES

This appendix argues for the inverse-dynamics term of §4.3. Sobal et al. [71] exhibit an encoder that copies episode-constant noise and on which a variance–covariance regularizer vanishes. That shows a degenerate encoder attains zero loss, not that the objective *prefers* it; it assumes the identity predictor; and it covers only Gaussian noise. We strengthen all three points; no result assumes a parametric encoder or predictor. We suppress the level index ℓ of Table 2: E, F, I are the encoder, predictor and inverse-dynamics head, $z_t = E(o_t)$, $\mathcal{L}_{\mathrm{pred}}$ and $\mathcal{L}_{\mathrm{idm}}$ the losses of Eqs. (2) and (8), λ, γ, λ_{sim} the SIGReg, IDM and temporal-similarity weights, and N, T, D the minibatch shapes of Table 2. SIGReg averages G independent normality tests of n samples each, with Gn the number of embeddings in the minibatch.

L.1 THREE KINDS OF CONTENT IN A SEQUENCE OF OBSERVATIONS

The observation $o_t = O(\xi_t)$ is a deterministic function of an unobserved latent state $\xi_t = (b, x_t, y_t)$ taking values in a standard Borel space, and $z_t = E(o_t)$ encodes it. The **background** b is constant within an episode, varies across episodes (scene, camera, lighting), and is readable from a single frame: b is $\sigma(o_t)$-measurable. The **foreground** x_t is the agent-controlled content, $x_{t+1} = \Phi(x_t, a_t, \varepsilon_{t+1})$, where the innovation ε_{t+1} is a random input independent of the past that makes the foreground only *partially* predictable (contact, occlusion). The **distractors** y_t move but are action-independent and unpredictable, $y_{t+1} = \Psi(y_t, \zeta_{t+1})$ with their own innovation ζ_{t+1} (other agents, shadows), exogenous in the sense of Efroni et al. [21], Wang et al. [80]. A distractor that moves predictably (a clock, a fan) has no innovation, and nothing below separates it from the background.

Definition 1 (Slow-feature collapse)**.** An encoder is *slow-feature collapsed* when z_t is measurable with respect to the episode-invariant σ-algebra $\mathcal{C} = \sigma(b)$, the events determined by the background alone, so z_t is constant within every episode. Its *degree* is the within-episode variance share

$$\chi(E) = \frac{\mathbb{E}_b[\operatorname{tr} \operatorname{Var}_t(z_t \mid b)]}{\operatorname{tr} \operatorname{Var}_{b,t}(z_t)}, \tag{10}$$

the complement of the between-episode share measured in §L.2, and 0 at exact collapse.

L.2 WHERE THE LATENT VARIANCE GOES

§4.3 claims scene diversity separates DROID from Cube. We measure it directly in the latents, splitting the total latent variance into a within-episode and a between-episode part (Fig. 20). Latent size, camera site do not explain the gap. The within-episode part is $\chi(E)$ of Definition 1, so the diverse-scene environments start close to collapse. What it costs them is capacity rather than reachability: a per-episode mean cancels in a same-episode state-minus-goal difference, so the issue is how little of the latent is left for the arm, not whether the goal can be reached. Level 2 barely moves the share, hence the inverse-dynamics term at both levels.

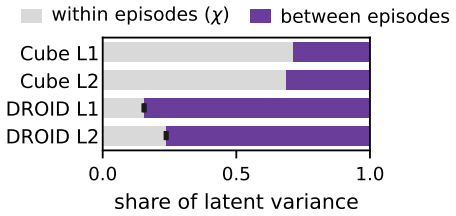

Figure 20: **Scene diversity sends latent variance between episodes.** Within-episode (the slow-feature degree χ of Definition 1) and between-episode shares of the latent variance. Bars are ± 1 standard error (SE) over three training seeds.

