# LPWM: A CASE FOR SPARSE REPRESENTATIONS IN WORLD MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Joint-embedding predictive architectures (JEPAs) learn latent dynamics for planning and avoid representation collapse by matching features to maximum-entropy distributions such as isotropic Gaussians, yielding dense representations. However, it is unclear if dense representations are the most favorable geometry for modeling dynamics. In this work, we ask whether a different geometry such as sparse representations can make action-conditioned latent dynamics easier to model, and what dynamical structure emerges from such representations. We first show that nonlinear Lipschitz dynamics can be approximated arbitrarily well by action-conditioned linear dynamics in a sufficiently high-dimensional one-hot latent space, with rollout error vanishing as the dimension grows. This motivates distributed sparse representations as a practical relaxation of one-hot sparsity. We introduce LpWorldModel (LpWM), a JEPA model regularized with Rectified Distribution Matching Regularization (RDMReg) to match encoder features to a Rectified Generalized Gaussian distribution, yielding non-negative sparse codes. Empirically, sparsity lowers the predictor complexity required for successful planning: on PushT, sparse LpWM outperforms dense LeWM by up to 57% in planning success at intermediate predictor capacities. This advantage also extends beyond Gaussian distribution matching, with LpWM outperforming dense VICReg representations across multiple predictor families. We further find and provide proven guarantee that the learned sparse representations on piecewise-affine dynamics are mode-factored, with support encoding discrete dynamical regimes and feature magnitudes capturing continuous within-regime state. Together, these results suggest that sparse representations reduce the predictor complexity for control while revealing interpretable structure.

## 1 INTRODUCTION

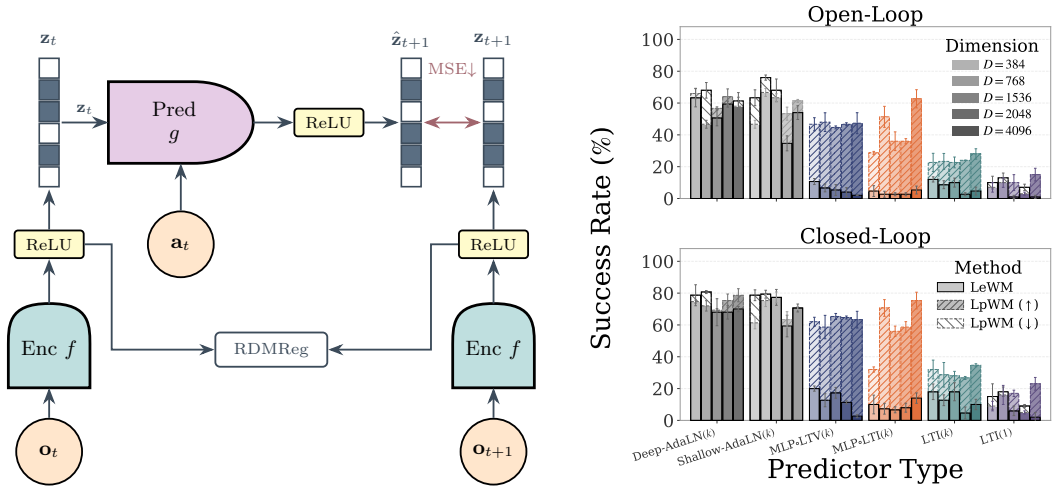
Joint-embedding predictive architectures (JEPA) have emerged as a practical recipe for model-based control from pixels: an encoder maps observations to a latent space, a predictor rolls the latent forward under actions, and planning is carried out by model-predictive control (MPC) in latent space (LeCun et al., 2022). In the case of end-to-end training, where the encoder and predictor are jointly trained, an anti-collapse loss is necessary to prevent the feature representation from degenerating to a constant representation or a low-rank subspace (Jing et al., 2021).

Prior methods such as PLDM (Sobal et al., 2025) use VCReg (Bardes et al., 2022), while LeWM (Maes et al., 2026b) uses SIGReg (Balestriero & LeCun, 2025) to prevent collapse through information maximization, yielding dense representations in which all latent coordinates are nonzero almost surely. While effective for preventing collapse, it remains unclear whether dense features are the most favorable geometry for modeling controlled dynamics.

In this work, we ask whether sparse representations provide a more favorable geometry for modeling action-conditioned dynamics. Our motivation comes from an idealized limit: for Lipschitz controlled dynamics on a compact state space, sufficiently high-dimensional one-hot representations can approximate the dynamics with action-conditioned linear latent dynamics (Proposition 3.1), with rollout error vanishing as the dimension grows (Corollary 3.2). While exact one-hot codes are impractical, this suggests distributed sparse representations, in which a nontrivial fraction of latent coordinates are exactly zero, as a learnable relaxation that may simplify the dynamics.



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.




(a) LpWM Schematic Diagram

(b) PushT Planning Performance.

Figure 1: **LpWM: a JEPA model with sparse representations.** (a) An encoder $f$ maps observation $\mathbf{o}_t$ to non-negative, sparse latents $\mathbf{z}_t$ via a terminal (Rep)ReLU function. The predictor $g$ takes inputs $\mathbf{z}_t$ and actions $\mathbf{a}_t$ to predict $\hat{\mathbf{z}}_{t+1}$, compared against the encoded target $\mathbf{z}_{t+1} = f(\mathbf{o}_{t+1})$. RDMReg (Kuang et al., 2026b) matches per-timestep latent marginals to a Rectified Generalized Gaussian, preventing collapse and inducing sparsity. (b). Arrows indicate the success-rate difference between LpWM and LeWM: ↑ denotes higher performance for LpWM, while ↓ denotes lower performance. LpWM outperforms LeWM with more linear predictor variants (MLP∘LTV($k$), MLP∘LTI($k$), and LTI($k$); see full specifications in Table 1) by up to $57\%$ in success rate, while performing on par with LeWM using the high-capacity DiT predictors. This suggests that sparse representations induce a latent space with simpler dynamics to learn compared to their dense counterparts.

We introduce LpWorldModel (**LpWM**), a JEPA world model regularized with RDMReg (Kuang et al., 2026b) to learn non-negative, sparse codes. RDMReg matches rectified feature distributions to Rectified Generalized Gaussian targets, whose underlying Generalized Gaussian distributions are maximum-entropy under expected $\ell_p$-norm constraints; choosing $p \leq 1$ yields sparse targets.

Empirically, sparsity lowers the predictor complexity required for successful planning: predictors that fail over dense codes succeed over sparse ones. The resulting representations are also mode-factored: their support encodes discrete dynamics regimes, while feature magnitudes encode continuous within-regime state on piecewise-affine dynamics. Together, these results suggest that sparse latent geometry can make dynamics both easier to model and more interpretable.

**Contributions.**

- **LpWorldModel**. We introduce LpWM, a JEPA world model that uses RDMReg to learn non-negative sparse latent representations for latent dynamics modeling and planning.
- **Sparse representations for latent linearization**. We formalize the intuition that sufficiently high-dimensional one-hot representations can approximate nonlinear controlled dynamics with linear latent transitions, motivating distributed sparse codes as a practical relaxation.
- **Lower-complexity latent dynamics**. We show empirically that sparsity reduces the predictor capacity needed for successful planning: while Wall dynamics are already solved by linear predictors, on the harder PushT environment sparse LpWM substantially outperforms dense LeWM with intermediate-capacity predictors, by up to $57\%$ in success rate.
- **Mode-factored latent dynamics**. We empirically find and prove formally that sparse representations organize dynamics into interpretable factors, with support encoding discrete dynamical regimes and feature magnitudes capturing continuous within-regime state.

# 2 METHOD

## 2.1 LPWM: JEPA WITH RECTIFIED DISTRIBUTION MATCHING REGULARIZATION (RDMREG)

Let $\mathbf{z}_{t+1} = f_{\boldsymbol{\theta}}(\mathbf{x}_{t+1}) \in \mathbb{R}^d$ be the encoder representation of the observation $\mathbf{x}_{t+1}$ and $\hat{\mathbf{z}}_{t+1} = g_{\boldsymbol{\phi}}(\mathbf{z}_t, \mathbf{a}_t)$ be the predictor output with action inputs $\mathbf{a}_t \in \mathbb{R}^{d_a}$. The action-conditioned JEPA uses the following loss function to jointly train the encoder $f_{\boldsymbol{\theta}}$ and the predictor $g_{\boldsymbol{\phi}}$:

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \quad \|\hat{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}\|_2 + \lambda_{\mathrm{RDMReg}} \cdot \mathcal{R}(\mathbf{z}_{t+1}) \tag{1}$$

where $\lambda_{\mathrm{RDMReg}}$ is a hyperparameter and $\mathcal{R}(\cdot)$ is an anti-collapse regularizer. We focus on the setting where $\mathcal{R}(\cdot)$ is the Rectified Distribution Matching Regularization (RDMReg) (Kuang et al., 2026b):

$$\mathcal{R}(\mathbf{z}) = \mathbb{E}_{\mathbf{c}\sim\mathrm{Unif}(\mathbb{S}^{d-1}_{\ell_2})}\left[\mathcal{L}(\mathbb{P}_{\mathbf{c}^\top\mathbf{z}} \| \mathbb{P}_{\mathbf{c}^\top\mathbf{y}})\right] \tag{2}$$

where the random projection vector $\mathbf{c} \in \mathbb{R}^d$ is sampled uniformly from the $\ell_2$ unit sphere $\mathbb{S}^{d-1}_{\ell_2} := \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x}\|_2 = 1\}$, the target random vector $\mathbf{y} \sim \prod_{i=1}^d \mathrm{ReLU}(\mathcal{GN}_p(\mu, \sigma))$ follows the Rectified Generalized Gaussian (RGG) distribution, $\mathbb{P}_x$ denotes the probability distribution for the random variable $x$, and $\mathcal{L}(\cdot\|\cdot)$ is a loss function that computes discrepancies between probability distributions. In our case, we follow Kuang et al. (2026b) and set $\mathcal{L}(\cdot\|\cdot)$ to be the 2-Wasserstein distance. By default we choose $\mu = 0$, $\sigma = \sqrt{1/2}$, and $p = 1$ and hence our target distribution reduces to the Rectified Laplace distribution (Kuang et al., 2026b). Note that when $\mu = 0$, $\sigma = 1$, $p = 2$, and in the absence of $\mathrm{ReLU}$, we recover the dense isotropic Gaussian distribution used in LeWM (Maes et al., 2026b).

## 2.2 ARCHITECTURE AND RECTIFICATION

Following Maes et al. (2026b), we design our encoder $f_{\boldsymbol{\theta}}$ as a ViT (Dosovitskiy et al., 2021) with the output CLS token passed through a three-layer MLP projector to produce the embedding $\mathbf{z} \in \mathbb{R}^d$. The predictor $g_{\boldsymbol{\phi}}$ is a Transformer with AdaLN-zero action conditioning (Peebles & Xie, 2023) and a final three-layer MLP projector that predicts the next embedding $\hat{\mathbf{z}}_{t+1}$.

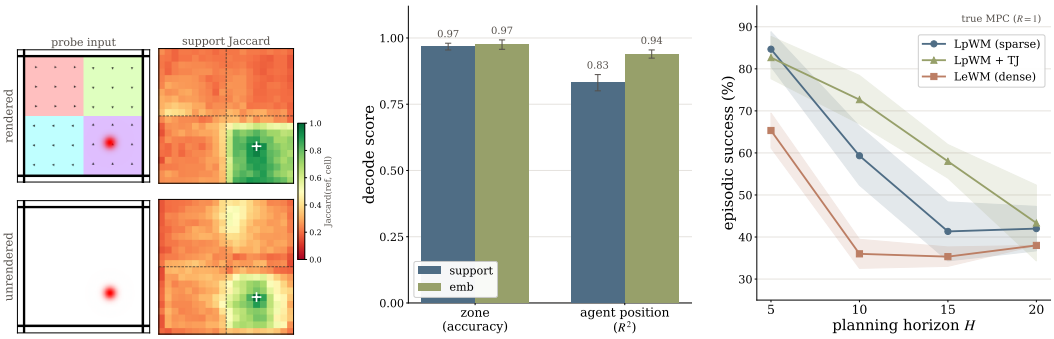
To induce exact zeros, the outputs of both the encoder and predictor MLP projectors pass through a reparameterized ReLU(Wang et al., 2024; Kuang et al., 2026b): $\mathrm{RepReLU}(x) = \mathrm{sg}(\mathrm{ReLU}(x)) + \mathrm{GeLU}(x) - \mathrm{sg}(\mathrm{GeLU}(x))$, where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. In the forward pass, $\mathrm{RepReLU}(x) = \mathrm{ReLU}(x)$, yielding exact zeros, while in the backward pass gradients flow through $\mathrm{GeLU}(x)$ to mitigate the dying-ReLU pathology (Hendrycks & Gimpel, 2023). Importantly, LpWM does not rely on stop-gradient for collapse prevention: standard $\mathrm{ReLU}$ with RDMReg trains without collapse and yields exactly sparse representations (Kuang et al., 2026b). We therefore use $\mathrm{RepReLU}$ as an optimization safeguard against dead neurons rather than as an essential component of the method. See Figure 1a for the LpWM architecture.

## 2.3 PLANNING WITH THE LEARNED WORLD MODEL

At test time the world model serves as the dynamics model for model-predictive control (MPC): we encode the current and goal observation images to $\mathbf{z}_0$ and $\mathbf{z}_g$, and search for an action sequence whose predicted latent rollout minimizes the goal cost $\mathcal{C} = \|\hat{\mathbf{z}}_T - \mathbf{z}_g\|_2$ using CEM (Rubinstein, 1999; Zhou et al., 2025). See additional experimental details in Appendix D.

# 3 DOES SPARSITY SIMPLIFY LATENT DYNAMICS?

In the following sections, we ask whether a sparse code makes the action-conditioned latent dynamics simpler by requiring a less expressive predictor than a dense code. We first show that a sufficiently high-dimensional sparse code renders the latent dynamics exactly linear under appropriate assumptions (Proposition 3.1). We then relax this to a learnable distributed sparse code and test, along a predictor-complexity ladder (Table 1), whether a simpler predictor plans better under sparsity.




(a) Support Jaccard Heatmap. (b) State Decoding. (c) Planning Evaluation.

Figure 2: **The sparse support recovers the piecewise structure of the dynamics. (a)** Support Jaccard heatmap over a $20 \times 20$ grid: the Jaccard index of each cell's support with a fixed reference cell (white $+$) is high within the reference zone and drops across zone boundaries (dashed), for a model trained with rendered zones (top) and one with no visual cues (bottom). **(b)** The binary support decodes the ground-truth zone identity as accurately as the full embedding, while agent position is decoded better from the magnitudes than from the support. **(c)** Planning success vs. horizon (closed-loop, 3-seed mean$\pm$std). Goal locations of the dot agent are sampled randomly. Both sparse variants beat dense LeWM at every horizon.

### 3.1 Linearization of Nonlinear Dynamics with Sparse Representations

In Proposition 3.1, we show that compact controlled systems with Lipschitz dynamics admit finite one-hot sparse encodings with exactly linear latent dynamics. This is closely related to classical symbolic abstractions based on state-space quantization (Pola et al., 2008), which we reinterpret as one-hot sparse latent codes.

**Proposition 3.1** (Linear Approximation of Nonlinear Dynamics with Sparse Representations)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$ be a compact state space and let $\mathcal{A} \subset \mathbb{R}^p$ be a compact continuous action space. Consider a deterministic discrete-time controlled dynamical system $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{a}_t)$ where $f : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$. Assume that $f$ is uniformly Lipschitz in the state variable, i.e. there exists a finite constant $L \geq 0$ such that $\|f(\mathbf{x}, \mathbf{a}) - f(\mathbf{y}, \mathbf{a})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ for every $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and $\mathbf{a} \in \mathcal{A}$. Then, for every $\varepsilon > 0$, there exist 1) a finite latent dimension $N$; 2) a one-hot sparse encoder $E_\varepsilon : \mathcal{X} \to \mathbb{R}^N$; 3) a decoder $D_\varepsilon : \{\mathbf{e}_1, \ldots, \mathbf{e}_N\} \to \mathcal{X}$; and 4) an action-conditioned family of matrices $P_\varepsilon : \mathcal{A} \to \{0, 1\}^{N \times N}$, $\mathbf{a} \mapsto \mathbf{P}_\varepsilon(\mathbf{a})$, such that the latent dynamics $\mathbf{z}_{t+1} = \mathbf{P}_\varepsilon(\mathbf{a}_t)\mathbf{z}_t$ are exactly linear in $\mathbf{z}_t$, and the decoded one-step prediction satisfies*

$$\sup_{\mathbf{x}_t \in \mathcal{X},\, \mathbf{a}_t \in \mathcal{A}} \|f(\mathbf{x}_t, \mathbf{a}_t) - D_\varepsilon(\mathbf{P}_\varepsilon(\mathbf{a}_t) E_\varepsilon(\mathbf{x}_t))\| \leq (L + 1)\varepsilon. \tag{3}$$

*Proof.* See Appendix H.1 □

In Corollary 3.2, we consider one example of a compact domain and show that the rollout error with respect to the ground-truth dynamics vanishes as the encoding dimension grows.

