WORLD2VEC: STABLE INFERENCE OF MECHANISTIC WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World models learned from raw pixels have become a practical basis for planning and control, and recent joint-embedding predictive architectures can be trained stably end-to-end. Yet these models are typically evaluated in environments whose dynamics are largely driven by the agent's actions, and stable training alone does not ensure that learned latent representations recover the underlying state or dynamics of the environment. Here, we introduce world2vec, a framework for learning mechanistic world models directly from observations. We show that world2vec recovers the ground-truth state under nonlinear observation mixing and learns latent dynamics that faithfully reproduce the true system over long prediction horizons while using one order of magnitude fewer parameters than existing world models. We show state of the art performance in controlling continuous control environments from pixels. world2vec treats the representation objective and dynamics model as hyperparameters, allowing their inductive biases to be automatically matched to the data-generating process. This enables recovery of ground-truth dynamical systems and their governing equations across simple physical environments and a dataset of 71 strange attractors and paves the way towards using world2vec for scientific discovery.

1 INTRODUCTION

Latent world models for control aim to jointly estimate a representation and its dynamics from observations and interventions. Given a data-generating process of the form

$$s_{t+1} = p(s_t, u_t), \quad x_t = g(s_t) \tag{1}$$

a world model should infer a representation of the environment that faithfully models both the states s and their temporal evolution p. Such a world model can support model-predictive control and model-based reinforcement learning. In recent years, this type of world modeling has become feasible and gained theoretical grounding. Early models such as Dreamer (Hafner et al., 2025) use reconstruction-based forecasting. TD-MPC2 (Hansen et al., 2024) performs planning in a learned latent space without reconstructing observations. More recently LeWorldModel (Maes et al., 2026) used a joint-embedding predictive architectures (JEPA) to demonstrate world modeling for control tasks (Balestriero & LeCun, 2025).

However, world modeling extends beyond solving a particular control task. We also seek to ensure that the world model accurately reflects the world in a statistical sense: Does it reproduce the relevant dynamics when the statistical assumptions used to build it are satisfied? This can be measured. An appropriate parameterization can further enable mechanistic interpretability by allowing us to read out the environmental dynamics. This has value both for interpretability and for scientific applications. World models with these properties have recently been characterized as *mechanistic world models* by Posner et al. (2026).

In this paper, we propose *world2vec* to advance the shared goal of building a world model that (1) achieves state-of-the-art performance on control tasks after training on offline data alone, (2) faithfully models the world and supports long-range rollouts and predictions, and (3) is mechanistically interpretable and thus suitable for scientific applications. World2vec is agnostic to the exact training objective, provided it belongs to a class of objectives known to yield identifiable representations under idealized conditions. We consider LeJEPA (Balestriero & LeCun, 2025; Maes et al., 2026)

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

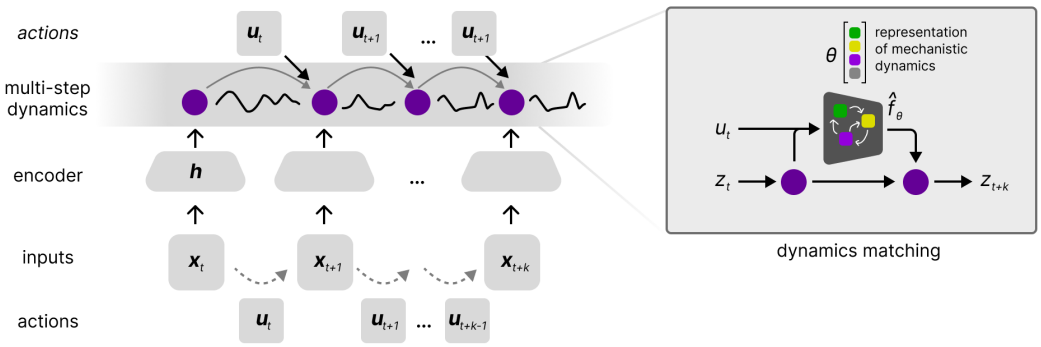

Figure 1: The world2vec framework. world2vec jointly learns representations and dynamics in latent space; the dynamics model resembles an action-conditioned neural ODE and can be parameterized to provide an interpretable vector representation of the dynamics. The objective function can be chosen to incorporate assumptions about the underlying data-generating process.

and dynamics contrastive learning (DCL) (González Laiz et al., 2025) as two possible approaches to achieving this goal. These objectives encode slightly different statistical assumptions using different numerical formulations, and an increasing body of theoretical work addresses the identifiability (Zimmermann et al., 2021; González Laiz et al., 2025; Klindt et al., 2026; Mikulasch & Zenke, 2026) of these objectives.