L.3 No regularizer on the marginal law can exclude it

Proposition 1 (Feasibility)**.** *Let $\mathcal{R}$ be any functional of the marginal law of z, the distribution of a single embedding $E(o)$ over the pooled corpus irrespective of time, with $\mathcal{R}(\mu_0) = 0$ for some law μ_0 on $\mathbb{R}^D$, and let $\mathrm{law}(b)$ be atomless. Then some E and F attain $\mathcal{L}_{\mathrm{pred}} = 0$ and $\mathcal{R}(\mathrm{law}(z)) = 0$ simultaneously, with F the identity in its state argument.*

Proof. An atomless law on a standard Borel space admits U with $U(b) \sim \mathrm{Unif}[0,1]$, and Q pushing $\mathrm{Unif}[0,1]$ to μ_0 exists for any μ_0; take $E = Q \circ U \circ b$, well defined on o because b is $\sigma(o)$-measurable (§L.1), and $F(z,a) = z$. Then $z_{t+1} = z_t$ within every episode, so $\mathcal{L}_{\mathrm{pred}} = 0$, and $\mathrm{law}(z) = \mu_0$. □

The atomless hypothesis is a population idealization. A corpus of finitely many scenes makes $\mathrm{law}(b)$ atomic and $\mathrm{law}(z)$ supported on at most that many points, which a normality test on far fewer embeddings per group cannot distinguish from μ_0; the construction fails outright only in the point-mass limit of a single scene (§L.4). Variance–covariance regularization, whitening, entropy penalties and SIGReg (Eq. (9)) all look only at the marginal law, so none can exclude the collapse; a barrier has to act on the joint law of z across time.

L.4 The objective selects by predictability, and the collapse is an optimum

Since F ranges over measurable functions, $\min_F \mathcal{L}_{\mathrm{pred}}$ equals the *irreducible risk* $\mathcal{E}(E) = \mathbb{E}\|z_{t+1} - \mathbb{E}[z_{t+1} \mid z_t, a_t]\|_2^2$ [84]. Define the *innovation sensitivity* $\nu(E) = \mathbb{E}\,\mathrm{tr}\,\mathrm{Var}(E(o_{t+1}) \mid \xi_t, a_t)$, the next-embedding variance that survives knowing the full latent state and the action.

Theorem 1 (Predictability bound)**.** *For any measurable E, $\mathcal{E}(E) \geq \nu(E) \geq 0$, and $\mathcal{E}(E) = 0$ if and only if $\sigma(z)$ is* action-closed*, i.e. $z_{t+1} = \Gamma(z_t, a_t)$ almost surely for some measurable Γ.*

Proof. $\sigma(z_t, a_t) \subseteq \sigma(\xi_t, a_t)$ because $o_t = O(\xi_t)$, and coarser conditioning cannot reduce expected conditional variance, which gives $\mathcal{E}(E) = \mathbb{E}\,\mathrm{tr}\,\mathrm{Var}(z_{t+1} \mid z_t, a_t) \geq \nu(E)$; $\mathcal{E}(E) = 0$ iff $\mathrm{Var}(z_{t+1} \mid z_t, a_t) = 0$ a.s., i.e. iff z_{t+1} is a measurable function of (z_t, a_t). □

Reading only b gives $\nu = 0$ and $\Gamma = \mathrm{id}$, hence $\mathcal{E} = 0$; reading x_t or y_t pays the variance injected by ε or ζ, unless the feature read is itself action-closed (a component of x_t that (x_t, a_t) determines, such as arm state under deterministic dynamics), which attains $\mathcal{E} = 0$ as well. Under any marginal constraint, which Proposition 1 shows the background can satisfy exactly, the collapse is therefore an optimum the objective cannot exclude. Among zero-loss solutions the theorem does not select; the implicit bias of training does [51], and in our runs it selects the collapse, whose predictor is the identity, which Table 12 measures: action sensitivity decays below 0.05 in all fifteen IDM-free seeds while the IDM-on baseline holds 0.28. In the tabular case the theorem is a corollary of Tang et al. [75, Thm. 6], whose non-collapse guarantee needs semi-gradient targets and a near-optimal predictor; our collapsed runs use the former and lack the latter. Conversely, a background identical in every episode makes $\mathrm{law}(b)$ a point mass, so Proposition 1 fails and SIGReg alone excludes the collapse: the fixed-camera environments of §4 need no inverse-dynamics term, and DROID, whose scenes change every episode, does (§4.3, Fig. 18).