**Corollary 3.2** (Sparsity in High-Dimensions)**.** *Let the state space be $\mathcal{X} = [0, 1]^d$ and consider dividing each coordinate into $n$ equal intervals, producing $N = n^d$ grid cells. We represent each cell by its center with the covering radius $\varepsilon_N = \frac{\sqrt{d}}{2} N^{-1/d}$. Hence the one-step prediction error satisfies*

$$\sup_{\mathbf{x}, \mathbf{a}} \|f(\mathbf{x}, \mathbf{a}) - D_{\varepsilon_N}(\mathbf{P}_{\varepsilon_N}(\mathbf{a}) E_{\varepsilon_N}(\mathbf{x}))\| \leq \frac{\sqrt{d}}{2}(L + 1) N^{-1/d} = O(N^{-1/d}). \tag{4}$$

*For any fixed rollout horizon $H$, $\|\mathbf{x}_H - \hat{\mathbf{x}}_H\| \leq \frac{\sqrt{d}}{2} N^{-1/d} \sum_{k=0}^{H} L^k$ and $\lim_{N \to \infty} \|\mathbf{x}_H - \hat{\mathbf{x}}_H\| = 0$.*

*Proof.* See Appendix H.2. □

Table 1: **Predictors with Different Levels of Complexity.** $\hat{\mathbf{z}}_{t+1}$ is the predicted latent, $\mathbf{z}_{t-i}$ are past latents, and $\mathbf{a}_t$ is the action. AdaLN predictors are Transformer-based models with Adaptive Layer Normalization for action conditioning, i.e. DiT architecture (Peebles & Xie, 2023). The Deep-AdaLN($k$) predictor is the one used in LeWM (Maes et al., 2026b). LTI denotes a linear time-invariant predictor with fixed operators $\mathbf{A}_i, \mathbf{B}$, while LTV denotes a linear time-varying predictor with state-dependent operators $\mathbf{A}_i(\mathbf{z}_t), \mathbf{B}(\mathbf{z}_t)$. We use MLP$\circ$ to denote one extra nonlinear transform. The history $k$ denotes context length, with 1 the single-frame case. The output link is $\sigma = \mathrm{identity}$ for dense LeWM and $\sigma = \mathrm{RepReLU}$ for sparse LpWM. See Appendix B for details.

| Predictor | Layers | History ($k$) | Functional form |
|---|---|---|---|
| **Deep-AdaLN**($k$) | 6 | 3 | $\hat{\mathbf{z}}_{t+1} = \sigma\Big(\mathrm{AdaLN}^{(6)}(\mathbf{z}_{t-k+1:t},\, \mathbf{a}_t)\Big)$ |
| **Shallow-AdaLN**($k$) | 1 | 3 | $\hat{\mathbf{z}}_{t+1} = \sigma\Big(\mathrm{AdaLN}^{(1)}(\mathbf{z}_{t-k+1:t},\, \mathbf{a}_t)\Big)$ |
| **MLP$\circ$LTV**($k$) | 1 | 3 | $\hat{\mathbf{z}}_{t+1} = \sigma\Big(\mathbf{W}\,\mathrm{ReLU}(\sum_{i=0}^{k-1} \mathbf{A}_i(\mathbf{z}_t)\,\mathbf{z}_{t-i} + \mathbf{B}(\mathbf{z}_t)\,\mathbf{a}_t)\Big)$ |
| **MLP$\circ$LTI**($k$) | 1 | 3 | $\hat{\mathbf{z}}_{t+1} = \sigma\Big(\mathbf{W}\,\mathrm{ReLU}(\sum_{i=0}^{k-1} \mathbf{A}_i\,\mathbf{z}_{t-i} + \mathbf{B}\,\mathbf{a}_t)\Big)$ |
| **LTI**($k$) | 1 | 3 | $\hat{\mathbf{z}}_{t+1} = \sigma\Big(\sum_{i=0}^{k-1} \mathbf{A}_i\,\mathbf{z}_{t-i} + \mathbf{B}\,\mathbf{a}_t\Big)$ |
| **LTI**(1) | 1 | 1 | $\hat{\mathbf{z}}_{t+1} = \sigma(\mathbf{A}\,\mathbf{z}_t + \mathbf{B}\,\mathbf{a}_t)$ |

We note that the approximation error of one-step prediction under an exact one-hot sparse encoding decreases as $O(N^{-1/d})$, and therefore suffers from the curse of dimensionality. Nonetheless, Corollary 3.2 illustrates a useful trade-off: higher-dimensional sparse representations can simplify the latent transition map, motivating distributed sparse codes as a practical relaxation that may reduce the complexity of the learned dynamics model. In the following experiments, we test whether this simplification enables successful planning with less expressive predictors. In the special case where the learned latent dynamics reduce further to a standard linear controlled system, classical control methods such as LQR become directly applicable under standard assumptions (Kalman et al., 1960).

### 3.2 Experiment Designs

In practice, exact high-dimensional one-hot encodings are infeasible to optimize, and we need exponentially large feature dimensions to linearize the latent dynamics. LpWM instead learns distributed sparse codes whose sparsity is controlled by the target-distribution parameter $\mu$. Although these codes may not linearize the dynamics exactly, we hypothesize that they make the latent dynamics simpler to model than dense representations, so that lower-capacity predictors can suffice.

To test this, we consider a list of predictor options in Table 1, ranging from nonlinear DiT predictors with AdaLN action conditioning (Peebles & Xie, 2023) to linear time-invariant (LTI) state-space models. We train sparse LpWM and dense LeWM with the same encoder architecture described in Section 2.2, but replace the standard DiT predictor with candidates in Table 1 across latent dimensions $D \in \{384, 768, 1536, 2048, 4096\}$ with their corresponding parameter counts in Table 2.

Additionally, we test a variant of LeWM in which SIGReg (Balestriero & LeCun, 2025) is replaced with VICReg (Bardes et al., 2022), yielding another JEPA model with dense representations. For simplicity, we use VICReg to denote this LeWM variant in Figure 3b.

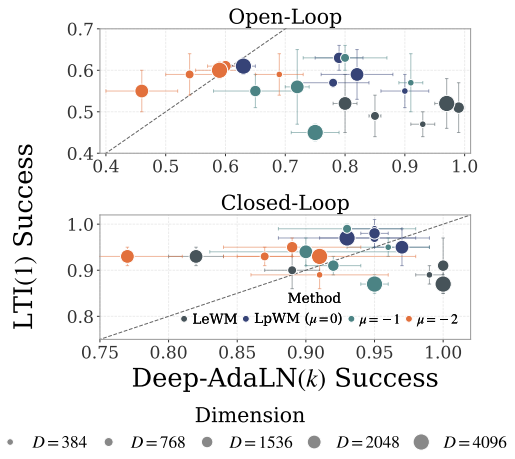
Since our experiment design involves comparing models under width scaling, we use maximum update parameterization ($\mu$P) to stabilize training across widths, with details in Appendix G. Experiments are conducted on Wall and PushT environments (Appendix C).

### 3.3 Results

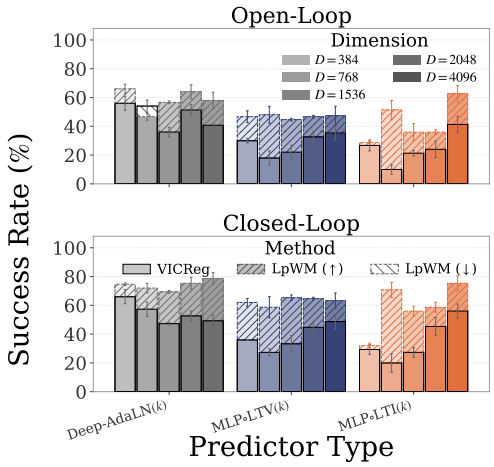
In Figure 3a, we observe that for the Wall environment, even the simplest linear LTI(1) predictor reaches nearly $100\%$ closed-loop CEM success rate for both LeWM and LpWM, leaving little room to distinguish sparse and dense representations.

Table 2: **Predictor parameter counts across latent dimensions $D$.** Counts are for the predictor alone for each $D$ used in our experiments. LTI, MLP∘LTI, and MLP∘LTV are attention-free and far smaller than the AdaLN Transformers.

| Predictor | $D$=384 | $D$=768 | $D$=1536 | $D$=2048 | $D$=4096 |
|---|---|---|---|---|---|
| **Deep-AdaLN**$(k)$ | 25.8M | 62.2M | 166.9M | 260.2M | 822.4M |
| **Shallow-AdaLN**$(k)$ | 5.6M | 13.0M | 33.1M | 50.4M | 151.1M |
| **MLP∘LTV**$(k)$ | 0.81M | 3.10M | 12.1M | 21.4M | 84.7M |
| **MLP∘LTI**$(k)$ | 0.74M | 2.95M | 11.8M | 21.0M | 83.9M |
| **LTI**$(k)$ | 0.59M | 2.36M | 9.44M | 16.8M | 67.1M |
| **LTI**$(1)$ | 0.30M | 1.18M | 4.72M | 8.39M | 33.6M |




(a) Wall Environment. LeWM vs. LpWM



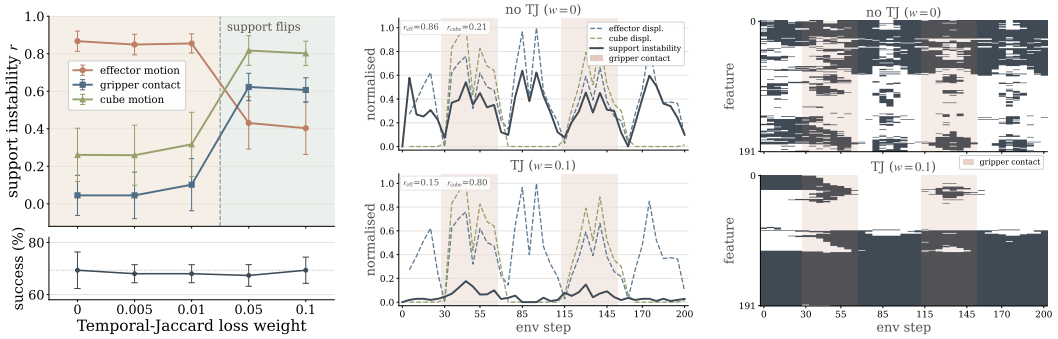

(b) PushT Environment. VICReg vs. LpWM

Figure 3: **A linear predictor is sufficient for the wall environment regardless of dense or sparse embeddings, and LpWM outperforms VICReg-based JEPA across predictor settings in PushT. (a)** We plot the planning success rate of the linear LTI$(1)$ predictor against the nonlinear Deep-AdaLN$(k)$, where each dot denotes one setting of method and feature dimension. For simplicity, we use $\mu = *$ to denote LpWM with the target distribution parameter $\mu$. We show the open-loop evaluation on top and closed-loop results at bottom. Both are run using a CEM planner. We observe that the closed-loop success rate for LTI$(1)$ predictor using SIGReg is already approaching $100\%$, indicating that the wall environment can already be described well by a linear dynamics model in the latent space. (b) We have already shown LeWM vs. LpWM in Figure 1b. Here we compare LpWM with a variant of LeWM where we replaced SIGReg with VICReg in the PushT environment across predictor settings. As opposed to SIGReg, we observe consistent improvement from LpWM over VICReg even for the Deep-AdaLN$(k)$ predictor, and the gap between sparse LpWM and dense methods (i.e. VICReg here) continues to hold for MLP∘LTV$(k)$ and MLP∘LTI$(k)$ predictors. See full hyperparameter sweeps in Appendix I.2.

We therefore turn to the harder PushT environment with results shown in Figure 1b. At the lowest predictor capacity, LTI$(1)$ fails for both LeWM and LpWM. Thus PushT differs from Wall in that the underlying dynamics cannot be modelled well by a linear predictor in practice.

Further, at the highest capacity, Deep-AdaLN$(k)$ and Shallow-AdaLN$(k)$ perform similarly for both representations, indicating that the predictor complexity saturates and we don't observe significant advantages of sparse LpWM against dense LeWM.

However, the difference emerges at intermediate capacity: on PushT, sparse LpWM improves planning success over dense LeWM by $24\%$–$57\%$ with MLP∘LTI$(k)$, $36\%$–$45\%$ with MLP∘LTV$(k)$, and $11\%$–$23\%$ with LTI$(k)$. These performance gaps are verified across extensive hyperparameter




(a) Temporal Jaccard Regularization

(b) Example Episode.

(c) Support Raster Plot.

Figure 4: **Temporal-Jaccard (TJ) makes the cube support track contact, not effector motion.** **(a)** As TJ strength increases, support-instability correlation with effector motion drops ($0.87 \rightarrow 0.40$) while cube-motion ($0.26 \rightarrow 0.80$) and gripper-contact ($0.05 \rightarrow 0.61$) correlations rise, with no change in planning success. **(b)** One example episode: without TJ (top) support instability tracks effector displacement ($r_{\text{eff}}=0.86$); with TJ (bottom) it tracks cube displacement ($r_{\text{cube}}=0.80$), spiking inside the shaded contact windows. **(c)** Support raster: without TJ the support rapidly transitions across frames; with TJ it varies smoothly over time, reorganizing primarily during the shaded contact window.

sweeps for both sparse LpWM and dense LeWM in Appendix I.1. We also report the $\ell_0$ norms of the LpWM embeddings in Table 3.

Additionally, Figure 3b shows that LpWM outperforms VICReg across Deep-AdaLN$(k)$, MLP$\circ$LTV$(k)$, and MLP$\circ$LTI$(k)$ predictors. This indicates that the benefit of sparsity extends beyond comparison with the specific dense isotropic Gaussian targets used in LeWM to more general dense representations. Moreover, LpWM outperforms VICReg even with the high-capacity Deep-AdaLN$(k)$ predictor. We hypothesize that this is because second-order moment constraints in VICReg alone can be insufficient as anti-collapse targets because they leave the target distribution underspecified.

In Appendix E, we present additional evaluations on the PointMaze dataset. Overall, these results show that the benefit of sparsity depends on predictor capacity relative to dynamics complexity. In the simple Wall environment, even linear predictors are sufficient, leaving little room for sparsity to improve performance. In the harder PushT environment, sparsity provides substantial gains at intermediate capacities, but these gains diminish with sufficiently expressive predictors. As the underlying dynamics become more complex, the advantages of sparsity may extend to increasingly expressive predictors, potentially making sparse representations a preferable default over dense representations for modeling latent dynamics. We leave a systematic investigation of this hypothesis to future work.

## 4 Does Sparsity Lead to Interpretable Latent Dynamics?

Section 3 shows that sparse codes can simplify latent prediction. We now ask what structure emerges in these representations. We call a representation mode-factored when its binary support primarily identifies the discrete dynamics regime, while feature magnitudes capture finer within-regime state. We first test whether this structure emerges naturally in Piecewise, where the dynamics mode is known by construction, and then examine whether it extends to OGBench-Cube (Park et al., 2025), where contact provides a natural temporal regime change.

### 4.1 Discovering Piecewise-Affine Dynamics

Piecewise is a 2D navigation environment partitioned into zones with different force fields, yielding piecewise-affine dynamics (Appendix C.2). Empirically, we observe that success rate reaches $84.7\%$

for LpWM versus $65.3\%$ for LeWM with random goals, while performance is saturated for goals sampled from the set of evaluation episodes for both methods (Table 4).

To test whether the sparse support recovers the ground-truth piecewise structure, we discretize the Piecewise environment into a $20 \times 20$ grid, yielding $400$ possible agent locations. For each location, we compute the corresponding embedding and binarize it into a support vector in $\{0,1\}^D$. For two binary support vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^D$, we measure their overlap using the Jaccard index

$$J(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{D} \mathbb{1}_{\mathbf{x}_i=1 \wedge \mathbf{y}_i=1}}{\sum_{i=1}^{D} \mathbb{1}_{\mathbf{x}_i=1 \vee \mathbf{y}_i=1}}, \tag{5}$$

where $\wedge$ and $\vee$ denote the logical "and" and "or" operations. We defer further details and its differentiable relaxation to Appendix F. We then select a reference cell, marked by $+$ in Figure 2a, and compute its Jaccard similarity with every other cell. The resulting similarity is high within the reference zone and drops sharply across zone boundaries (dashed), a pattern that persists even when the zones contain no visual cues (bottom). This indicates that LpWM recovers the underlying dynamics regime from action-conditioned prediction rather than appearance alone, organizing the mode structure into which features are active.

Linear probes further confirm a factorized code (Figure 2b): the binary support decodes the ground-truth zone at near-perfect accuracy for sparse LpWM, as accurately as the full embedding, while the continuous agent position is decoded better from the magnitudes. Therefore, the support carries the discrete zone mode, and the magnitudes carry the agent's within-zone state.

Finally, this structure also accompanies strong long-horizon planning (Figure 2c): sparse LpWM outperforms dense LeWM at every horizon $H$, with the gap widening as $H$ increases.

These results suggest a mode-factored organization: support identifies the discrete dynamical regime, while feature magnitudes retain the continuous within-regime state.

### 4.2 Why Piecewise Dynamics Admit Mode-Factored Latent Structure

We now formalize the mode-factored structure suggested by the Piecewise environment results above. Let $X \subset \mathbb{R}^r$ be an $r$-dimensional compact state space with a finite mode partition $X = \bigsqcup_{q=1}^{m} X_q$, where shared boundary points are assigned according to a fixed rule. Every state $\mathbf{x}_t$ belongs to exactly one mode region $X_q$. For each mode $q$, choose a non-degenerate $r$-simplex $\overline{X_q} \subseteq S_q := \mathrm{conv}(\mathbf{v}_{q,0}, \ldots, \mathbf{v}_{q,r})$, where $\mathrm{conv}(\cdot)$ denotes the convex hull. Let $V_q := [\mathbf{v}_{q,0} \ \cdots \ \mathbf{v}_{q,r}] \in \mathbb{R}^{r \times (r+1)}$ be its vertex matrix, and let $\lambda_q : S_q \to \Delta^r$ denote the corresponding barycentric-coordinate map, where $\Delta^r := \{\boldsymbol{\lambda} \in \mathbb{R}^{r+1}_{\geq 0} : \mathbf{1}_{r+1}^\top \boldsymbol{\lambda} = 1\}$. Thus $V_q \lambda_q(\mathbf{x}) = \mathbf{x}$ for every $\mathbf{x} \in S_q$. Since $\lambda_q$ is affine, write

$$\lambda_q(\mathbf{x}) = R_q \mathbf{x} + \boldsymbol{\rho}_q, \qquad R_q \in \mathbb{R}^{(r+1)\times r}, \quad \boldsymbol{\rho}_q \in \mathbb{R}^{r+1}. \tag{6}$$

Set $N := m(r+1)$. Let $J_q : \mathbb{R}^{r+1} \to \mathbb{R}^N$ denote the canonical injection into the $q$-th coordinate block, and define the encoder by

$$E(\mathbf{x}) := J_q \lambda_q(\mathbf{x}), \qquad \mathbf{x} \in X_q. \tag{7}$$

We denote by $\Lambda_q := \lambda_q(X_q) \subseteq \Delta^r$ the admissible active coordinates for mode $q$, and by $\mathcal{Z} := E(X) \subset \mathbb{R}^N$ the set of valid latent vectors.