The key distinction is the principle of *dynamics learning*, which focuses on matching the model's latent dynamics to those of the environment rather than focusing solely on representation learning. This is achieved by predicting encoded future observations from observation histories and controls using a simple ODE solver. The dynamics model family can be parameterized and constrained to incorporate assumptions about the data-generating process and support mechanistic interpretability.

2 Mechanistic world modeling by dynamics matching

Mechanistic world modeling seeks a complete representation of the world state, faithful latent dynamics, and a compact parameterization of the governing equations. Consider a system of the form

$$\dot{\boldsymbol{s}} = \mathbf{f}(\boldsymbol{s}, \boldsymbol{u}), \qquad \boldsymbol{x} = \mathbf{g}(\boldsymbol{s}), \qquad \boldsymbol{z} = \mathbf{h}(\boldsymbol{x}) = \mathbf{r}(\boldsymbol{s}). \tag{2}$$

with state $\boldsymbol{s} \in \mathcal{S}$, control $\boldsymbol{u}$, observation $\boldsymbol{x}$, and learned state $\boldsymbol{z}$. The agent observes $\boldsymbol{x}$ and $\boldsymbol{u}$, but does not use $\boldsymbol{s}$ or $\mathbf{f}$ as training targets. The question is whether the learned vector field $\mathbf{v}$ reproduces the true evolution after the change of coordinates $\mathbf{r} = \mathbf{h} \circ \mathbf{g}$. Assume that $\mathbf{r}$ is differentiable and invertible on the reachable state manifold. With Jacobian $\boldsymbol{J}_{\mathbf{r}}$, the chain rule gives the latent rate and desired matching identity:

$$\dot{\boldsymbol{z}} = \boldsymbol{J}_{\mathbf{r}}(\boldsymbol{s})\mathbf{f}(\boldsymbol{s}, \boldsymbol{u}), \qquad \mathbf{v}(\mathbf{r}(\boldsymbol{s}), \boldsymbol{u}) \stackrel{!}{=} \boldsymbol{J}_{\mathbf{r}}(\boldsymbol{s})\mathbf{f}(\boldsymbol{s}, \boldsymbol{u}). \tag{3}$$

We seek an encoder–dynamics pair $(\mathbf{h}, \mathbf{v})$ whose velocity field generates the encoded trajectories. The idealized prediction loss is

$$\begin{aligned} \mathcal{L}_{\text{rate}} &= \mathbb{E}\Big[\|\boldsymbol{J}_{\mathbf{r}}(\boldsymbol{s})\mathbf{f}(\boldsymbol{s}, \boldsymbol{u}) - \mathbf{v}(\mathbf{r}(\boldsymbol{s}), \boldsymbol{u})\|_2^2\Big] \\ &= \mathbb{E}\Big[\|\dot{\boldsymbol{z}} - \mathbf{v}(\boldsymbol{z}, \boldsymbol{u})\|_2^2\Big]. \end{aligned} \tag{4}$$

The idealized learning problem is $\min_{\mathbf{h},\mathbf{v}} \mathcal{L}_{\text{rate}} + \lambda\mathcal{R}$, where $\mathcal{R}$ regularizes the representation. We hypothesize that modeling state *change* through the velocity field $\mathbf{r}$ in continuous time or an increment in discrete time provides a useful inductive bias for world modeling.