It predicts the experiments it was not built for. The taxonomy and Theorem 1 account for three results of Sobal et al. [71]. *Fixed* distractors are fatal and *changing* ones harmless: fixed noise is background and becomes the collapse target, changing noise is a distractor and is shed. Their three-dot probe errors order as ν does: stationary 0.066, action-controlled 0.277, random 0.273, chance 0.235. And their inverse-dynamics model recovers the action-controlled dot (0.035) but loses the stationary one (0.272), as Proposition 3 predicts.

L.5 Pooling time into the sample axis: a real but small barrier

Proposition 1 concerns the distribution of a single embedding, but SIGReg is computed on a minibatch, and at exact collapse every sequence contributes T identical copies of one embedding, and a grouping with time in the sample axis puts them into the same normality test.

Proposition 2 (The barrier, per grouping)**.** *Let $\kappa \approx 1.053$ be the expected value of $\mathcal{T}_{\mathrm{EP}}$ on n samples drawn from its Gaussian target, asymptotically constant in n and the null floor of §K.3. At exact collapse, with the per-episode embeddings distributed as the target (the construction of Proposition 1), the SIGReg term exceeds its null value $\lambda\kappa$ by*

$$\Delta = \lambda\,(T-1)\,\kappa \tag{11}$$

whenever the grouping places time in the sample axis, so that a group holds several timesteps of the same sequence, and by zero whenever it places time in the group axis.

Proof. $\mathcal{T}_{\mathrm{EP}}$ compares an empirical characteristic function, an unweighted average over the n samples of a group, against its target, and scales the squared discrepancy by n. Repeating each of $k = n/T$ distinct values T times leaves that average unchanged, so its variance about the target remains that of k samples while the prefactor stays n, giving $\mathbb{E}[\mathcal{T}_{\mathrm{EP}}] = T\kappa$. With time in the group axis every group holds n distinct embeddings. □

The grouping this paper uses, (T, N, D), places time in the group axis: each normality test sees one timestep of N different sequences, so $\Delta = 0$ and SIGReg carries no barrier against slow-feature collapse at all. Even where a grouping does carry one it is negligible in training, for three reasons. First, the SIGReg statistic of a real encoder is dominated by how non-Gaussian its embeddings are: the logged values range from 570 to 3800, while duplicating every embedding T times adds only $(T-1)\kappa \approx 7$, so a collapsed encoder whose per-episode embeddings are Gaussian scores far *below* any healthy one; the barrier is relative to the null, not to training values. Second, weighted by λ, Δ is 10^{-5} to 10^{-3} against a prediction loss of order one. Third, λ cannot be raised to compensate, since it scales as $1/n$ to keep the encoder gradient fixed, so Δ shrinks with batch size. Table 12 shows the consequence: every IDM-free run collapses, later at larger λ but to the same endpoint, χ between 0.008 and 0.016 on every run past epoch 50 against 0.122 ± 0.001 for the IDM-on parent.

Table 12: **Training without the inverse-dynamics term, 5 fps setting: the SIGReg ladder (top) and time repulsion at its largest coefficient (middle).** Level-1 models at 5 fps, $\gamma = 0$, three train seeds, 100 epochs. These runs use the SIGReg grouping that places time in the sample axis, so they carry the Δ of Proposition 2; the barrier is what the ladder probes. "Onset" is the first epoch at which Fréchet fidelity (Eq. (7)) drops clearly below the zero-action floor, the next column the epoch at which the predictor's action sensitivity first falls below 0.05 (seed spread ± 2 epochs). Best fidelity is the highest before turning over; final fidelity and sensitivity are read over epochs 98–100; χ is Eq. (10) on the final checkpoint (one seed for the repulsion runs, three for the IDM-on row). Ladder rows carry no terminal value (at most two seeds reach epoch 100).

λ	λ_{sim}	Onset ↑	Sens. < 0.05 ↑	Best fidelity ↑	Final fidelity ↑	Final sens.	χ
9.77×10^{-6}	0	7	13	-0.64 (2.74)	—	—	—
1.95×10^{-5}	0	21	20	3.35 (1.40)	—	—	—
3.91×10^{-5}	0	27	26	0.63 (5.49)	—	—	—
7.81×10^{-5}	0	34	39	5.86 (5.10)	—	—	—
1.56×10^{-4}	0	48	57	8.38 (2.29)	-81.03	0.014	0.011
1.56×10^{-4}	-0.05	—	none	6.46	**2.44** (5.80)	0.301 (0.001)	0.17
1.56×10^{-4}	-0.5	—	none	-8.05	-15.18 (17.39)	0.210 (0.024)	0.84
1.56×10^{-4}	-5.0	—	none	-3.26	-5.96 (2.35)	0.529 (0.083)	0.92
IDM-on, $\gamma = 50$	0	none	none	**30.66** (1.05)	**30.66** (1.05)	0.280 (0.016)	0.12