**Lemma 4.1** (Existence of Mode-Factored Sparse Coordinates)**.** *For every $\mathbf{x} \in X_q$, the latent vector $E(\mathbf{x})$ satisfies*

$$E(\mathbf{x}) \geq 0, \qquad \mathbf{1}_N^\top E(\mathbf{x}) = 1, \qquad \|E(\mathbf{x})\|_0 \leq r+1. \tag{8}$$

*Moreover,*

$$J_j^\top E(\mathbf{x}) = \begin{cases} \lambda_q(\mathbf{x}), & j = q, \\ \mathbf{0}, & j \neq q, \end{cases} \qquad \mathbf{x} = V_q J_q^\top E(\mathbf{x}). \tag{9}$$

*Hence the unique active block identifies the dynamical mode $q$, while the entries within that block determine the continuous state exactly. In particular, $E$ is injective.*

*Proof.* See Appendix H.3 . □

For the encoder of Lemma 4.1, suppose the physical dynamics follow $f(\mathbf{x}, \mathbf{a}) = A_q\mathbf{x} + B_q\mathbf{a} + \mathbf{c}_q$ on each $X_q$. Let $\boldsymbol{\lambda}_0$ denote the initial active coordinates and $\mathbf{a} = (\mathbf{a}_0, \ldots, \mathbf{a}_{H-1})$ the action sequence. The theorem below provides an interpretable normal form consistent with the empirical structure observed in Piecewise: the support identifies the discrete dynamical mode, while, conditional on a mode sequence, the continuous latent evolution is affine in the initial active coordinates and actions.

**Theorem 4.2** (Mode-factored finite-horizon dynamics)**.** *For every horizon $H \geq 1$ and mode sequence $\sigma = (q_0, \ldots, q_H)$, the encoded trajectory is affine in $(\boldsymbol{\lambda}_0, \mathbf{a})$ on inputs realizing $\sigma$:*

$$\mathbf{z}_t = E(\mathbf{x}_t) = J_{q_t}\big(M_{\sigma,t}\boldsymbol{\lambda}_0 + N_{\sigma,t}\mathbf{a} + \mathbf{d}_{\sigma,t}\big), \qquad 1 \leq t \leq H, \tag{10}$$

*for initial-state and action-response matrices $M_{\sigma,t}, N_{\sigma,t}$ and offsets $\mathbf{d}_{\sigma,t}$. Throughout the rollout, $\mathbf{z}_t \in \mathcal{Z}$ and $\|\mathbf{z}_t\|_0 \leq r + 1$.*

*Proof.* See Appendix H.4. □

**Planning implication.** For fixed initial and goal latents, minimizing $\|\mathbf{z}_H(\mathbf{a}) - \mathbf{z}_g\|_2^2$ over the actions realizing $\sigma$ is a convex quadratic program whenever that feasible set is a nonempty polyhedron. Global planning still requires mode-sequence selection. Please refer to Corollary H.3 in Appendix H.4 for constructions and proofs.

### 4.3 From Navigation to Contact-Rich Dynamics

Does the sparse code also separate distinct phases of the dynamics in contact-rich manipulation? On OGBench-Cube, contact provides a natural regime change between free arm motion and arm–cube interaction. As a temporal analogue of the spatial partition in Piecewise, we measure the support instability $1 - J(\mathbf{z}_t, \mathbf{z}_{t+1})$ between consecutive frames and compute its Pearson correlation with effector motion, cube motion, and gripper contact.

Unlike Piecewise, sparsity alone does not determine which temporal factor is encoded by the support. RDMReg constrains only the per-frame marginal, and without an explicit temporal prior the support tends to follow the dominant fast-varying signal. Accordingly, without temporal regularization, support instability behaves largely as a motion detector: it correlates strongly with effector motion, with $r \approx 0.87$, but only weakly with contact, with $r \approx 0.05$ (Figure 4a).

To test whether the support can instead track slower physical regime changes, we additionally consider an optional temporal Jaccard (TJ) loss that encourages support stability across adjacent frames. As the TJ weight increases, support instability becomes less aligned with effector motion and increasingly aligned with cube motion and contact (Figure 4b); for example, the correlation with cube motion increases from $r_{\text{cube}} = 0.21$ to $0.80$. The support raster in Figure 4c shows the same effect visually.

These results highlight a limitation of vanilla LpWM: sparsity alone may not ensure that support transitions correspond to semantically meaningful dynamical events, and without temporal structure the support can instead behave largely as a motion detector. However, they also suggest that, with an appropriate temporal prior, sparse support can become an interpretable tracker of physical regime changes such as object interaction and contact. We leave the design of temporal regularization for more complex dynamics to future work.

## 5 Conclusion

We introduced LpWM, an action-conditioned JEPA regularized with RDMReg to produce non-negative, exactly sparse latents. Motivated by a one-hot linearization result, we showed empirically that sparsity simplifies the action-conditioned latent dynamics: at a fixed, intermediate predictor capacity, a shallow predictor plans over sparse codes where it fails over dense ones on PushT. We further found and proved that provable guarantee that the sparse code is mode-factored—its support encodes the discrete dynamics regime (94–99% decodable on Piecewise) and, with a temporal-Jaccard prior, tracks contact on OGBench-Cube. Hence sparse representations can reduce the predictor complexity for control while leading to more interpretable dynamics.

### AI USE STATEMENT

In this work, we used generative AI tools to assist with writing proofs, formulating mathematical claims, identifying key ingredients for proving mathematical claims, cleaning and reformatting datasets, implementing methods, and designing or providing feedback on research methodology and experiments. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, propose or refine hypotheses, assist with translation, support qualitative or thematic data analysis, or interpret results. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, and format references. We reviewed all AI-assisted work. We began with our own intuition and proof sketches and used generative AI tools only to complete algebraic details. We also used generative AI tools to port baseline methods into the codebase, and we verified the resulting implementations. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

## REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023. URL https://arxiv.org/abs/2301.08243.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL https://arxiv.org/abs/2506.09985.

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

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning, 2023. URL https://arxiv.org/abs/2304.12210.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2022. URL https://arxiv.org/abs/2105.04906.

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, 2024. URL https://arxiv.org/abs/2404.08471.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

Deep Chakraborty, Yann LeCun, Tim G. J. Rudner, and Erik Learned-Miller. Improving pre-trained self-supervised embeddings through effective entropy maximization, 2025. URL https://arxiv.org/abs/2411.15931.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. URL https://arxiv.org/abs/2002.05709.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning, 2021. URL https://arxiv.org/abs/2007.06346.

Raktim Gautam Goswami, Prashanth Krishnamurthy, and Farshad Khorrami. Data-driven deep learning based feedback linearization of systems with unknown dynamics. In *2023 American Control Conference (ACC)*, pp. 66–71. IEEE, 2023.

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, 2020. URL https://arxiv.org/abs/2006.07733.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2022. URL https://arxiv.org/abs/2010.02193.

Olivier J Henaff, Robbe LT Goris, and Eero P Simoncelli. Perceptual straightening of natural videos. *Nature neuroscience*, 22(6):984–991, 2019.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jiaming Hu, Yan Zheng, and Tian Wang. Scale: State-calibrated latent embeddings for jepa planning in the right geometry, 2026. URL https://arxiv.org/abs/2608.16287.

Arnav Kumar Jain, Shivakanth Sujit, Shruti Joshi, Vincent Michalski, Danijar Hafner, and Samira Ebrahimi-Kahou. Learning robust dynamics through variational sparse gating, 2022. URL https://arxiv.org/abs/2210.11698.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*, 2021.

Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5 (2):102–119, 1960.

Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

Yilun Kuang, Yash Dagade, Deep Chakraborty, Erik Learned-Miller, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Radial-vcreg: More informative representation learning through radial gaussianization, 2026a. URL https://arxiv.org/abs/2602.14272.

Yilun Kuang, Yash Dagade, Tim G. J. Rudner, Randall Balestriero, and Yann LeCun. Rectified lpjepa: Joint-embedding predictive architectures with sparse and maximum-entropy representations, 2026b. URL https://arxiv.org/abs/2602.01456.

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

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-François Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, December 2018. ISSN 0893-6080. doi: 10.1016/j.neunet.2018.07.006. URL http://dx.doi.org/10.1016/j.neunet.2018.07.006.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation, 2026a. URL https://arxiv.org/abs/2605.21800.

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

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl, 2025. URL https://arxiv.org/abs/2410.20092.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Giordano Pola, Antoine Girard, and Paulo Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

Mikuláš Ružička. Anwendung mathematisch-statistischer methoden in der geobotanik: Synthetische bearbeitung von aufnahmen. *Biológia, Bratislava*, 13:647–661, 1958.

Xu Shang, Masih Haseli, Jorge Cortés, and Yang Zheng. On the existence of koopman linear embeddings for controlled nonlinear systems. *arXiv preprint arXiv:2602.14537*, 2026.

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, 2025. URL https://arxiv.org/abs/2502.14819.

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, 2026. URL https://arxiv.org/abs/2602.03604.

Yifei Wang, Qi Zhang, Yaoyu Guo, and Yisen Wang. Non-negative contrastive learning, 2024. URL https://arxiv.org/abs/2403.12459.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. Adajepa: An adaptive latent world model, 2026a. URL https://arxiv.org/abs/2606.32026.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning, 2026b. URL https://arxiv.org/abs/2603.12231.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer, 2022. URL https://arxiv.org/abs/2203.03466.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning, 2024. URL https://arxiv.org/abs/2310.17813.

Thomas Yerxa, Yilun Kuang, Eero Simoncelli, and SueYeon Chung. Learning efficient coding of natural images with maximum manifold capacity representations, 2023. URL https://arxiv.org/abs/2303.03307.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021. URL https://arxiv.org/abs/2103.03230.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, and Nicolas Ballas. Hierarchical planning with latent world models, 2026. URL https://arxiv.org/abs/2604.03208.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2025. URL https://arxiv.org/abs/2411.04983.

# A Related Work

**Latent world models for planning.** Joint-Embedding Predictive Architectures (JEPA) predict in a learned latent space rather than reconstructing pixels (Assran et al., 2023; Bardes et al., 2024). Equipped with action conditioning, JEPA-style world models can be viewed as a latent dynamics model of the environment and can be used for planning with model predictive control (Assran et al., 2025; Zhou et al., 2025; Sobal et al., 2025; Maes et al., 2026b). Hierarchical JEPA has also been shown to outperform standard flat JEPA models in planning performance (Zhang et al., 2026).

**Feature Regularization for Latent World Models.** JEPA-style world models plan in the latent space, and hence proper regularizations of the embedding matters for both anti-collapse and also better planning performances. Preventing collapse without negative examples (Chen et al., 2020) is the central design problem of non-contrastive SSL (Balestriero et al., 2023). The stop-gradient and EMA techniques (Grill et al., 2020) have been used in DINO-WM (Zhou et al., 2025), V-JEPA 2 AC (Assran et al., 2025) and related work. More recently, information maximization based techniques have also been proposed, which aims at reducing feature redundancies (Zbontar et al., 2021; Ermolov et al., 2021; Yerxa et al., 2023) and maximizing feature entropy through distribution-matching regularizations (Chakraborty et al., 2025; Balestriero & LeCun, 2025; Kuang et al., 2026a;b).

Beyond collapse preventions, other techniques have been proposed to improve the latent space for planning and control. Wang et al. (2026b) show that adding a temporal straightening loss (Henaff et al., 2019) reduces the curvature of the embeddings across time and hence enables better planning. SCALE (Hu et al., 2026) aligns latent distances with task-relevant state differences to make the planner-facing geometry more informative. Wang et al. (2026a) also considers adapting the final layer features online to improve planning performance and achieves better out-of distribution generalization. It's also helpful to use inverse-dynamics modeling (Lesort et al., 2018), temporal smoothness (Sobal et al., 2025), which can lead to better downstream planning performances (Terver et al., 2026).

**Sparse and Discrete Representations in World Models.** Discrete latent representations have previously been explored in world models. DreamerV2 (Hafner et al., 2022) replaces Gaussian stochastic states with a product of categorical variables, which can equivalently be viewed as a structured sparse binary representation, and finds that categorical latents substantially improve performance over Gaussian ones. The authors hypothesize that sparsity may contribute to this improvement and that categorical states may better capture non-smooth transitions. Relatedly, Variational Sparse Gating (Jain et al., 2022) introduces sparsity into latent dynamics by sparsely updating recurrent-state dimensions through stochastic binary gates. In contrast, we study sparsity directly as a geometry for continuous latent representations: LpWM learns distributed, non-negative sparse codes whose support is not fixed by a categorical grouping. Our focus is also different in that we ask whether sparsity reduces the complexity of the action-conditioned transition model itself, and whether the learned support recovers discrete dynamical regimes while the

**Linearizing Nonlinear Dynamics.** Koopman operator theory studies nonlinear autonomous systems through lifted observables with linear evolution, with extensions to controlled systems enabling linear prediction and MPC (Proctor et al., 2016; Korda & Mezić, 2018). Recent work further characterizes the conditions for which exact finite-dimensional Koopman embeddings exist for controlled nonlinear systems (Shang et al., 2026). Related approaches include SINDy, which identifies sparse nonlinear governing equations from candidate function libraries (Brunton et al., 2016). Data-driven feedback linearization also learns state and input transformations of yielding linear controlled dynamics (Goswami et al., 2023).

# B Predictor Designs

## B.1 Overall Designs

Table 1 lists the predictors we compare, in order of decreasing complexity. **Deep-** and **Shallow-AdaLN**$(k)$ are 6- and 1-block Transformers with AdaLN-zero action conditioning (Peebles & Xie, 2023); **LTI**$(k)$ and **LTI**$(1)$ are linear (time-invariant) state-space maps over $k$ and 1 history lags; and **MLP**$\circ$**LTI**$(k)$ places a single nonlinear readout $\mathbf{W}\,\mathrm{ReLU}(\cdot)$ on top of the linear core. One caveat is

Table 3: **Sparsity of the LpWM encoder representations across predictor types and latent dimensions.** Each entry is the fraction of non-zero coordinates $\mathbb{E}[\|\mathbf{z}\|_0]/D$ in the sparse code for the target distribution with $\{\mu = 0, \sigma = \sqrt{1/2}, p = 1\}$, measured on the validation set at the tuned-best cell for the LpWM run paired with each predictor at latent dimension $D$. Dense LeWM has active fraction 1.0 by construction. The learned code stays $\sim$ 30–65% active regardless of predictor or dimension, confirming the codes underlying the planning comparison are genuinely sparse.

| $D$ | **Deep-AdaLN**$(k)$ | **Shallow-AdaLN**$(k)$ | **MLP∘LTV**$(k)$ | **MLP∘LTI**$(k)$ | **LTI**$(k)$ | **LTI**$(1)$ |
|---|---|---|---|---|---|---|
| 384 | 0.34 | 0.33 | 0.37 | 0.28 | 0.41 | 0.63 |
| 768 | 0.44 | 0.37 | 0.54 | 0.53 | 0.41 | 0.63 |
| 1536 | 0.49 | 0.39 | 0.49 | 0.54 | 0.41 | 0.60 |
| 2048 | 0.42 | 0.46 | 0.51 | 0.46 | 0.43 | 0.61 |
| 4096 | 0.48 | 0.49 | 0.47 | 0.43 | 0.44 | 0.56 |

Table 4: **Success Rate (%) over 50 Test Samples in the Piecewise Environments with Varying Numbers of Zones**. Piecewise $(n \times n)$ denotes the environment with varying numbers of zones. Goal images are either sampled from the evaluation dataset or randomly sampled from all possible locations of the dot agent. $H$ denotes the horizon and $R$ represents the receding-horizon. Results are shown as mean±std over 3 seeds. **Bold** denotes the best performance.

| | | Goals from Evaluation Set | Random Goals | |
|---|---|---|---|---|
| Env | Method | $H = R = 5$ | $H = 5, R = 1$ | $H = R = 5$ |
| Piecewise $(2 \times 2)$ | LpWM (Ours) | **99.33±1.15** | **84.67±4.16** | 59.33±2.31 |
| | LpWM+TJ (Ours) | **99.33±1.15** | 82.67±5.03 | **68.67±4.62** |
| | LeWM | **99.33±1.15** | 65.33±4.16 | 36.00±3.46 |
| Piecewise $(3 \times 3)$ | LpWM (Ours) | **100.00±0.00** | 58.67±1.15 | 56.00±4.00 |
| | LpWM+TJ (Ours) | **100.00±0.00** | **60.00±2.00** | **58.00±5.29** |
| | LeWM | **100.00±0.00** | 50.00±5.29 | 47.33±6.11 |

that we use the $\mathrm{RepReLU}$ link function in LpWM when using **LTI**$(k)$ and **LTI**$(1)$ predictors, so we don't have exact linear dynamics models.

We have also designed **MLP∘LTV**$(k)$, a data-dependent (linear-time-varying) predictor that moves beyond the fixed weight matrices in the linear-time invariant case.