3 World2vec: A framework for mechanistic world models

World2vec combines an observation encoder $\mathbf{h}$ and a latent dynamics model $\hat{\mathbf{f}}$, trained jointly through multi-step prediction. The offline dataset $\mathcal{D}$ contains observations sampled at intervals Δt and

recorded controls. We sample valid rollout starting points $n \sim p_{\mathcal{D}}$, each selecting observations $\boldsymbol{x}_{n-k+1:n+K}$ and controls $\boldsymbol{u}_{n:n+K-1}$. The encoder maps the history $\boldsymbol{X}_n = (\boldsymbol{x}_{n-k+1}, \ldots, \boldsymbol{x}_n)$ to an initial latent state. The dynamics model $\hat{\mathbf{f}}(\boldsymbol{z}, \boldsymbol{u})$ predicts the latent rate of change, which a numerical integration step $\boldsymbol{\Phi}$ uses to advance the state by Δt. Holding each recorded control constant within its sampling interval, we obtain the rollout

$$\hat{\boldsymbol{z}}_{n,0} = \mathbf{h}(\boldsymbol{X}_n), \qquad \hat{\boldsymbol{z}}_{n,j+1} = \boldsymbol{\Phi}(\hat{\boldsymbol{z}}_{n,j}, \boldsymbol{u}_{n+j}), \quad j = 0, \ldots, K-1. \tag{5}$$

Here, $\hat{\boldsymbol{z}}_{n,j}$ is the inferred latent state predicted j steps ahead from starting point n. Our main experiments use explicit Euler integration:

$$\boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{u}) = \boldsymbol{z} + \Delta t\, \hat{\mathbf{f}}(\boldsymbol{z}, \boldsymbol{u}). \tag{6}$$

We apply this procedure K times to produce a rollout in latent state and minimize the objective function

$$\min_{\mathbf{h}, \hat{\mathbf{f}}} \mathbb{E}_{n \sim p_{\mathcal{D}}} \left[\sum_{j=1}^{K} w_j \left\| \mathbf{h}(\boldsymbol{X}_{n+j}) - \hat{\boldsymbol{z}}_{n,j} \right\|_2^2 \right] + \mathcal{R}(\mathbf{h}, \hat{\mathbf{f}}) \tag{7}$$

where w_j are optional weightings for each step and $\mathcal{R}(\mathbf{h}, \hat{\mathbf{f}})$ is a regularizer to prevent representational collapse. In dynamics contrastive learning (DCL), prediction scores based on squared distances are normalized against sampled negative embeddings (González Laiz et al., 2025). In the LeJEPA instantiation, SIGReg regularizes embedding projections toward an isotropic Gaussian (Balestriero & LeCun, 2025). Both objectives discourage representation collapse within the same encoder–dynamics architecture. We refer to the resulting models as world2vec-dcl and world2vec-sigreg.

Identifiability and embedding alignment. We assess empirical identifiability by testing whether independently trained encoders recover equivalent representations up to an affine change of coordinates. For two encoders $\mathbf{h}_1$ and $\mathbf{h}_2$, we fit

$$\mathbf{h}_2(\boldsymbol{X}_n) \approx L\mathbf{h}_1(\boldsymbol{X}_n) + \mathbf{b}. \tag{8}$$

The R^2 of this alignment measures representation consistency across training runs. Within each model, we assess whether the learned dynamics reproduce the evolution of its representation using rollout equivariance,

$$\text{Eq. } R_j^2 = R^2(\{\mathbf{h}(\boldsymbol{X}_{n+j})\}_{n \in \mathcal{V}}, \{\hat{\boldsymbol{z}}_{n,j}\}_{n \in \mathcal{V}}), \tag{9}$$

where $\mathcal{V}$ indexes sampled validation windows and j is the prediction horizon. This score compares predictions and encoded futures from the same encoder–dynamics pair and serves as our primary model-selection criterion.

Dynamics parameterization. An MLP can parameterize $\hat{\mathbf{f}}$, taking the latent state and control as inputs and returning their induced rate of change. Alternatively, a structured dynamics model takes the form

$$\hat{\mathbf{f}}(\boldsymbol{z}, \boldsymbol{u}) = \boldsymbol{A}\boldsymbol{\psi}(\boldsymbol{z}, \boldsymbol{u}), \tag{10}$$

where $\boldsymbol{\psi}$ is a fixed feature library and $\boldsymbol{A}$ contains learned coefficients. Our SINDy instantiation uses polynomial features for $\boldsymbol{\psi}$ and an L_1 coefficient penalty (Brunton et al., 2016; Champion et al., 2019; Gao & Kutz, 2022). The encoder and coefficients are learned jointly through rollout prediction, yielding equations expressed in the learned coordinates.