L.6 Time repulsion: a barrier that needs no action labels

The temporal-similarity term $\mathcal{L}_{\mathrm{sim}} = \frac{1}{T-1}\sum_t \|z_{t+1} - z_t\|_2^2$ enters with a signed weight λ_{sim}, so $\lambda_{\mathrm{sim}} < 0$ rewards temporal change; it depends on the joint law of (z_t, z_{t+1}), outside Proposition 1, and needs no action labels. For a scalar feature of the observation $f(o_t)$ at unit variance, let $\varrho(f) = \mathbb{E}\,(f(o_{t+1}) - f(o_t))^2$ be its temporal roughness, the objective of slow feature analysis [86], and $\mathcal{E}(f)$ its irreducible risk. The identity is an admissible predictor, so $\theta(f) = \mathcal{E}(f)/\varrho(f) \in [0, 1]$ is the fraction of the feature's motion no predictor explains away [85]. The marginal cost of encoding f is $\varrho(f)\,(\theta(f) - |\lambda_{\mathrm{sim}}|)$, negative, i.e. preferred to background, exactly when $\theta(f) < |\lambda_{\mathrm{sim}}|$.

Writing θ_F for the ratio θ of a feature reading the foreground x_t alone and θ_D for that of a feature reading the distractors y_t alone, the weights preferring the foreground to the background while still rejecting distractors are those with $\theta_F < |\lambda_{\text{sim}}| < \theta_D$, a window nonempty precisely when $\theta_F < \theta_D$.

In the middle block of Table 12, time repulsion does what no marginal regularizer can: the three repulsion runs keep action sensitivity between 0.21 and 0.53, while the control at $\lambda_{\text{sim}} = 0$ falls to 0.014, so the predictor no longer degenerates to the identity. But no repulsion run clears the zero-action floor, the best reaching $+2.44 \pm 5.80$ where the IDM-on baseline reaches $+30.66 \pm 1.05$. On the healthy checkpoint the trained predictor's one-step error is $\hat{\theta} = 0.51$ of the identity predictor's, so about half of the encoded motion is unpredictable. Taking that as θ_F: an uncorrelated distractor also has $\theta = 1/2$ and a correlated one more, so the window above θ_F is thin at best. The configured weights, 0.057, 0.57 and 5.7 on θ's scale, confirm that it does not select: the run that escapes is the one far below the window, at 0.057, and the run inside it, at 0.57, fails worst. A small repulsion thus acts as a motion floor keeping the predictor off the identity, not as the selector that window suggests. Above $|\lambda_{\text{sim}}| = 1$ the marginal cost $\varrho(f)(\theta(f) - |\lambda_{\text{sim}}|)$ is negative for every feature, including one whose motion is pure noise ($\theta = 1$), so the term rewards any temporal change and strips even the episode identity: χ rises from 0.12 on the healthy parent to 0.84 and 0.92 at the two larger weights.

L.7 The inverse-dynamics term is a barrier of a different order

Proposition 3 (IDM floor)**.** *If z_t is $\mathcal{C}$-measurable at every t then, for any head I,*

$$\mathcal{L}_{\text{idm}} \;\geq\; \frac{1}{d_a}\,\frac{1}{T-1}\sum_{t=1}^{T-1}\left(\operatorname{tr}\operatorname{Var}(a_t) - \operatorname{tr}\operatorname{Var}(\mathbb{E}[a_t \mid \mathcal{C}])\right), \tag{12}$$

a constant fixed by the data alone.