### B.2 MLP∘LTV$(k)$

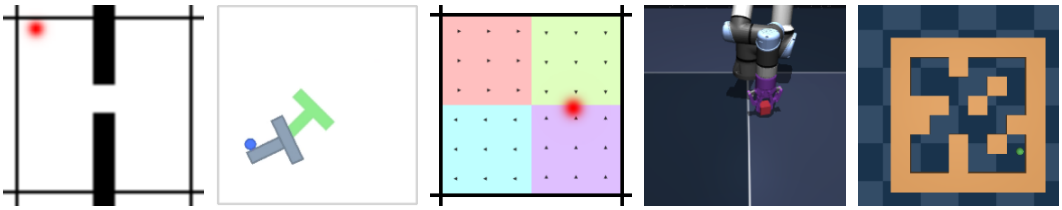
**MLP∘LTV**$(k)$ retains the MLP∘LTI form $\hat{\mathbf{z}}_{t+1} = \sigma\big(\mathbf{W}\,\mathrm{ReLU}(\sum_{i=0}^{k-1} \mathbf{A}_i(\mathbf{z}_t)\,\mathbf{z}_{t-i} + \mathbf{B}(\mathbf{z}_t)\,\mathbf{a}_t)\big)$ but makes the operator matrices state-dependent. Naively, designing a linear map with a matrix-valued output given a vector-valued input will require a $O(D^3)$ tensor for the feature dimension $D$.

Instead, we consider a structured matrix parameterization, where the state emits only a small gate that modulates a fixed low-rank correction:

$$\mathbf{A}_i(\mathbf{z}_t) \;=\; \mathbf{A}_i \;+\; \mathbf{U}_i\,\mathrm{diag}\big(\mathbf{g}_i(\mathbf{z}_t)\big)\,\mathbf{V}_i^\top, \qquad \mathbf{g}(\mathbf{z}_t) = \mathrm{sigmoid}(\mathbf{G}\,\mathbf{z}_t) \in [0,1]^r, \tag{11}$$

where $\mathbf{A}_i$ is a fixed base operator, $\mathbf{V}_i^\top : \mathbb{R}^D \to \mathbb{R}^r$ and $\mathbf{U}_i : \mathbb{R}^r \to \mathbb{R}^D$ are learned low-rank factors, and the gate $\mathbf{g}(\mathbf{z}_t)$—a single projection $\mathbf{G} : \mathbb{R}^D \to \mathbb{R}^{(k+1)r}$ shared across the $k$ lags and the action term—selects among $r$ fixed low-rank "modes" ($r$=16 in our experiments). The action operator $\mathbf{B}(\mathbf{z}_t)$ is gated identically. This costs $O(Dr)$ with no $D^3$ term. Crucially, only $\mathbf{z}_t$ drives the gate, so the action $\mathbf{a}_t$ still enters affinely: the dynamics core stays control-affine up to the shared nonlinear readout.

We zero-initialize the up-projections $\mathbf{U}_i$ (LoRA-style (Hu et al., 2021)) as an override after $\mu$P initialization, so at initialization every correction vanishes and MLP∘LTV$(k)$ coincides exactly with MLP∘LTI$(k)$. It is therefore a strict superset that recovers the fixed-operator model unless the data reward state-dependence.



(a) Wall (b) PushT (c) Piecewise (d) OGBench-Cube (e) PointMaze

Figure 5: **Environments. (a)** Wall: 2D navigation through a doorway (red agent). **(b)** PushT: push a T-shaped block to a target pose (green). **(c)** Piecewise: an open room partitioned into force-field zones (colored) with a known ground-truth segmentation. **(d)** OGBench-Cube: a robot arm manipulating a cube. **(e)** PointMaze-Medium: a green dot agent navigates in a fixed-layout MuJoCo maze, shown as a native $224 \times 224$ RGB observation.

# C Environment

## C.1 TwoRoom, PushT, and OGBench Cube

We use the saved dataset in Zhou et al. (2025) for TwoRoom, PushT, and in Maes et al. (2026a) for OGBench Cube. During planning, we fix the goal location to be 25 steps away from the starting location and select episodes from the evaluation set. The action block size is fixed to be 5.

## C.2 Piecewise

**Piecewise.** To measure what the sparse code encodes about the dynamics, we use a synthetic, analytically-known environment called Piecewise (Maes et al., 2026b). An open $224 \times 224$ room (no walls) is divided into a $g \times g$ grid of zones. Each zone $i$ applies a fixed additive bias to motion, $\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{a}_t \cdot \text{speed} + \mathbf{b}[\text{zone}(\mathbf{p}_t)]$, with radial biases $\mathbf{b}_i = \text{bias_scale} \cdot (\cos\theta_i, \sin\theta_i)$, $\theta_i = 2\pi i/g^2$. The dynamics are thus locally linear but globally piecewise-affine, with a known ground-truth zone segmentation and an exact expert policy (it inverts the motion equation). This is a controlled environment where the latent mode is known, so we can test directly whether the sparse support recovers it. We use $g \in \{2, 3\}$ (i.e. 4 and 9 zones) in our experiments.

## C.3 PointMaze

**PointMaze-Medium and Diverse PointMaze.** Both are image-based MuJoCo navigation tasks with two-dimensional controls. We reuse AdaJEPA's environment and evaluation code (Wang et al., 2026a). Medium has one fixed maze with 26 traversable cells and $4{,}000$ released 100-step trajectories, whose saved states we render at $224 \times 224$ (a $3{,}600/400$ trajectory-level train/validation split).

For Diverse, we generate $2{,}000$ 100-step trajectories on each of 25 connected random $8 \times 8$ layouts using HWM-PLDM's map generator and collection policy (Zhang et al., 2026): random actions have uniform direction and magnitude in $[0, 1]$ and are repeated four environment steps per recorded transition. This reconstructs the $50{,}000$-trajectory training corpus. We use AdaJEPA's five released held-out layouts, with 250 validation trajectories, separately from the 50 planning tasks.

## C.4 Visualization of Environments

See Figure 5 for example snapshots of the environments. Note that there is another version of the Piecewise environment without the background color rendering as we have shown in Figure 2a.

# D Experimental Details

Our experiments comprise two complementary studies implemented in separate experimental pipelines. Section 3 systematically varies predictor complexity and latent dimension to test whether sparsity reduces the complexity of the learned dynamics in Wall and PushT environments. The

Table 5: **Experimental configurations for Section 3.** Wall, PushT, and the two PointMaze environments use the same row definitions.

| | Wall | PushT | PointMaze-Medium | Diverse PointMaze |
|---|---|---|---|---|
| Encoder | from-scratch ViT, 12 layers, width 384 | | from-scratch ViT, 12 layers, width 384 | |
| Predictor | See Table 1 | | Deep-AdaLN($k$) | |
| Dimension ($D$) | $\{384, 768, 1536, 2048, 4096\}$ | | 384 | 384 |
| History $k$ | 1 | 3 | 3 | 3 |
| Frameskip | 5 | 5 | 5 | 5 |
| Batch size | 128 | 64 | 128 | 128 |
| Epochs | 20 | 2 | 20 | 3 |
| Optimizer | Adam, $\mu$P (with base width 384) | | AdamW, $\mu$P (with base width 384) | |
| Base LR | See Appendix I | | See Appendix I.3 | |
| Grad. clip | 0.0 | | 0.0 | 0.0 |
| Reg. weight $\lambda$ | See Appendix I | | See Appendix I.3 | |
| Number of projections | 8192 | | 8192 | 8192 |
| Parameter $\mu$ | $\{0, -1, -2\}$ | | 0 | 0 |
| Temporal Jaccard | off | off | off | off |

Table 6: **Experimental configurations for Section 4.**

| | Piecewise | OGBench-Cube |
|---|---|---|
| Encoder | from-scratch ViT-Tiny, width 192 | |
| Predictor | Deep-AdaLN($k$) | |
| Dimension ($D$) | 192 | |
| History $k$ | 3 | 3 |
| Frameskip | 5 | 5 |
| Batch size | 128 | 128 |
| Epochs | 10 | 10 |
| Optimizer | AdamW | |
| Base LR | $5 \times 10^{-5}$ | |
| Grad. clip | 1.0 | |
| Reg. weight $\lambda$ | 25 | 10 |
| Number of projections | 1024 | |
| Parameter $\mu$ | 0 | 0 |
| Temporal Jaccard | $\{0.0, 0.01\}$ | $\{0.0, 0.005, 0.01, 0.05, 0.1\}$ |

interpretability study in Section 4 instead fixes the model configuration and probes the structure of the learned sparse support in the Piecewise and OGBench-Cube environments. The two pipelines instantiate the same underlying LpWM formulation but differ in architecture and optimization details, summarized in Table 5 and Table 6. All comparisons and ablations are performed within the corresponding experimental pipeline.

In both pipelines, we follow the settings in Section 2.1 and Section 2.2 for the loss, architecture, and link function designs. The model is an action-conditioned JEPA trained end-to-end. The encoder and predictor outputs pass through a link function $\sigma$, and per-timestep latent marginals are matched by 2-Wasserstein distance to a target distribution obtained by applying the same link to a Generalized Gaussian with shape parameter $p$ and location or mean-shift parameter $\mu$. Dense LeWM uses $\sigma = \mathrm{identity}$ with an isotropic Gaussian target ($p = 2$), whereas sparse LpWM uses $\sigma = (\mathrm{Rep})\mathrm{ReLU}$ with a Rectified Laplace target ($p = 1$), yielding sparse representations.

**Experimental Details for Section 3.** We follow the settings in DINO-WM (Zhou et al., 2025) and set the context length ($k$) for the Wall environment to be $k = 1$. PushT uses $k = 3$ because velocity is not recoverable from a single observation. For each predictor type and latent feature dimension configuration, we sweep the RDMReg weight $\lambda_{\mathrm{RDMReg}}$ and the $\mu$P base learning rate $\ell_{\mathrm{muP}}$(Appendix I) and report the best-performing cell in the corresponding grid. We also swept the

hyperparameter for LeWM and its VICReg variants in Appendix I to make sure we obtain strong baseline performances.

**PointMaze.** Both environments use a ViT encoder trained from scratch (12 layers, width 384, patch size 14, three heads, MLP width 768) and Deep-AdaLN($k$) predictor (six layers, 16 heads, MLP width 2048, dropout 0.1). We use $D = 384$, history $k = 3$, and stride five, concatenating five controls per model transition. Images are scaled to $[-1, 1]$ without augmentation. AdamW uses $\mu$P base width 384, weight decay 0.01, and a constant training learning rate. LeWM uses an identity link and Gaussian target ($p = 2$). LpWM uses RepReLU and a rectified Laplace target ($p = 1, \mu = 0$). Temporal Jaccard and auxiliary variance/covariance penalties are disabled.

**Experimental Details for Section 4.** We follow the settings in stable-worldmodel (Maes et al., 2026a). For both the Piecewise and OGBench-Cube environments, we use a from-scratch ViT-Tiny encoder (width 192, CLS token), latent dimension $D = 192$, and a depth-6 Deep-AdaLN($k$) predictor with history $k = 3$. Models are trained for 10 epochs using AdamW with learning rate $5 \times 10^{-5}$, weight decay $10^{-3}$, a linear-warmup cosine schedule with warmup over approximately 1% of the optimization steps, batch size 128, bf16 precision, and gradient clipping at 1.0. These experiments do not use $\mu$P since the feature dimension is relatively small, i.e. $D = 192$.

**Planning and Evaluation.** The Piecewise, Wall, PushT, and OGBench-Cube experiments use the cross-entropy method CEM to optimize action sequences in latent space by minimizing the MSE between predicted latent rollouts and the encoded goal representation. We use 300 candidate action sequences per CEM iteration, retain the top 30 best-performing action sequences, and run CEM for 30 iterations with an initial variance scale of 1.0 and planning horizon $H = 5$. With a frameskip of 5, the goal observation is therefore 25 raw environment steps ahead.

In Section 3, we report both open-loop and closed-loop planning success rates. Open-loop evaluation executes a single CEM plan without replanning and therefore provides a more direct measure of accumulated model-prediction quality. Closed-loop evaluation uses receding-horizon MPC, replanning after each block of 5 executed actions, with at most 10 replanning iterations. Wall and PushT are evaluated over 50 trajectories whose goals are drawn from the evaluation dataset.

For PointMaze, we follow AdaJEPA's frozen-model protocols (Wang et al., 2026a, Appendices A.3–A.4). PiontMaze Medium samples distinct start–goal cells at least three cell units apart in Euclidean distance. Diverse samples starts uniformly from open cells and goals at BFS distance 3–5, with ten tasks per held-out layout. MPC plans five model transitions at stride five, executes five controls, and replans at most 20 times. GD uses 100 Adam steps per replan, zero initialization, and learning rate 0.1 cosine-decayed to zero. CEM uses 200 candidates, 30 elites, ten iterations, and initial variance scale 1. Both minimize latent goal MSE. Medium bounds optimizer actions to $[-1, 1]$. Diverse preserves AdaJEPA's unconstrained optimizer with simulator clipping. Success requires position error below 0.5 world units.

In Section 4, we additionally evaluate Piecewise using randomly sampled goal images, since performance for both LeWM and LpWM saturates when goals are sampled from the evaluation set. In Table 4, we also consider a receding horizon of 1. For OGBench-Cube, we follow the evaluation protocol of Maes et al. (2026b), where goal observations are sampled only 25 raw environment steps ahead. This short horizon can make the planning task relatively easy, potentially leading to performance saturation and limiting the discriminative power of the benchmark. We leave evaluation under more challenging longer-horizon and larger-frameskip settings to future work.

## E Point-Maze Result

We additionally compare dense LeWM and our sparse LpWM with the same Deep-AdaLN($k$) predictor over both PointMaze-Medium and Diverse PointMaze. At the selected configurations, LpWM achieves similar planning success to LeWM while activating approximately 58% and 47% of latent coordinates, respectively, versus 100% for LeWM (Table 7). These results show that comparable planning performance can be retained with fewer active coordinates. We also visualize the planning trajectories in Figures 15–16.

Table 7: **PointMaze planning success and representation sparsity.** Results are mean ± sample standard deviation across three independent training seeds. Planning uses 50 shared tasks per environment. Active is the percentage of exactly nonzero latent coordinates, averaged over validation batches at the final epoch.

| Environment | Method | GD (%) | CEM (%) | Active (%) |
|---|---|---|---|---|
| Medium | LeWM | $92.7 \pm 6.1$ | $93.3 \pm 1.2$ | $100.0 \pm 0.0$ |
| | LpWM | $97.3 \pm 1.2$ | $94.7 \pm 3.1$ | $57.0 \pm 0.9$ |
| Diverse | LeWM | $52.7 \pm 2.3$ | $49.3 \pm 4.2$ | $100.0 \pm 0.0$ |
| | LpWM | $54.0 \pm 2.0$ | $42.7 \pm 5.0$ | $46.5 \pm 1.2$ |

# F JACCARD INDEX

## F.1 HARD AND SOFT JACCARD INDEX

The standard Jaccard index between two sets $A$ and $B$ is $\mathrm{Jac}(A, B) = |A \cap B|/|A \cup B|$. For two binary vectors $\mathbf{x}, \mathbf{y} \in \{0, 1\}^D$ this is $J(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{D} \mathbb{1}_{\mathbf{x}_i=1 \wedge \mathbf{y}_i=1} \,/\, \sum_{i=1}^{D} \mathbb{1}_{\mathbf{x}_i=1 \vee \mathbf{y}_i=1}$, where $\wedge$ and $\vee$ denote the logical "and", "or" operations. Because the encoder's output is non-negative but not binary, directly optimizing this set Jaccard is non-differentiable. We therefore use the soft relaxation (Ružička, 1958), for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D_{\geq 0}$,

$$J_S(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{D} \min(\mathbf{x}_i, \mathbf{y}_i)}{\sum_{i=1}^{D} \max(\mathbf{x}_i, \mathbf{y}_i) + \epsilon}, \tag{12}$$

where $\min(\cdot, \cdot)$ and $\max(\cdot, \cdot)$ are smooth proxies for $\wedge$ and $\vee$; a higher $J_S$ means more similar supports, and $J_S$ recovers the exact set Jaccard when its arguments are binary.

## F.2 TEMPORAL JACCARD LOSS

Enforcing the feature distribution to be sparse independently across time steps via RDMReg imposes no constraints on the desired temporal transition. Therefore we also consider the optional temporal Jaccard loss (TJ) as a slowness prior on the binary support vector. The temporal Jaccard loss penalizes rapid support transitions between consecutive frames and hence encourages the support to be smoothly varying across time while not interfering with the global sparsity constraints brought by RDMReg.

**Temporal Jaccard loss.** Given encoder embeddings $\mathbf{Z} \in \mathbb{R}^{B \times T \times D}_{\geq 0}$, where $B$ is batch size, $T$ is the temporal dimension, and $D$ is the feature dimension, the temporal jaccard loss is the mean support instability over all consecutive frame pairs:

$$\mathcal{L}_{\mathrm{TJ}} = \frac{1}{B(T-1)} \sum_{b=1}^{B} \sum_{t=1}^{T-1} \Big(1 - J_S\big(\mathbf{Z}[b, t, :], \mathbf{Z}[b, t+1, :]\big)\Big) \tag{13}$$

where $\mathbf{Z}[b, t, :] \in \mathbb{R}^D$ denotes the slice of the embedding tensor at batch index $b$ and time index $t$. Minimizing $\mathcal{L}_{\mathrm{TJ}}$ encourages the support between embeddings of consecutive frames to be smoothly varying, and hence reduces the chance of drastic support changes between temporally adjacent frames.