Goal-conditioned planning. Given a goal history $\boldsymbol{X}^\star$, the encoder provides a target $\boldsymbol{z}^\star = \mathbf{h}(\boldsymbol{X}^\star)$. Starting from the current history $\boldsymbol{X}_n$, we roll out candidate control sequences and minimize

$$\mathrm{J}(\boldsymbol{u}_{n:n+P-1}) = \sum_{j=1}^{P} \alpha_j \|\hat{\boldsymbol{z}}_{n,j} - \boldsymbol{z}^\star\|_2^2, \qquad \alpha_j \geq 0, \quad \sum_{j=1}^{P} \alpha_j = 1, \tag{11}$$

subject to the environment's control bounds. The controller executes a prefix of the selected plan and replans from newly observed data, using neither a learned reward model nor a state decoder.

4 EXPERIMENTS

Environments and dataset collection. We evaluate our method on three MuJoCo-based DeepMind Control Suite (Tassa et al., 2018) environments: Pendulum, Cartpole, and Reacher. Training world2vec requires an offline dataset for each of them. To simulate realistic offline data collection, we avoid any form of expert control policy and drive the data collection with random policies instead. For Reacher, we sample actions uniformly at random. For Pendulum and Cartpole, we use an Ornstein–Uhlenbeck (OU) process to generate mean-reverting but temporally correlated action sequences, clipped to the valid action range. For Cartpole, we additionally push the cart back whenever it approaches the boundary of the rail so that it stays in the visible part of the scene. We record 64×64 RGB frames at 50 Hz for Pendulum and Cartpole and at 10 Hz for Reacher (200k frames for Pendulum and Cartpole, 400k for Reacher) to train all world2vec models.

For comparison with the state of the art, we additionally reproduced the data collection of LeWM (Maes et al., 2026) (dataset D2), with 200k frames per environment, half of them from random-policy episodes and half from a TD-MPC2 expert. We also construct variants of D1 with non-visual observations: following González Laiz et al. (2025), the true state is passed through a randomly initialized, injective mixing MLP of varying depth and projected to 50 dimensions. Crucially, in the variant we report, only the position and angle coordinates are mixed, not the velocities, so the world model has to infer velocities from sequences of observations. Appendix D gives the full collection protocol, including initial-state distributions and dataset sizes.

Model families and training. Encoders take a short sequence of observations as input ($T = 3$ frames in the main experiments; 1, 5, and 9 frames in ablations). The image encoder applies a small vision transformer to each frame (16×16 patches, 2 layers of width 64 with 4 heads, and a keypoint readout) and an MLP head (2 hidden layers of 128 units) that maps the sequence of frame representations to the latent state. On nonlinear-mixing observations, we concatenate the observation sequence and pass it through an MLP encoder (3 hidden layers of 128 units). Dynamics models are MLPs with 3 hidden layers of 128 units and come in two flavors: discrete-time dynamics, $\boldsymbol{z}_{t+1} = \mathbf{v}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, \boldsymbol{u}_t)$, where the MLP directly predicts the next latent state, and continuous-time dynamics, where the MLP parametrizes the vector field and $\boldsymbol{z}_{t+1}$ is obtained by one Euler step (Equation (6)). With a 32-dimensional latent space, our main model has 263k parameters (221k for the encoder, 42k for the dynamics). All models are trained for 20k iterations with Adam (learning rate 10^{-3}, cosine decay). Appendix E describes the architectures in detail.

Model predictive control. Similar to LeWM (Maes et al., 2026), we control the environments with model predictive control (MPC) in latent space, completely reward-free: we only assume knowledge of an observation sequence that shows the target. We encode this sequence to obtain a goal latent that describes which part of the latent space the environment should be driven towards. For Pendulum and Cartpole, this is a rendered sequence of frames that shows the pole or pendulum upright at rest, with the cart centered. For Reacher, the target changes in every episode and the goal is underdefined, since several arm configurations place the fingertip on the target. We therefore compute, per episode, the arm configuration that places the fingertip on the target via inverse kinematics, render it, and use its encoding as the goal; both elbow configurations are planned and the better one is reported. We then alternate between planning and acting: the cross-entropy method (CEM) samples candidate action sequences, the dynamics model predicts where in latent space each of them leads, and the candidates are ranked by their distance to the goal latent (Equation (11)). We execute the first actions of the best sequence, observe the environment, and plan again. Because the continuous-time dynamics can be integrated with a different step size, Reacher models trained at 10 Hz can plan at 50 Hz without retraining. Appendix F lists all planner settings.