Proof. Under collapse (z_t, z_{t+1}) is $\mathcal{C}$-measurable, so for each t the Bayes risk of regressing a_t on it is at least $\mathbb{E}\operatorname{tr}\operatorname{Var}(a_t \mid \mathcal{C})$; the law of total variance and the $1/d_a$ normalization of Eq. (8) give Eq. (12) term by term. □

With actions normalized to unit variance per dimension, so that $\operatorname{tr}\operatorname{Var}(a_t) = d_a$, and that episode identity does not predict, the floor is 1: at $\gamma = 50$ the collapse costs 50, against a SIGReg barrier Δ that is zero for the grouping we use and at most 10^{-3} for any of them. The floor does not shrink with batch size; it is a statement at exact collapse, and the near-collapsed runs of Table 12 ($\chi \leq 0.016$) are covered by the measurements, not by the bound. It never penalizes keeping distractors [63, 47, 49]. It is the only $\Omega(1)$ barrier here and needs action labels, which is why §4.3 adds it and why SIGReg alone restores rank but not fidelity in Fig. 18.

M Extended related work

M.1 World models and planning

Self-supervised learning by predicting the next sensory state has long been thought central to how animals come to understand the structure of their world [78, 66, 16]. As early as Sutton & Barto [73], AI research has sought agents that build an internal model of the world and use it for planning and control. Modern instantiations learn such models directly from high-dimensional observations, in pixel space or in a learned latent space, and scale to both simulation and real-world robotics; models in this vein such as PlaNet, the Dreamer line, TD-MPC and MuZero [29, 83, 30, 31, 33, 34, 35, 69, 53, 41, 88] plan or learn behavior against a reward or value signal. A distinct robotics line plans directly with action-conditioned video prediction (visual foresight) to control real robots from raw pixels [24, 20]. More recently, task-agnostic generative world models trained on large-scale real-world video produce realistic and diverse simulations of physical environments [11, 89, 10, 59, 87, 19], showing that predictive modeling scales beyond task-specific datasets. Both directions remain coupled to a reward signal or to modeling the pixels themselves.

Joint-embedding predictive architectures (JEPAs) drop both couplings, to pixel reconstruction and to reward, by predicting *latent* targets directly, with no decoder and no reward model [48, 1, 9]. A recent line of action-conditioned approaches plans in the resulting latent space [77, 7, 4, 81, 82, 76, 46, 58],

and they differ mainly in how the encoder is obtained: DINO-WM [92] trains a dynamics model on *frozen* pretrained features; V-JEPA 2 [2] pretrains a video JEPA and then attaches an action-conditioned predictor, Terver et al. [77] unifies the two in the JEPA-WMs framework; and end-to-end variants such as PLDM [72] and LeWM [52] train encoder and predictor together from pixels. H-JEPA builds directly on this end-to-end, action-conditioned setting and extends it from a single latent space to a stack of JEPA world models at increasing temporal abstraction.

M.2 Preventing collapse in non-contrastive SSL

Predicting latent targets risks a trivial *collapsed* solution in which the encoder and predictor emit a constant vector, driving the prediction loss to zero while encoding nothing about the input. Contrastive methods [13, 12] avoid it with negatives; non-contrastive methods instead regularize the embedding distribution: BYOL and SimSiam [26, 14, 65] through architectural asymmetries, VICReg and Barlow Twins [8, 90] through explicit variance/covariance criteria. LeJEPA [6] makes this a distributional target, pushing embeddings toward an isotropic Gaussian via a sketched normality test (SIGReg). H-JEPA applies SIGReg at every level of the hierarchy (§2.1), and we contribute a distributed formulation that is exactly invariant to how a batch is sharded (§K.3). When learning JEPA representations from temporal input sequences (like video in the subsequently mentioned papers), a second collapse mode can arise where the predictor becomes the identity mapping, which we call *slow-feature collapse*, studied first by Sobal et al. [71]. We generalize their result in §L, showing that the mode is the objective's optimum rather than merely one of its minima, that it is unaffected by the choice of marginal anti-collapse term, and which terms do exclude it. In [40, 76, 72], the JEPA encoders receive an additional regularization loss term, an Inverse-Dynamics Modeling (IDM) loss [63], applied to this collapse mode by Sobal et al. [71]. The LeWM+IDM level-1 model of §4.3 is close to PLDM [72] and EB-JEPA [76], the three differing mainly in whether the anti-collapse term is SIGReg or variance-covariance [8], which serve the same purpose.