# G MAXIMAL-UPDATE PARAMETERIZATION ($\mu$P) FOR JEPA

Our predictor-complexity experiments vary the latent dimensions over more than an order of magnitude, from $D = 384$ to $D = 4096$. A naive parameterization uses one learning rate for all layers in the neural network as $D$ varies, so performance differences across widths can be confounded by width-specific optimization effects rather than properties of the learned representation. We therefore use maximal-update parameterization ($\mu$P) (Yang et al., 2022) to preserve feature-learning dynamics across width and enable learning-rate transfer.

For a dense linear map $\mathbf{h} = \mathbf{W}\mathbf{x}$ with $\mathbf{W} \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$, $\mu$P chooses initialization and learning-rate scalings such that both the hidden activations and their training-induced updates remain of order

$\Theta(\sqrt{d_{\text{out}}})$ as width changes. Equivalently, the weight and update operators remain at the appropriate spectral scale for stable feature learning (Yang et al., 2024). For Adam-type optimizers, this gives an effective learning-rate scaling proportional to the inverse fan-in,

$$\eta_{\mathbf{W}} \propto \frac{1}{d_{\text{in}}}. \tag{14}$$

We apply this rule to all width-dependent dense maps in the JEPA encoder and predictor. Let $D_0$ denote the reference latent width at which a base learning rate $\eta_{\text{base}}$ is selected. In our experiment, we fix $D_0 = 384$. For parameter matrices whose fan-in scales linearly with the latent dimension $D$, we use the following scaling equation:

$$\eta_{\mathbf{W}}(D) = \eta_{\text{base}} \frac{D_0}{D}. \tag{15}$$

Width-independent parameter groups retain their base learning rate. Initialization follows the corresponding $\mu$P fan-in scaling. For all other layers or components like layer norm or bias, we follow the design recipe in Appendix B.1 in Yang et al. (2022).

Given that $\mu$P has not been demonstrated in JEPA architectures before for hyperparameter transfer across model width to our best knowledge, we don't intend to use $\mu$P to reduce the hyperparameter tuning efforts. Instead, we use $\mu$P here simply to ensure stable feature learning, i.e. maintaining a stable spectral norm of the weight matrices to be around $\|\mathbf{W}\|_* = \Theta(\sqrt{d_{\text{out}}/d_{\text{in}}})$.

In Appendix I, we show the $\mu$P parameterization of learning rate can transfer across width most of the time, although there are occasional exceptions. We intend to investigate this more in future work.

# H Theoretical Results and Proofs

## H.1 Proof of Proposition 3.1 (one-hot linearization)

*Proof.* Since $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$, the open cover of $\mathcal{X}$

$$\mathcal{X} \subseteq \bigcup_{\mathbf{c} \in \mathcal{X}} B_\varepsilon(\mathbf{c}) \tag{16}$$

contains a finite subcover. In other words, for every $\varepsilon > 0$, there exists $C_\varepsilon = \{\mathbf{c}_1, \ldots, \mathbf{c}_N\} \subset \mathcal{X}$ such that

$$\mathcal{X} \subseteq \bigcup_{\mathbf{c} \in C_\varepsilon} B_\varepsilon(\mathbf{c}) \tag{17}$$

where $B_\varepsilon(\mathbf{c}) := \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \mathbf{c}\| < \varepsilon\}$. Now consider a deterministic quantization map $Q_\varepsilon : \mathcal{X} \to \{1, \ldots, N\}$ such that $\left\|\mathbf{x} - \mathbf{c}_{Q_\varepsilon(\mathbf{x})}\right\| \leq \varepsilon$ for all $\mathbf{x} \in \mathcal{X}$.

Let's define the encoder as $E_\varepsilon(\mathbf{x}) = \mathbf{e}_{Q_\varepsilon(\mathbf{x})}$, where $\mathbf{e}_i$ is the $i$-th standard basis vector of $\mathbb{R}^N$. Hence $\|E_\varepsilon(\mathbf{x})\|_0 = 1$ for every $\mathbf{x} \in \mathcal{X}$, i.e. $E_\varepsilon(\cdot)$ is a one-hot encoder.

Correspondingly, we also define the decoder by $D_\varepsilon(\mathbf{e}_i) = \mathbf{c}_i$. Now fix any action $\mathbf{a} \in \mathcal{A}$. For each representative $\mathbf{c}_i$, define $\tau_{\mathbf{a}}(i) := Q_\varepsilon(f(\mathbf{c}_i, \mathbf{a}))$ as the transition dynamics map. Then the transition matrix $\mathbf{P}_\varepsilon(\mathbf{a})$ can be defined column-wise by $\mathbf{P}_\varepsilon(\mathbf{a})\mathbf{e}_i = \mathbf{e}_{\tau_{\mathbf{a}}(i)}$.

Thus every column of $\mathbf{P}_\varepsilon(\mathbf{a})$ contains exactly one nonzero entry, and

$$\mathbf{z}_{t+1} = \mathbf{P}_\varepsilon(\mathbf{a}_t)\mathbf{z}_t \tag{18}$$

is exactly linear in $\mathbf{z}$. It remains to bound the decoded one-step error. Fix arbitrary $\mathbf{x}_t \in \mathcal{X}$ and $\mathbf{a}_t \in \mathcal{A}$, and let $i := Q_\varepsilon(\mathbf{x}_t)$. Then by the definition of the encoder we have $E_\varepsilon(\mathbf{x}_t) = \mathbf{e}_i$ with $\|\mathbf{x}_t - \mathbf{c}_i\| \leq \varepsilon$. We can roll out the latent dynamics to obtain

$$\mathbf{P}_\varepsilon(\mathbf{a}_t)E_\varepsilon(\mathbf{x}_t) = \mathbf{P}_\varepsilon(\mathbf{a}_t)\mathbf{e}_i = \mathbf{e}_{\tau_{\mathbf{a}_t}(i)} \tag{19}$$

The decoded state is thus

$$\hat{\mathbf{x}}_{t+1} := D_\varepsilon(\mathbf{P}_\varepsilon(\mathbf{a}_t)E_\varepsilon(\mathbf{x}_t)) = \mathbf{c}_{Q_\varepsilon(f(\mathbf{c}_i, \mathbf{a}_t))}. \tag{20}$$

Let the ground-truth next state be

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{a}_t). \tag{21}$$

By the triangle inequality, we have

$$\|\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1}\| = \left\|f(\mathbf{x}_t, \mathbf{a}_t) - \mathbf{c}_{Q_\varepsilon(f(\mathbf{c}_i, \mathbf{a}_t))}\right\| \tag{22}$$

$$\leq \|f(\mathbf{x}_t, \mathbf{a}_t) - f(\mathbf{c}_i, \mathbf{a}_t)\| + \left\|f(\mathbf{c}_i, \mathbf{a}_t) - \mathbf{c}_{Q_\varepsilon(f(\mathbf{c}_i, \mathbf{a}_t))}\right\|. \tag{23}$$

Since $f$ is Lipschitz continuous in state, we know

$$\|f(\mathbf{x}_t, \mathbf{a}_t) - f(\mathbf{c}_i, \mathbf{a}_t)\| \leq L\|\mathbf{x}_t - \mathbf{c}_i\| \leq L\varepsilon. \tag{24}$$

Now because $f(\mathbf{c}_i, \mathbf{a}_t) \in \mathcal{X}$, the compactness property implies that

$$\left\|f(\mathbf{c}_i, \mathbf{a}_t) - \mathbf{c}_{Q_\varepsilon(f(\mathbf{c}_i, \mathbf{a}_t))}\right\| \leq \varepsilon. \tag{25}$$

Therefore the approximation error is upper bounded by

$$\|\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1}\| \leq L\varepsilon + \varepsilon = (L+1)\varepsilon. \tag{26}$$

Since the bound is uniform over $\mathbf{x}_t$ and $\mathbf{a}_t$,

$$\sup_{\mathbf{x}_t \in \mathcal{X},\, \mathbf{a}_t \in \mathcal{A}} \|f(\mathbf{x}_t, \mathbf{a}_t) - D_\varepsilon(\mathbf{P}_\varepsilon(\mathbf{a}_t)E_\varepsilon(\mathbf{x}_t))\| \leq (L+1)\varepsilon. \tag{27}$$

Now since the Lipschitz constant $L$ is fixed, we know that $(L+1)\varepsilon \to 0$ as $\varepsilon \to 0$. Therefore, we arrive at

$$\lim_{\varepsilon \to 0} \sup_{\mathbf{x}_t \in \mathcal{X},\, \mathbf{a}_t \in \mathcal{A}} \|f(\mathbf{x}_t, \mathbf{a}_t) - D_\varepsilon(\mathbf{P}_\varepsilon(\mathbf{a}_t)E_\varepsilon(\mathbf{x}_t))\| = 0. \tag{28}$$

□

**Lemma H.1** (Finite-horizon rollout bound)**.** *Under the assumptions of Proposition 3.1, let* $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{a}_t)$ *for* $t = 0, \ldots, H-1$ *be the ground-truth trajectory under an arbitrary action sequence* $\mathbf{a}_0, \ldots, \mathbf{a}_{H-1} \in \mathcal{A}$*. Initialize* $\mathbf{z}_0 = E_\varepsilon(\mathbf{x}_0)$ *and recursively define the latent transition as* $\mathbf{z}_{t+1} = \mathbf{P}_\varepsilon(\mathbf{a}_t)\mathbf{z}_t$*. Let* $\hat{\mathbf{x}}_t = D_\varepsilon(\mathbf{z}_t)$ *be the decoded state. Then*

$$\|\mathbf{x}_H - \hat{\mathbf{x}}_H\| \leq \varepsilon \sum_{k=0}^{H} L^k. \tag{29}$$

*Proof.* Define the decoded quantization map as $q_\varepsilon(\mathbf{x}) := D_\varepsilon(E_\varepsilon(\mathbf{x})) = \mathbf{c}_{Q_\varepsilon(\mathbf{x})}$. By construction, $\|\mathbf{x} - q_\varepsilon(\mathbf{x})\| \leq \varepsilon$ for all $\mathbf{x} \in \mathcal{X}$. Since every latent state $\mathbf{z}_t$ is one-hot, we know that if $\mathbf{z}_t = \mathbf{e}_i$, then $\hat{\mathbf{x}}_t = D_\varepsilon(\mathbf{z}_t) = \mathbf{c}_i$. Since the latent dynamics give

$$\mathbf{z}_{t+1} = \mathbf{P}_\varepsilon(\mathbf{a}_t)\mathbf{e}_i = \mathbf{e}_{Q_\varepsilon(f(\mathbf{c}_i, \mathbf{a}_t))}. \tag{30}$$

We know that

$$\hat{\mathbf{x}}_{t+1} = D_\varepsilon(\mathbf{z}_{t+1}) = q_\varepsilon(f(\hat{\mathbf{x}}_t, \mathbf{a}_t)). \tag{31}$$

Define the scalar rollout error as $e_t := \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|$. Then

$$e_{t+1} = \|f(\mathbf{x}_t, \mathbf{a}_t) - q_\varepsilon(f(\hat{\mathbf{x}}_t, \mathbf{a}_t))\| \tag{32}$$

$$\leq \|f(\mathbf{x}_t, \mathbf{a}_t) - f(\hat{\mathbf{x}}_t, \mathbf{a}_t)\| + \|f(\hat{\mathbf{x}}_t, \mathbf{a}_t) - q_\varepsilon(f(\hat{\mathbf{x}}_t, \mathbf{a}_t))\|$$

$$\leq Le_t + \varepsilon. \tag{33}$$

At $t = 0$, $e_0 = \|\mathbf{x}_0 - \hat{\mathbf{x}}_0\| \leq \varepsilon$. Rolling out $H$ steps gives us

$$e_H \leq L^H e_0 + \varepsilon \sum_{k=0}^{H-1} L^k \leq L^H \varepsilon + \varepsilon \sum_{k=0}^{H-1} L^k = \varepsilon \sum_{k=0}^{H} L^k. \tag{34}$$

Hence we arrive at our conclusion

$$\|\mathbf{x}_H - \hat{\mathbf{x}}_H\| \leq \varepsilon \sum_{k=0}^{H} L^k. \tag{35}$$

□

### H.2 Proof of Corollary 3.2 (finite-horizon one-hot bound)

*Proof.* Partition each coordinate interval $[0,1]$ into $n$ subintervals of equal length $h = \frac{1}{n}$. The resulting partition of $[0,1]^d$ consists of $n^d = N$ axis-aligned hypercubes, each with side length $h$.

Let $\mathbf{c}_i$ denote the center of one such hypercube. For any point $\mathbf{x}$ in that cell, each coordinate differs from the corresponding coordinate of $\mathbf{c}_i$ by at most $h/2$. Hence

$$|\mathbf{x}_j - \mathbf{c}_{i,j}| \le \frac{h}{2}, \qquad j = 1, \dots, d. \tag{36}$$

Therefore, using the Euclidean norm,

$$\|\mathbf{x} - \mathbf{c}_i\|^2 = \sum_{j=1}^{d} |\mathbf{x}_j - \mathbf{c}_{i,j}|^2 \le \sum_{j=1}^{d} \left(\frac{h}{2}\right)^2 = d\frac{h^2}{4}. \tag{37}$$

Taking square roots gives $\|\mathbf{x} - \mathbf{c}_i\| \le \frac{\sqrt{d}}{2} h$. Since $h = 1/n$, we arrive at

$$\varepsilon_N = \frac{\sqrt{d}}{2n} = \frac{\sqrt{d}}{2} N^{-1/d}. \tag{38}$$

Based on Proposition 3.1, the one-step decoded prediction error is bounded by

$$\sup_{\mathbf{x},\mathbf{a}} \|f(\mathbf{x},\mathbf{a}) - D_{\varepsilon_N}(\mathbf{P}_{\varepsilon_N}(\mathbf{a}) E_{\varepsilon_N}(\mathbf{x}))\| \le (L+1)\varepsilon_N. \tag{39}$$

Substituting the expression for $\varepsilon_N$ gives

$$\sup_{\mathbf{x},\mathbf{a}} \|f(\mathbf{x},\mathbf{a}) - D_{\varepsilon_N}(\mathbf{P}_{\varepsilon_N}(\mathbf{a}) E_{\varepsilon_N}(\mathbf{x}))\| \le (L+1)\frac{\sqrt{d}}{2} N^{-1/d} = \frac{\sqrt{d}}{2}(L+1) N^{-1/d}. \tag{40}$$

Since $d$ and $L$ are fixed constants,

$$\sup_{\mathbf{x},\mathbf{a}} \|f(\mathbf{x},\mathbf{a}) - D_{\varepsilon_N}(\mathbf{P}_{\varepsilon_N}(\mathbf{a}) E_{\varepsilon_N}(\mathbf{x}))\| = O(N^{-1/d}). \tag{41}$$

Similarly, the finite-horizon rollout bound from Lemma H.1 gives

$$\|\mathbf{x}_H - \hat{\mathbf{x}}_H\| \le \varepsilon_N \sum_{k=0}^{H} L^k. \tag{42}$$

Substituting the covering radius, we have

$$\|\mathbf{x}_H - \hat{\mathbf{x}}_H\| \le \frac{\sqrt{d}}{2} N^{-1/d} \sum_{k=0}^{H} L^k. \tag{43}$$

Since $H$, $d$, and $L$ are fixed, it follows that

$$\lim_{N\to\infty} \|\mathbf{x}_H - \hat{\mathbf{x}}_H\| = 0. \tag{44}$$

□

### H.3 Proof of Lemma 4.1, model-factored sparse representation

*Proof.* For $\mathbf{x} \in X_q \subseteq S_q$, its barycentric coordinates satisfy $\lambda_q(\mathbf{x}) \in \Delta^r$ and $V_q \lambda_q(\mathbf{x}) = \mathbf{x}$. Since $E(\mathbf{x}) = J_q \lambda_q(\mathbf{x})$, the latent vector is nonnegative, has total mass one, and has at most $r+1$ nonzero coordinates.

Since the coordinate blocks are disjoint, $J_j^\top J_q = \delta_{jq} I_{r+1}$, which gives the stated block identity. The reconstruction formula follows from $V_q J_q^\top E(\mathbf{x}) = V_q \lambda_q(\mathbf{x}) = \mathbf{x}$. Therefore equality of two latent vectors implies equality of both their active modes and their corresponding continuous states, so $E$ is injective. □

### H.4 Mode-Factored Dynamics and Planning

Assume that the controlled dynamics $f : X \times \mathcal{A} \to X$ are affine within each mode:

$$f(\mathbf{x}, \mathbf{a}) = A_q \mathbf{x} + B_q \mathbf{a} + \mathbf{c}_q, \qquad \mathbf{x} \in X_q, \tag{45}$$

where $\mathbf{a} \in \mathcal{A} \subseteq \mathbb{R}^p$.