M.3 Hierarchical World Models

Recent work builds hierarchical world models in which different levels occupy different learned representation spaces. Hierarchical latent-variable video models give each level its own latent and advance the coarser levels only once every several frames, on a fixed stride (Clockwork VAE [67]) or at learned boundaries (VTA [44]), so the top levels come to represent slower-changing content. STREAMER [54] and SUNTA [39] share a common structure: each reconstructs the input at the lowest level but, at higher levels, predicts the *latent* of the level below rather than pixels, segmenting the stream into events and summarizing each into a coarser, lower-detail code. The same hierarchy-of-distinct-representations idea has lately gained momentum in language modeling, via content-based dynamic chunking [61, 38] or a coarse latent predicting several steps ahead [70], though there the hierarchy is built over text tokens rather than perceptual observations of an environment.

None of these methods is action-conditioned or used for planning, and all model the raw input at the lowest level: reconstructing pixels (the video models) or predicting tokens (the language models). H-JEPA differs on both counts: it is latent-predictive at *every* level (even the base is a JEPA trained without reconstruction), and it is *action-conditioned* and used for *planning and control* (§2.2).

M.4 Hierarchical control

Hierarchical control has a long history in reinforcement learning, where the hierarchy is one of *decisions*: a higher-level policy commits to a subgoal or sub-policy that a low-level controller executes over many steps, as in the options framework and feudal approaches [74, 18, 79, 56]. Classical hierarchical model predictive control makes an analogous split for optimal control, pairing a high-level planner over subgoals with a low-level tracking controller [23, 50, 45]. Both organize *control* across time scales: RL methods learn subgoals and controllers by maximizing reward, and classical MPC derives its levels from supplied dynamics models and hand-designed state abstractions.

More recent work brings hierarchy into deep model-based RL, learning the hierarchical world model itself within a reward-driven, task-specific training loop [32, 36, 27, 28, 68]. Setting reward aside, these methods differ in their hierarchy of *representations*. Schiewer et al. [68] learns a separate state-space model per level, though the models are generative and reward-driven and each upper

Table 13: **Model-based hierarchical control methods along four axes that define H-JEPA.** *Pixels*: operates from high-dimensional visual observations rather than a low-dimensional state. *Task-agnostic*: reward-free training with no task signal; the task objective (a goal or a planning cost) is defined only at test time. *Reconstruction Free*: learns representations and dynamics with no reconstruction objective: predicting latent targets rather than decoding observations. *Hierarchy of learned latent spaces*: a stack of *distinct* latent spaces, each a coarser abstraction of the one below that discards detail (vs. a single/shared space, parallel spaces, or a hierarchy of decisions only). ✓ yes, ✗ no, ∼ partial. H-JEPA is the only method satisfying all four; HWM, our baseline, differs on exactly one axis.

Method	Pixels (In)	Task-Agnostic World Models	Reconstruction Free	Hierarchy of Learned Latent Spaces
Classical hierarchical MPC [23, 50, 45]	✗	✓	✗	✗
Director [32]	✓	✗	✗	✗
Puppeteer [36]	✓	✗	✓	✗
SPlaTES [28]	✗	✗	✓	∼
Schiewer et al. [68]	✗	✗	✗	✓
THICK [27]	✓	∼	✗	✗
GCP [64]	✓	✓	✗	✗
HWM [91]	✓	✓	✓	✗
H-JEPA (ours)	✓	✓	✓	✓

latent is only indirectly coupled to the level below. SPlaTES' [28] high-level space is an independent low-dimensional projection of the state; THICK [27] factorizes its latent into fast- and slow-changing components and uses sparse changes in the slow component to define event-scale time steps, all within a single latent space; and Puppeteer's [36] two world models share one grain, bridged by a hand-engineered command interface. In none of them is the hierarchy a stack of *distinct* latent spaces, each recursively derived from the one below at increasing temporal abstraction, as it is in H-JEPA.

Closest to us in setting are pixel-based methods that plan toward a goal zero-shot at test time. GCP [64] recursively predicts visual subgoals, decoding intermediate *images*. Our baseline HWM [91] is a JEPA-based hierarchical world model whose predictors model dynamics over different time horizons and that plans hierarchically at test time, yet all its predictors operate in a single shared latent space rather than a hierarchy of distinct representations.