By Lemma 4.1, $E : X \to \mathcal{Z}$ is injective. Let $D := E^{-1} : \mathcal{Z} \to X$ denote the latent decoder, and define the induced latent predictor by

$$P(\mathbf{z}, \mathbf{a}) := E\big(f(D(\mathbf{z}), \mathbf{a})\big), \qquad (\mathbf{z}, \mathbf{a}) \in \mathcal{Z} \times \mathcal{A}. \tag{46}$$

Since $f(X, \mathcal{A}) \subseteq X$, $P$ maps $\mathcal{Z} \times \mathcal{A}$ into $\mathcal{Z}$, and

$$P(E(\mathbf{x}), \mathbf{a}) = E(f(\mathbf{x}, \mathbf{a})). \tag{47}$$

Fix a planning horizon $H \geq 1$ and a mode sequence $\sigma = (q_0, \ldots, q_H)$. For $0 \leq s \leq t \leq H$, define the state-transition product

$$\Phi_\sigma(t, s) := \begin{cases} I_r, & t = s, \\ A_{q_{t-1}} \cdots A_{q_s}, & t > s. \end{cases} \tag{48}$$

For $1 \leq t \leq H$, define the accumulated affine drift along $\sigma$ by

$$\mathbf{b}_{\sigma,t} := \sum_{j=0}^{t-1} \Phi_\sigma(t, j+1) \mathbf{c}_{q_j}. \tag{49}$$

Given the initial active coordinates $\boldsymbol{\lambda}_0 \in \Lambda_{q_0}$ and an action sequence $\mathbf{a} = (\mathbf{a}_0, \ldots, \mathbf{a}_{H-1}) \in \mathcal{A}^H$, define the candidate physical trajectory associated with $\sigma$ by $\mathbf{x}_0^\sigma(\boldsymbol{\lambda}_0, \mathbf{a}) := V_{q_0} \boldsymbol{\lambda}_0$ and, for $1 \leq t \leq H$,

$$\mathbf{x}_t^\sigma(\boldsymbol{\lambda}_0, \mathbf{a}) := \Phi_\sigma(t, 0) V_{q_0} \boldsymbol{\lambda}_0 + \sum_{j=0}^{t-1} \Phi_\sigma(t, j+1) B_{q_j} \mathbf{a}_j + \mathbf{b}_{\sigma,t}. \tag{50}$$

The mode-sequence region associated with $\sigma$ is

$$\Gamma_\sigma := \left\{ (\boldsymbol{\lambda}_0, \mathbf{a}) \in \Lambda_{q_0} \times \mathcal{A}^H : \mathbf{x}_t^\sigma(\boldsymbol{\lambda}_0, \mathbf{a}) \in X_{q_t} \text{ for } 1 \leq t \leq H \right\}. \tag{51}$$

Thus, $\Gamma_\sigma$ consists of the initial active coordinates and action sequences whose trajectory follows the prescribed mode sequence. For $1 \leq t \leq H$, define the initial-state coefficient

$$M_{\sigma,t} := R_{q_t} \Phi_\sigma(t, 0) V_{q_0} \in \mathbb{R}^{(r+1) \times (r+1)}, \tag{52}$$

the response coefficient of action $\mathbf{a}_j$ at time $t$,

$$N_{\sigma,t,j} := R_{q_t} \Phi_\sigma(t, j+1) B_{q_j} \in \mathbb{R}^{(r+1) \times p}, \qquad 0 \leq j < t, \tag{53}$$

and the affine offset in the target barycentric coordinates,

$$\mathbf{d}_{\sigma,t} := R_{q_t} \mathbf{b}_{\sigma,t} + \boldsymbol{\rho}_{q_t} \in \mathbb{R}^{r+1}. \tag{54}$$

The theorem below shows that any finite-mode piecewise-affine controlled dynamics admit an exact mode-factored, nonnegative, $(r+1)$-sparse latent representation. In this representation, for any fixed planning horizon, the latent trajectory map is affine on each region associated with a fixed mode sequence.

**Theorem H.2** (Finite-Horizon Normal Form for Controlled Mode-Wise Affine Dynamics)**.** *Fix $H \geq 1$ and a mode sequence $\sigma = (q_0, \ldots, q_H)$. For any $(\boldsymbol{\lambda}_0, \mathbf{a}) \in \Gamma_\sigma$, let*

$$\mathbf{x}_0 := V_{q_0} \boldsymbol{\lambda}_0, \qquad \mathbf{x}_{t+1} := f(\mathbf{x}_t, \mathbf{a}_t), \tag{55}$$

*and initialize the latent rollout by*

$$\mathbf{z}_0 := J_{q_0} \boldsymbol{\lambda}_0, \qquad \mathbf{z}_{t+1} := P(\mathbf{z}_t, \mathbf{a}_t). \tag{56}$$

*Then, for every* $1 \le t \le H$,

$$\mathbf{x}_t = \mathbf{x}_t^\sigma(\boldsymbol{\lambda}_0, \mathbf{a}) \in X_{q_t}, \qquad \mathbf{z}_t = E(\mathbf{x}_t), \tag{57}$$

*and*

$$\mathbf{z}_t = J_{q_t}\left(M_{\sigma,t}\boldsymbol{\lambda}_0 + \sum_{j=0}^{t-1} N_{\sigma,t,j}\mathbf{a}_j + \mathbf{d}_{\sigma,t}\right). \tag{58}$$

*Consequently,* $\mathbf{z}_t \in \mathcal{Z}$ *and* $\|\mathbf{z}_t\|_0 \le r+1$ *for every* $0 \le t \le H$. *Thus, on each mode-sequence region* $\Gamma_\sigma$, *the entire finite-horizon latent trajectory is affine in the initial active coordinates and the action sequence, while remaining in the set of valid mode-factored sparse codes.*

*If, in addition, each mode cell* $X_q$ *is convex and describable by finitely many weak or strict affine inequalities, with the strict inequalities determined by the fixed boundary convention, and* $\mathcal{A}$ *is a polytope, then* $\Gamma_\sigma$ *is a convex region describable by finitely many weak or strict affine inequalities in* $(\boldsymbol{\lambda}_0, \mathbf{a})$.

*Proof.* We first verify that the candidate trajectory $\mathbf{x}_t^\sigma$ is the actual physical trajectory. Since $\boldsymbol{\lambda}_0 \in \Lambda_{q_0}$, we have $\mathbf{x}_0 = V_{q_0}\boldsymbol{\lambda}_0 \in X_{q_0}$, and $\mathbf{x}_0 = \mathbf{x}_0^\sigma(\boldsymbol{\lambda}_0, \mathbf{a})$.

Suppose for some $0 \le t < H$ that $\mathbf{x}_t = \mathbf{x}_t^\sigma(\boldsymbol{\lambda}_0, \mathbf{a}) \in X_{q_t}$. Since the dynamics on $X_{q_t}$ are

$$f(\mathbf{x}, \mathbf{a}) = A_{q_t}\mathbf{x} + B_{q_t}\mathbf{a} + \mathbf{c}_{q_t}, \tag{59}$$

we obtain

$$\begin{aligned} \mathbf{x}_{t+1} &= A_{q_t}\mathbf{x}_t + B_{q_t}\mathbf{a}_t + \mathbf{c}_{q_t} && (60)\\ &= A_{q_t}\mathbf{x}_t^\sigma + B_{q_t}\mathbf{a}_t + \mathbf{c}_{q_t} && (61)\\ &= \mathbf{x}_{t+1}^\sigma, && (62) \end{aligned}$$

where the last equality follows from the definitions of $\Phi_\sigma$ and $\mathbf{b}_{\sigma,t}$. Moreover, because $(\boldsymbol{\lambda}_0, \mathbf{a}) \in \Gamma_\sigma$,

$$\mathbf{x}_{t+1}^\sigma(\boldsymbol{\lambda}_0, \mathbf{a}) \in X_{q_{t+1}}. \tag{63}$$

Induction therefore gives

$$\mathbf{x}_t = \mathbf{x}_t^\sigma(\boldsymbol{\lambda}_0, \mathbf{a}) \in X_{q_t}, \qquad 0 \le t \le H. \tag{64}$$

We next show that the latent rollout remains exactly intertwined with the physical trajectory. Since $\mathbf{x}_0 \in X_{q_0}$ and $\lambda_{q_0}(\mathbf{x}_0) = \boldsymbol{\lambda}_0$,

$$\mathbf{z}_0 = J_{q_0}\boldsymbol{\lambda}_0 = E(\mathbf{x}_0). \tag{65}$$

If $\mathbf{z}_t = E(\mathbf{x}_t)$, then

$$\begin{aligned} \mathbf{z}_{t+1} &= P(E(\mathbf{x}_t), \mathbf{a}_t) && (66)\\ &= E(f(\mathbf{x}_t, \mathbf{a}_t)) && (67)\\ &= E(\mathbf{x}_{t+1}). && (68) \end{aligned}$$

Thus $\mathbf{z}_t = E(\mathbf{x}_t)$ for every $0 \le t \le H$. For $1 \le t \le H$, the physical-state formula gives

$$\mathbf{x}_t = \Phi_\sigma(t, 0)V_{q_0}\boldsymbol{\lambda}_0 + \sum_{j=0}^{t-1} \Phi_\sigma(t, j+1)B_{q_j}\mathbf{a}_j + \mathbf{b}_{\sigma,t}. \tag{69}$$

Since $\mathbf{x}_t \in X_{q_t} \subseteq S_{q_t}$, its barycentric coordinates with respect to $S_{q_t}$ satisfy

$$\begin{aligned} \lambda_{q_t}(\mathbf{x}_t) &= R_{q_t}\mathbf{x}_t + \boldsymbol{\rho}_{q_t} && (70)\\ &= M_{\sigma,t}\boldsymbol{\lambda}_0 + \sum_{j=0}^{t-1} N_{\sigma,t,j}\mathbf{a}_j + \mathbf{d}_{\sigma,t}. && (71) \end{aligned}$$

Because $\mathbf{z}_t = E(\mathbf{x}_t) = J_{q_t}\lambda_{q_t}(\mathbf{x}_t)$, it follows that

$$\mathbf{z}_t = J_{q_t}\left(M_{\sigma,t}\boldsymbol{\lambda}_0 + \sum_{j=0}^{t-1} N_{\sigma,t,j}\mathbf{a}_j + \mathbf{d}_{\sigma,t}\right). \tag{72}$$

Every $\mathbf{z}_t$ equals $E(\mathbf{x}_t)$, so Lemma 4.1 gives $\mathbf{z}_t \in \mathcal{Z}$ and $\|\mathbf{z}_t\|_0 \le r+1$. In particular, the rollout never spreads across multiple mode blocks.

For the final statement, observe that

$$\Lambda_{q_0} = \{\boldsymbol{\lambda} \in \Delta^r : V_{q_0}\boldsymbol{\lambda} \in X_{q_0}\}. \tag{73}$$

Hence $\Lambda_{q_0}$ is described by finitely many weak or strict affine inequalities whenever $X_{q_0}$ has that property. For fixed $\sigma$, every map $(\boldsymbol{\lambda}_0, \mathbf{a}) \mapsto \mathbf{x}_t^\sigma(\boldsymbol{\lambda}_0, \mathbf{a})$ is affine. Consequently, each condition $\mathbf{x}_t^\sigma(\boldsymbol{\lambda}_0, \mathbf{a}) \in X_{q_t}$ is also described by finitely many weak or strict affine inequalities in $(\boldsymbol{\lambda}_0, \mathbf{a})$. Intersecting these conditions with $\boldsymbol{\lambda}_0 \in \Lambda_{q_0}$ and $\mathbf{a} \in \mathcal{A}^H$ proves the claim. □

**Planning Implications of Mode-Factored Dynamics** Define the fixed-mode-sequence feasible action set by

$$\mathcal{U}_\sigma(\boldsymbol{\lambda}_0) := \left\{\mathbf{a} \in \mathcal{A}^H : (\boldsymbol{\lambda}_0, \mathbf{a}) \in \Gamma_\sigma\right\}. \tag{74}$$

Also define the stacked action-response matrix

$$\mathsf{N}_{\sigma,H} := [N_{\sigma,H,0} \quad \cdots \quad N_{\sigma,H,H-1}] \in \mathbb{R}^{(r+1)\times Hp}, \tag{75}$$

so that $\sum_{j=0}^{H-1} N_{\sigma,H,j}\mathbf{a}_j = \mathsf{N}_{\sigma,H}\mathbf{a}$.

**Corollary H.3** (Planning Within a Fixed Mode Sequence)**.** *Fix an initial latent state $\mathbf{z}_0 = J_{q_0}\boldsymbol{\lambda}_0$, a goal latent vector $\mathbf{z}_g \in \mathbb{R}^N$, and a mode sequence $\sigma = (q_0, \ldots, q_H)$. Suppose the assumptions of Theorem H.2 hold and that $\mathcal{U}_\sigma(\boldsymbol{\lambda}_0)$ is a nonempty polyhedron. Then the fixed-mode-sequence planning problem*

$$\min_{\mathbf{a}\in\mathcal{U}_\sigma(\boldsymbol{\lambda}_0)} \|\mathbf{z}_H(\mathbf{a}) - \mathbf{z}_g\|_2^2 \tag{76}$$

*is a convex quadratic program.*

*Proof.* By Theorem H.2, for every $\mathbf{a} \in \mathcal{U}_\sigma(\boldsymbol{\lambda}_0)$,

$$\mathbf{z}_H(\mathbf{a}) = J_{q_H}\left(M_{\sigma,H}\boldsymbol{\lambda}_0 + \mathsf{N}_{\sigma,H}\mathbf{a} + \mathbf{d}_{\sigma,H}\right). \tag{77}$$

Define the action-independent terminal offset

$$\mathbf{c}_{\sigma,H} := J_{q_H}\left(M_{\sigma,H}\boldsymbol{\lambda}_0 + \mathbf{d}_{\sigma,H}\right) - \mathbf{z}_g. \tag{78}$$

The terminal cost can then be written as

$$\|\mathbf{z}_H(\mathbf{a}) - \mathbf{z}_g\|_2^2 = \|J_{q_H}\mathsf{N}_{\sigma,H}\mathbf{a} + \mathbf{c}_{\sigma,H}\|_2^2 \tag{79}$$

$$= \mathbf{a}^\top Q_\sigma \mathbf{a} + 2\mathbf{b}_\sigma^\top \mathbf{a} + \gamma_\sigma. \tag{80}$$

where

$$Q_\sigma := \mathsf{N}_{\sigma,H}^\top J_{q_H}^\top J_{q_H}\mathsf{N}_{\sigma,H} \tag{81}$$

$$= \mathsf{N}_{\sigma,H}^\top \mathsf{N}_{\sigma,H} \succeq 0, \tag{82}$$

because $J_{q_H}^\top J_{q_H} = I_{r+1}$, while

$$\mathbf{b}_\sigma := \mathsf{N}_{\sigma,H}^\top J_{q_H}^\top \mathbf{c}_{\sigma,H}, \qquad \gamma_\sigma := \|\mathbf{c}_{\sigma,H}\|_2^2. \tag{83}$$

Hence the terminal cost is a convex quadratic function of the stacked action vector $\mathbf{a}$. By assumption, $\mathcal{U}_\sigma(\boldsymbol{\lambda}_0)$ is a polyhedron. Therefore the objective is convex quadratic and the feasible set is described by affine inequalities, so the fixed-mode-sequence planning problem is a convex quadratic program. □

# I HYPERPARAMETER SWEEPS

## I.1 LPWM AND LEWM SWEEPS IN THE PUSHT ENVIRONMENT

To ensure full reproducibility and transparency, we perform and report hyperparameter sweeps underlying Figure 1b.

We run a grid search over the RDMReg regularization weight $\lambda_{\text{RDMReg}} \in \{0.01, 0.1, 1.0, 10.0\}$ and the muP (or $\mu$P) learning rate $\ell_{\text{muP}} \in \{0.05, 0.005, 0.0005, 0.00005\}$ for each fixed feature dimension $d \in \{384, 768, 1536, 2048, 4096\}$, method (LeWM and LpWM with target distribution $\prod_{i=1}^{d} \text{ReLU}(\text{Laplace}(\mu = 0, \sigma = \sqrt{1/2}))$), and predictor setting in Table 1. To ensure fair comparison and standardized hyperparameter sweeps, we use our sliced Wasserstein distance formulation as the distribution-matching loss for isotropic Gaussian targets instead of the Epps-Pulley loss in SIGReg (Balestriero & LeCun, 2025). They are both in fact instantiations of Cramer-Wold theorem, and only differ in one-sample vs. two-sample style distribution matching (Kuang et al., 2026b).

Due to compute constraints, we fix one seed for training but average over 3 seeds during planning evaluations with CEM (Rubinstein, 1999). In practice, we execute our sweep sequentially. So there are some predictor types where we have done a very thorough grid search. After we have determined that some hyperparameters are always leading to bad performances (e.g. usually learning rate like $0.05$ is too large), we drop them in our subsequent sweeps so the resulting figures for different predictor types usually cover different subset of the full grid.

In Figure 9, we observe that $\ell_{\text{muP}} \in \{0.05, 0.005\}$ almost always leads to $0\%$ success rate. Subsequent hyperparameter sweeps shown in Figure 6, Figure 7, Figure 8, Figure 11, and Figure 10 selectively remove portions of the grid to save compute.

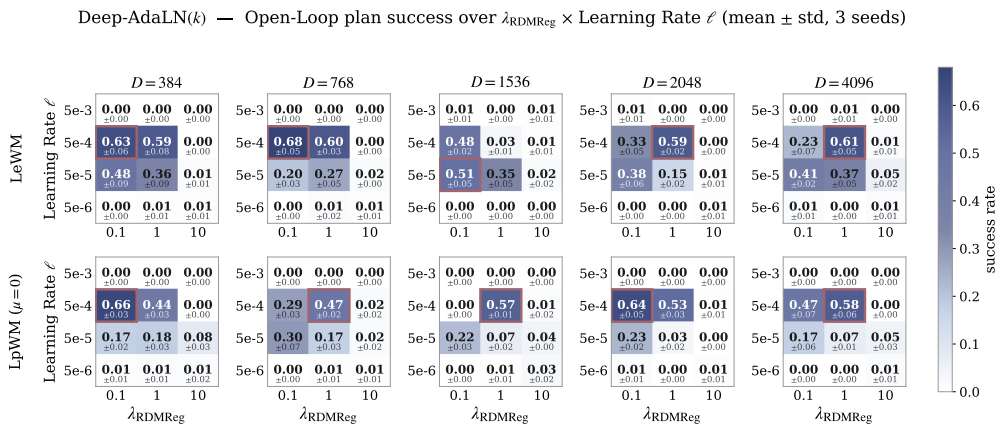
## I.2 VICREG SWEEP IN THE PUSHT ENVIRONMENT

We also perform extensive hyperparameter sweeps for JEPA regularized with VICReg. Basically, we take the LeWM setting and replaced the SIGReg regularizer with VICReg. The resulting model doesn't have a name yet, and we abuse the notation and simply call it VICReg here.

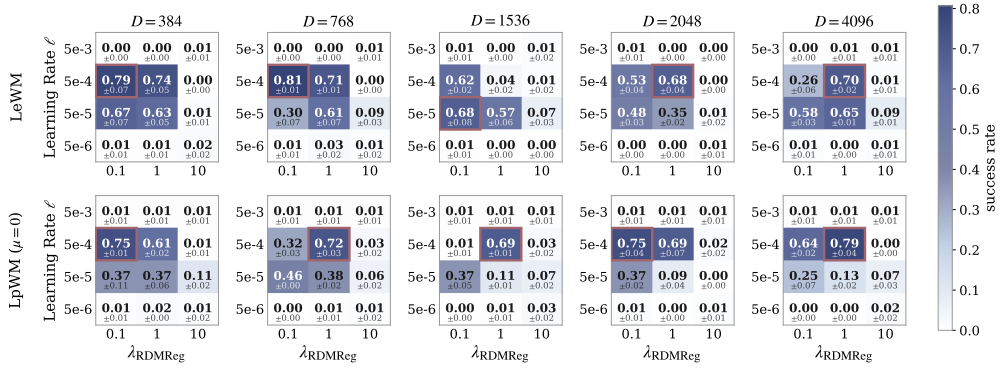
In Figure 12, Figure 13, and Figure 14, we show the sweep results for the predictors Deep-AdaLN($k$), MLP$\circ$LTV($k$), and MLP$\circ$LTI($k$) respectively.

## I.3 POINTMAZE LEARNING-RATE–REGULARIZATION SWEEPS

For both PointMaze environments, we sweep the $\mu$P learning rate over $\{4\times10^{-6}, 2\times10^{-5}, 10^{-4}, 5\times10^{-4}\}$ and RDMReg weight over $\{0.01, 0.1, 1, 10\}$. The PointMaze-Diverse variant additionally includes weight $100$. Medium uses $20$ training epochs and Diverse uses 3, with batch size $128$ and $8192$ projections throughout. The reported Medium configurations are $(\ell, \lambda) = (2 \times 10^{-5}, 0.1)$ for LeWM and $(10^{-4}, 0.1)$ for LpWM. Both Diverse models use $(2 \times 10^{-5}, 10)$. Table 7 reports planning success and sparsity for these configurations.

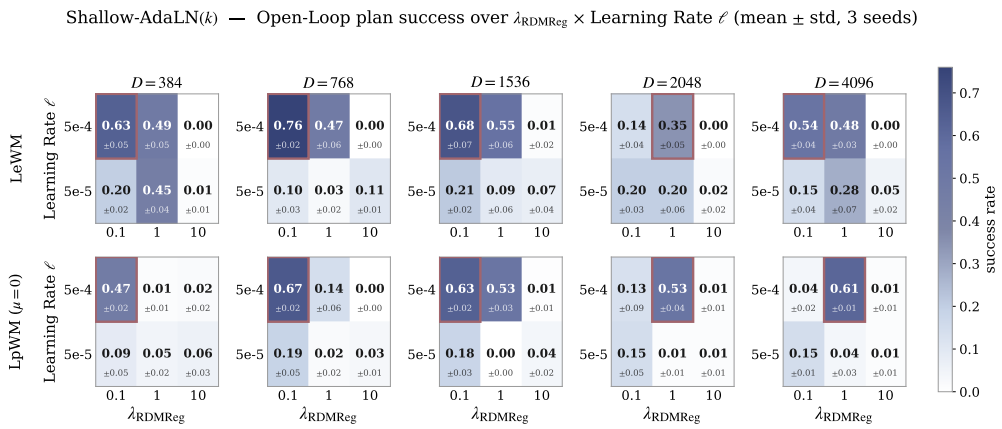



(a) Open-Loop.

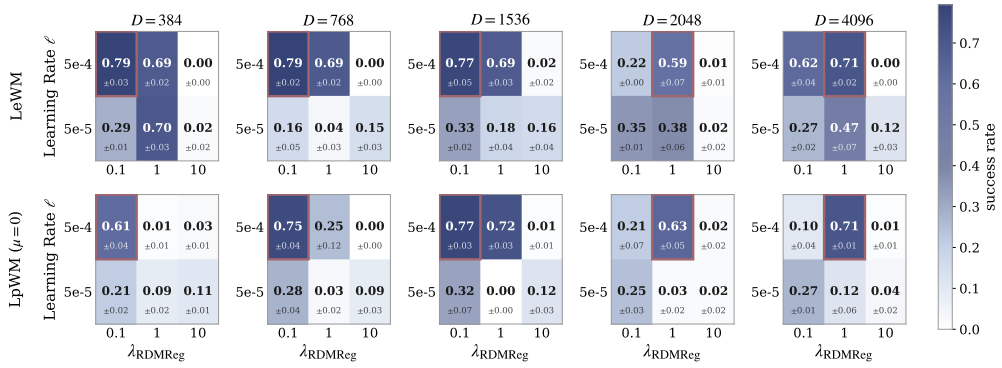



(b) Closed-Loop.

Figure 6: **Deep-AdaLN$(k)$ Predictor.** Hyperparameter Sweeps for the PushT Environment.

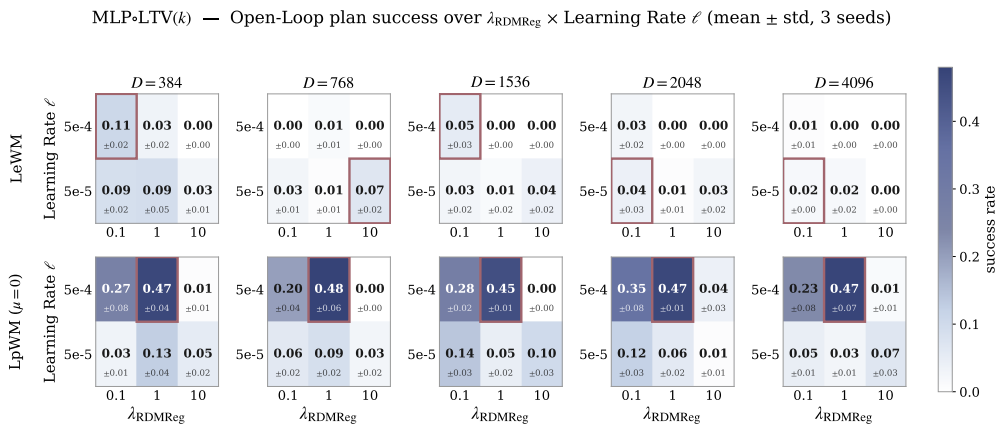



(a) Open-Loop.

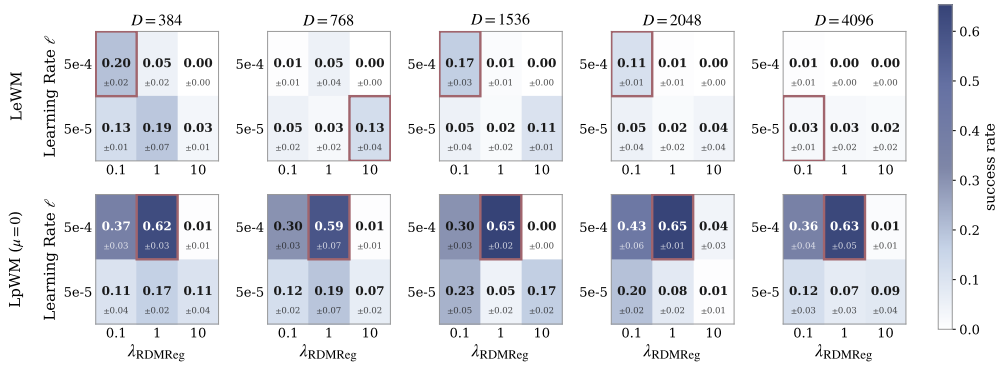



(b) Closed-Loop.

Figure 7: **Shallow-AdaLN**$(k)$ **Predictor.** Hyperparameter Sweeps for the PushT Environment.




(a) Open-Loop.

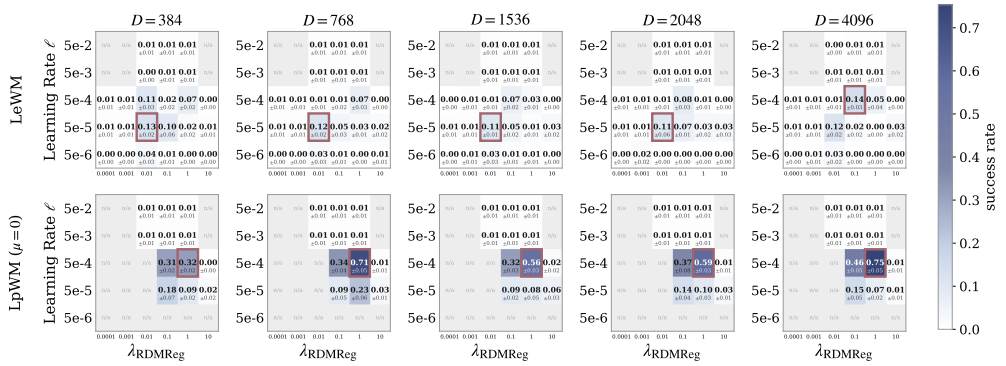



(b) Closed-Loop.

Figure 8: **MLP∘LTV$(k)$ Predictor.** Hyperparameter Sweeps for the PushT Environment.

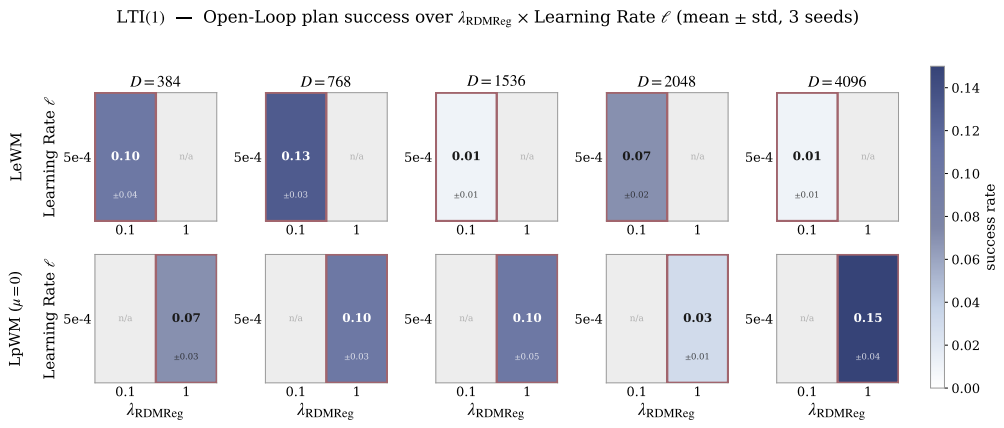



(a) Open-Loop.

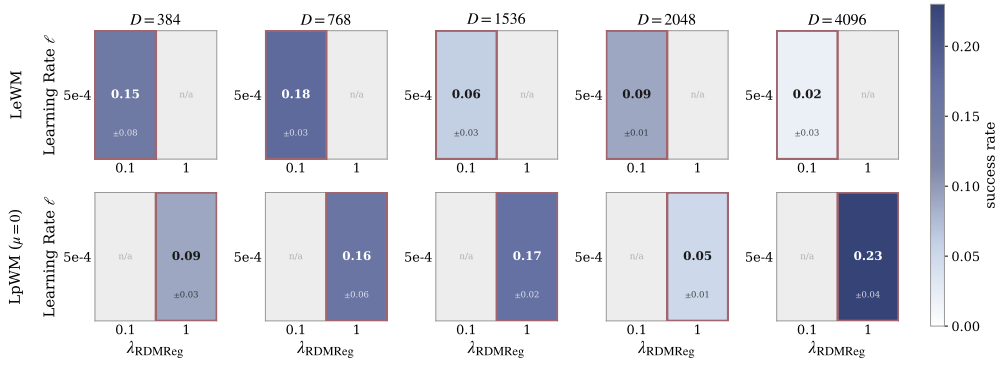



(b) Closed-Loop.

Figure 9: **MLP∘LTI$(k)$ Predictor.** Hyperparameter Sweeps for the PushT Environment.

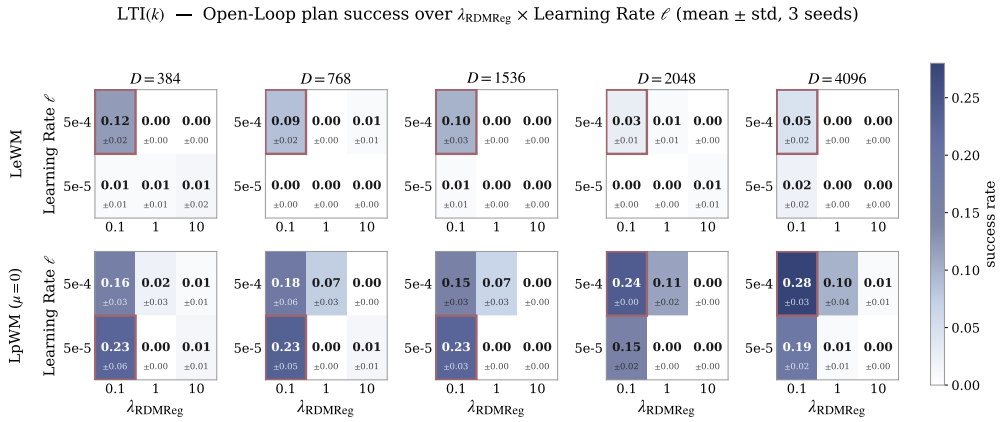



(a) Open-Loop.

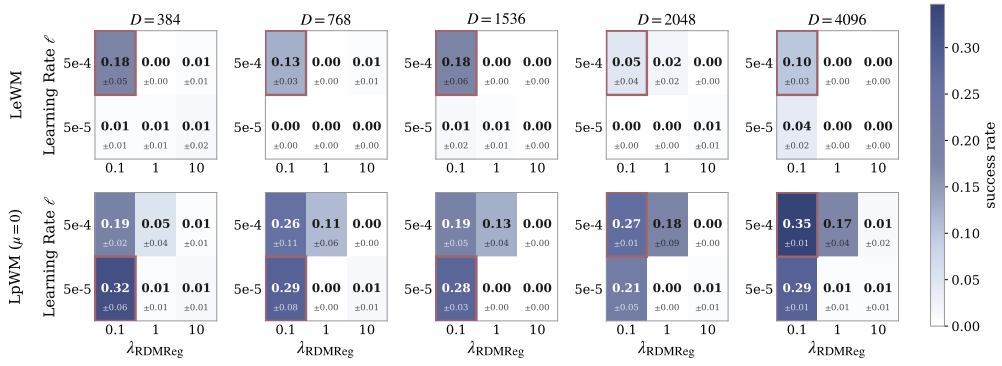



(b) Closed-Loop.

Figure 10: **LTI**(1) **Predictor.** Hyperparameter Sweeps for the PushT Environment.

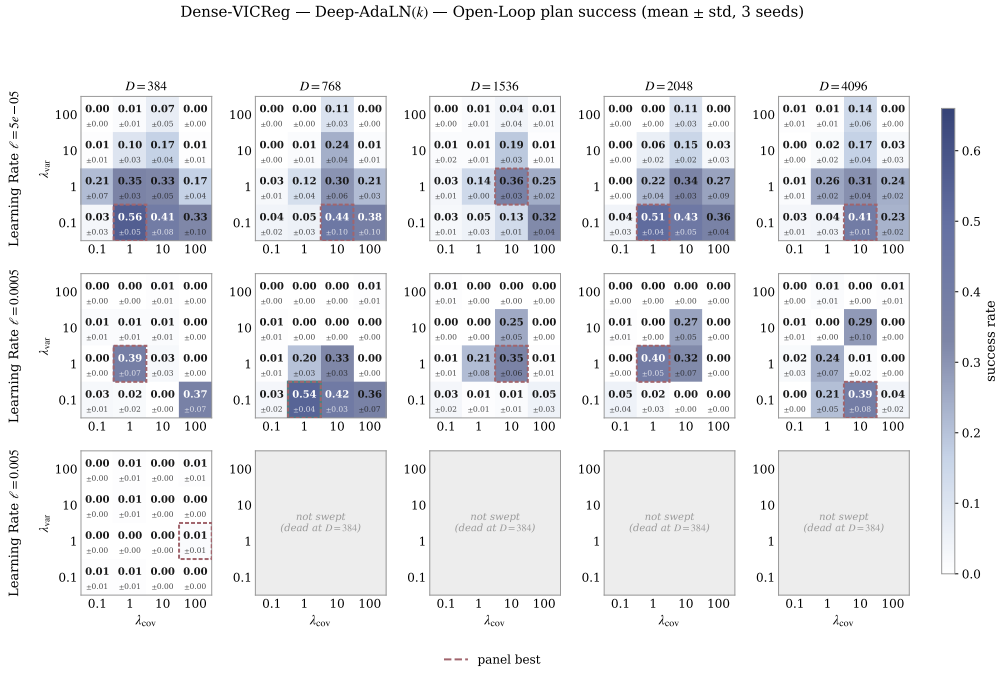



(a) Open-Loop.

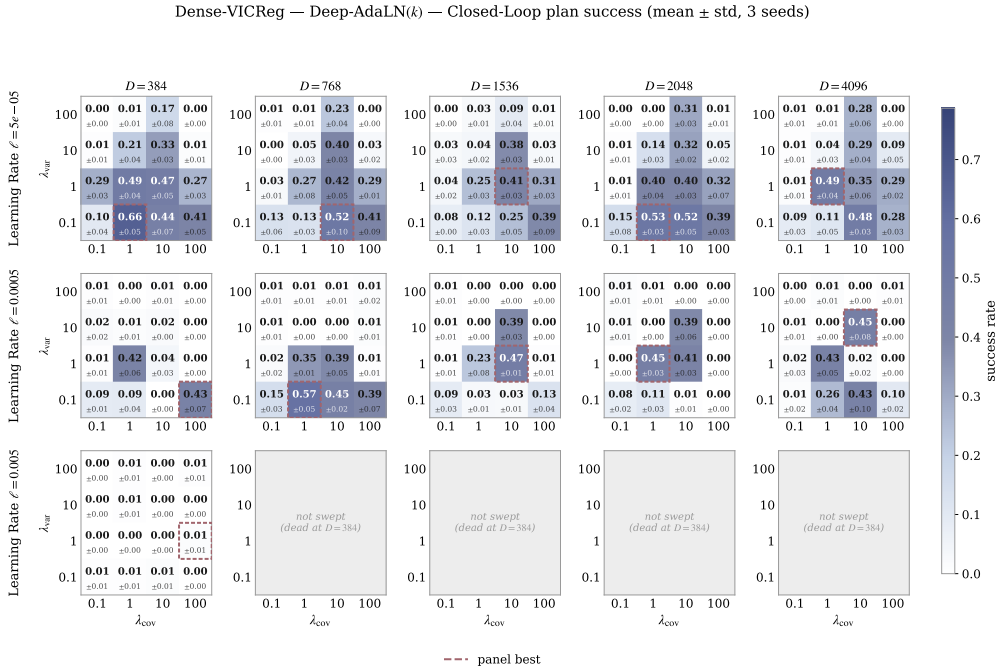



(b) Closed-Loop.

Figure 11: **LTI($k$) Predictor.** Hyperparameter Sweeps for the PushT Environment.

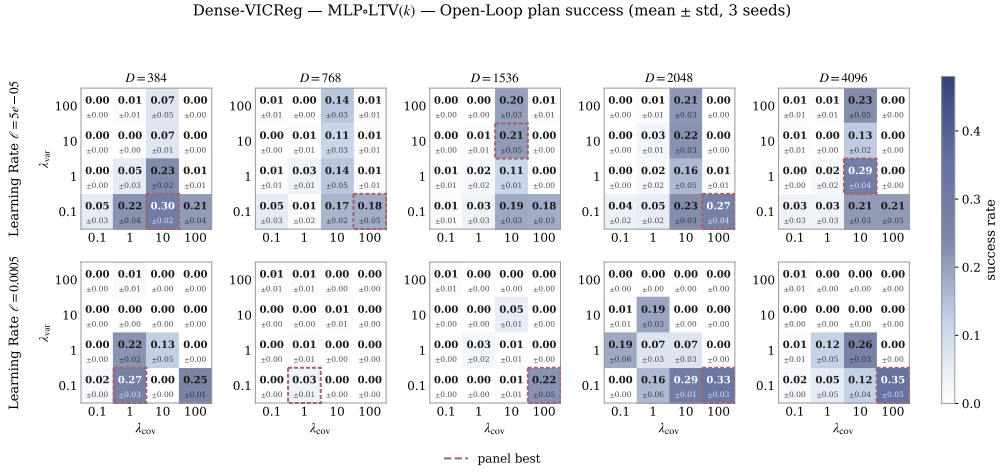



(a) Open-Loop.

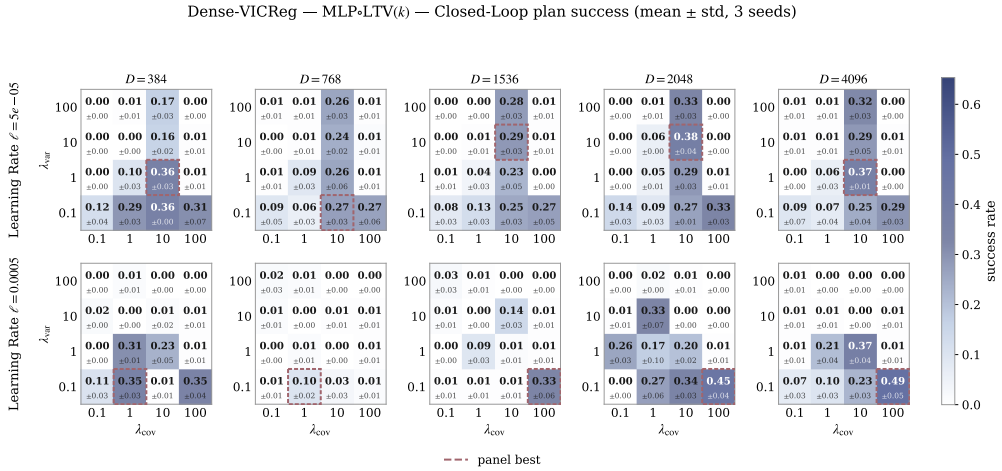



(b) Closed-Loop.

Figure 12: **Deep-AdaLN**$(k)$ **Predictor.** Hyperparameter Sweeps for VICReg-based JEPA models in the PushT Environment.

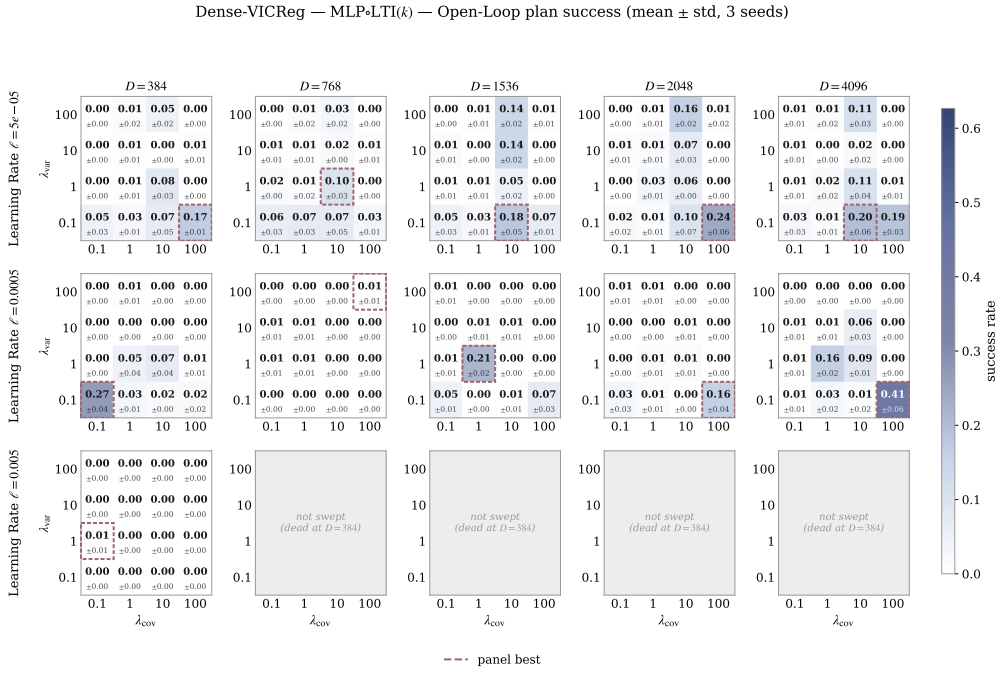



(a) Open-Loop.

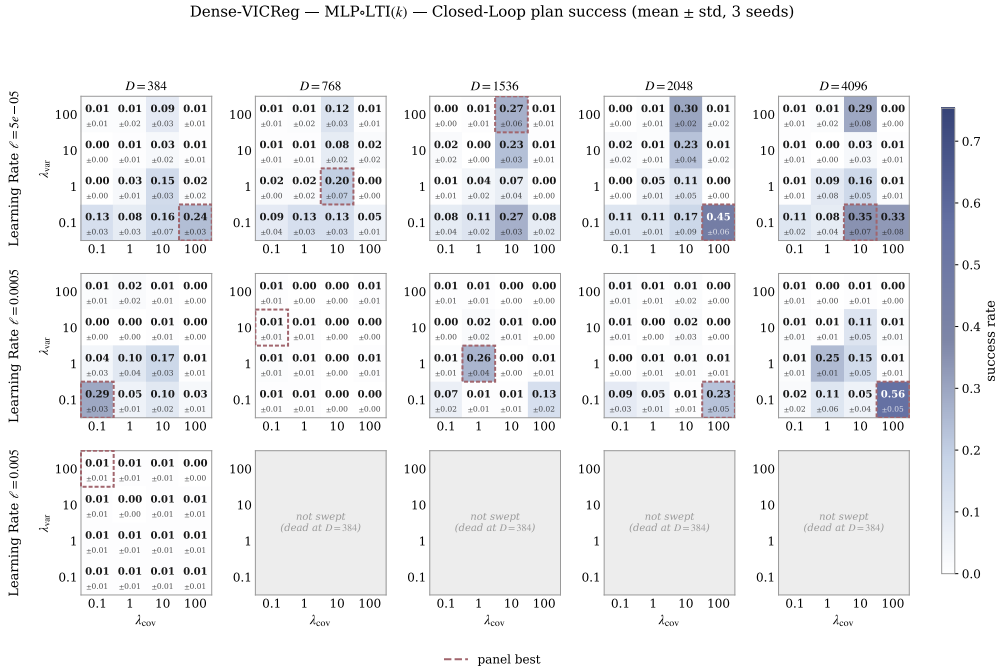



(b) Closed-Loop.

Figure 13: **MLP∘LTV$(k)$ Predictor.** Hyperparameter Sweeps for VICReg-based JEPA models in the PushT Environment.




(a) Open-Loop.




(b) Closed-Loop.

Figure 14: **MLP∘LTI$(k)$ Predictor.** Hyperparameter Sweeps for VICReg-based JEPA models in the PushT Environment.

# J POINTMAZE PLANNING TRAJECTORIES

## J.1 FIXED-LAYOUT NAVIGATION

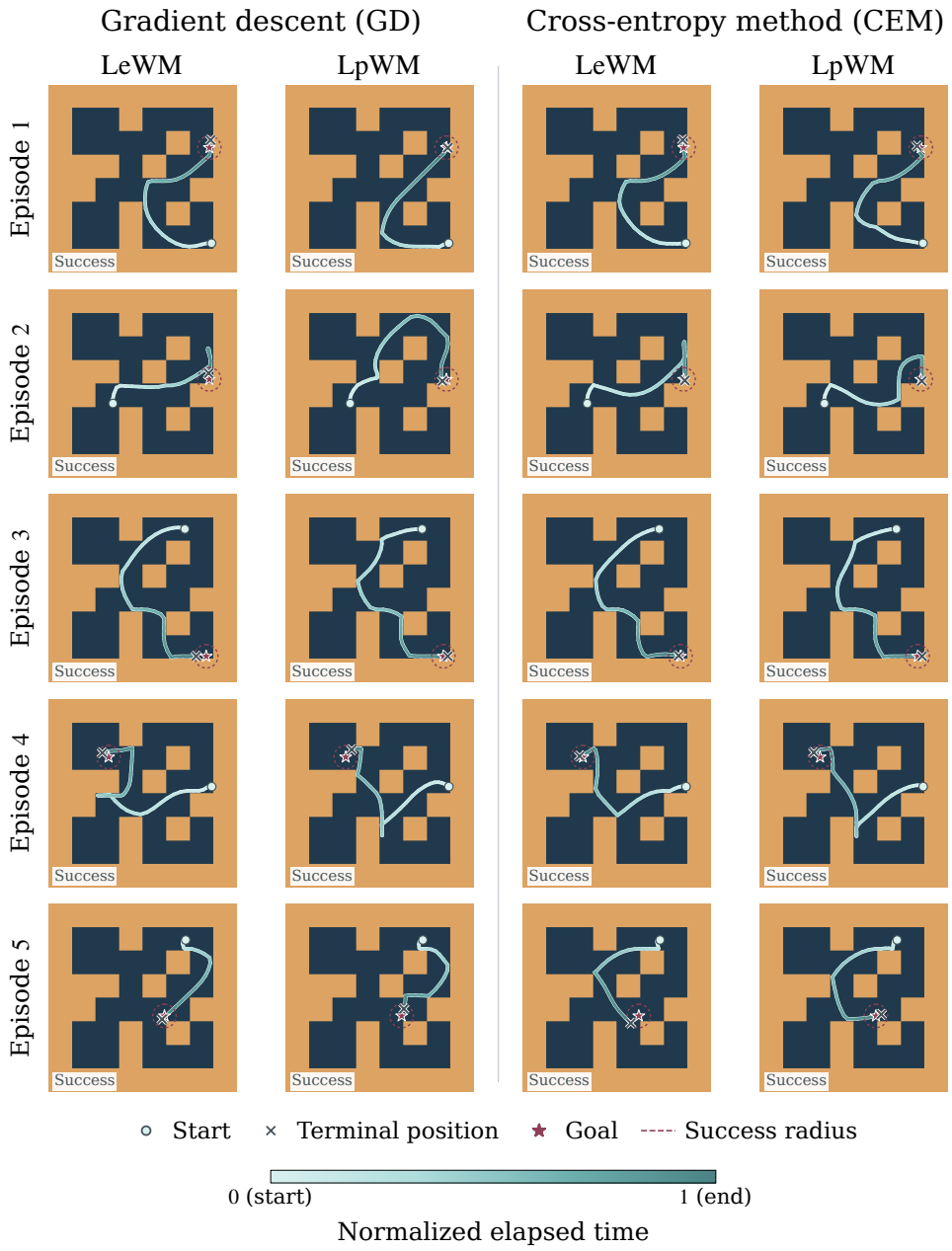



Figure 15: **PointMaze-Medium.** Rows show five matched tasks successful under all four model–planner combination. Shading indicates normalized elapsed time. Circles, crosses, stars, and dashed circles mark starts, terminal positions, goals, and the 0.5-unit success radius. Backgrounds preserve the original maze geometry and colors.

### J.2 NAVIGATION ON UNSEEN LAYOUTS

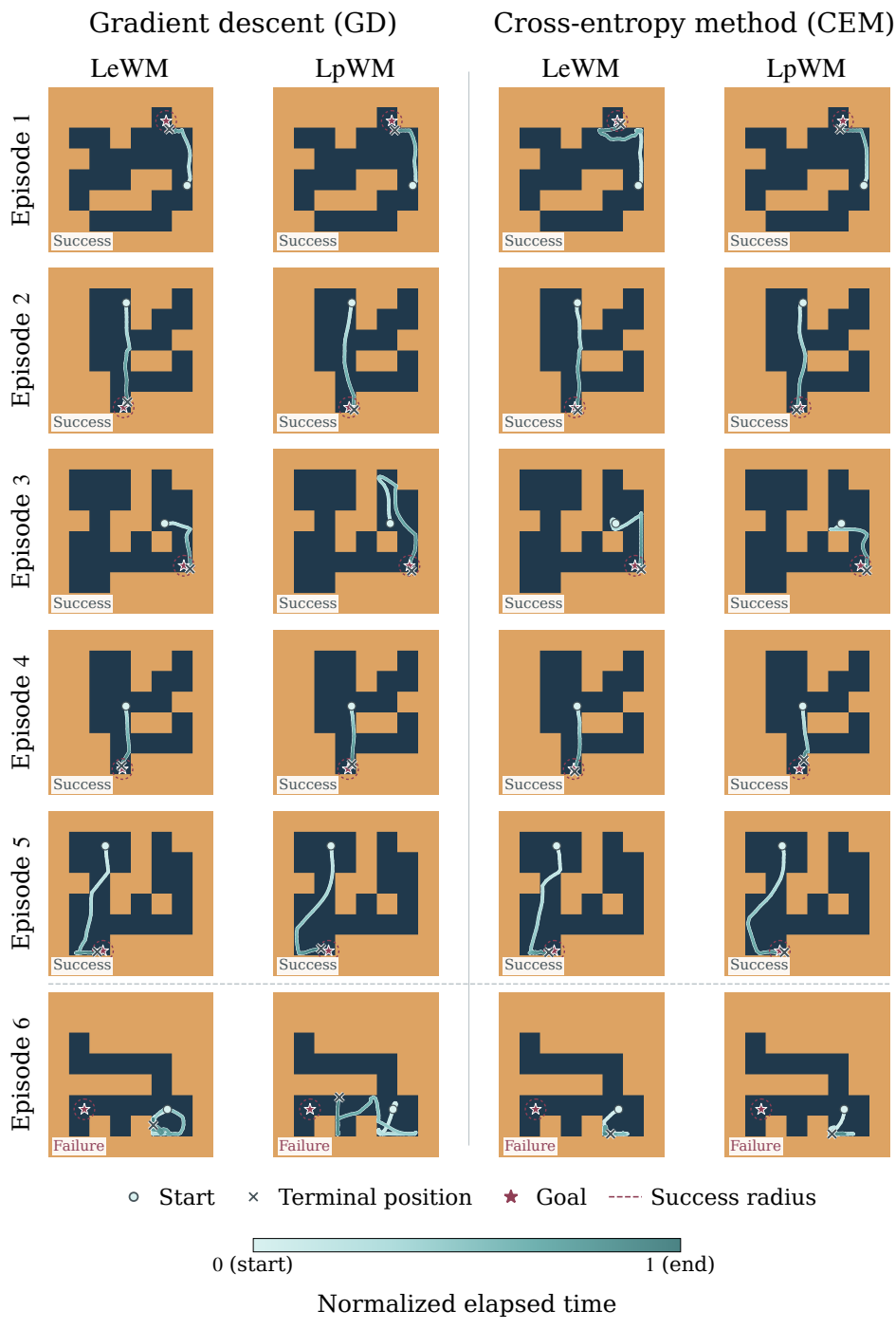



Figure 16: **Diverse PointMaze.** The first five rows show episodes successful under both LeWM and LpWM with both GD and CEM. The last row shows an episode in which all four combinations fail. Paths and symbols follow Figure 15.