Evaluation Metrics All our evaluation metrics are computed and reported on the validation set. Our metrics can be categorized along two main axes: First, whether they are unsupervised and can be computed without access to ground truth state/latent information. We measure quality of representation through consistency, which is the R^2 of an optimal affine alignment across representations derived from different model seeds. We measure quality of the rollouts and hence the dynamics through two equivariance scores called Equivariance R2 and Equivariance Accuracy. We encode a sequence of observations and predict their future latents $\hat{\boldsymbol{z}}_{t+k}$. We either compare these future latents to the representation using an R2, or perform a kNN lookup to identify the embedding in latent

space that is closest to $\hat{\boldsymbol{z}}_{t+k}$ over a set of 512 negatives. To assess task performance, we measure the cumulative reward. All metrics are outlined in detail in § A.

Hyperparameter tuning and selection. We split each dataset into train, validation, and test episodes (70/15/15 for D1) and train 3 seeds for every hyperparameter configuration. For every table cell, we select the configuration with the highest mean validation Equiv. Acc and report its metrics on the validation set. Appendix G lists, for every table and figure, which hyperparameters were fixed, shown, or selected.

Baselines. We compare against TD-MPC2 (Hansen et al., 2024), DreamerV3 (Hafner et al., 2025), and LeWM (Maes et al., 2026), which we reproduced from the authors' implementations (Appendix D.1). LeWM is trained on D2, while TD-MPC2 and DreamerV3 learn online; the world2vec models in Table 1 are trained on our random-policy dataset D1.

Symbolic regression in world2vec. To allow world2vec to discover interpretable latent space dynamics, we integrate our framework with SINDy Brunton et al. (2016). The resulting world2vec-SINDy algorithm represents the flow field of a dynamical system $\dot{\boldsymbol{z}} = f(\boldsymbol{z}) \in \mathbb{R}^d$. Across m timepoints, the encoded states are collected in $\mathbf{Z} \in \mathbb{R}^{m \times d}$ and their derivatives in $\dot{\mathbf{Z}} \in \mathbb{R}^{m \times d}$. From $\mathbf{Z}$, SINDy constructs a polynomial feature library $\boldsymbol{\Theta}(\mathbf{Z}) = [\mathbf{1}, \mathbf{Z}, \mathbf{Z}^{[\mathrm{P}_2]}, \ldots, \mathbf{Z}^{[\mathrm{P}_q]}] \in \mathbb{R}^{m \times p}$, where $\mathbf{Z}^{[\mathrm{P}_k]}$ contains all degree-k monomials. The latent dynamics are represented as $\dot{\mathbf{Z}} = \boldsymbol{\Theta}(\mathbf{Z})\boldsymbol{\Xi}$, where $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d] \in \mathbb{R}^{p \times d}$ is a sparse coefficient matrix whose columns define the governing equations of the individual latent variables, such that $\boldsymbol{\Theta}(\boldsymbol{z})\boldsymbol{\Xi} \approx f(\boldsymbol{z})$. We compare this dynamics model to identity dynamics, which provide the trivial baseline of, $\hat{\boldsymbol{z}}_{t+1} = \boldsymbol{z}_t$, and to an MLP dynamics model that parameterizes the continuous-time flow field as $\dot{\boldsymbol{z}} = f_\theta^{\mathrm{MLP}}(\boldsymbol{z})$. We evaluate all three dynamics models on 71 strange chaotic attractors from the `dysts` database Gilpin (2021), a benchmark previously used to evaluate SINDy Kaptanoglu et al. (2023). However, unlike this previous benchmark, we do not assume access to the physical state, but instead infer latent dynamics from nonlinearly mixed high-dimensional observations by jointly learning the observation-to-latent mapping. Full details of the experimental setup are provided in Appendix I.

Implementation world2vec is implemented in PyTorch. For the considered control tasks, small models with 64×64 frame inputs are sufficient. On a single NVIDIA A100 GPU, training takes 2–6 minutes for a one-step model, about 10 minutes for a 20-step model, and about 30 minutes for a 40-step model, and closed-loop control takes about 20 ms per control step, including simulation and rendering. Beyond the empirical performance, this efficiency matters for downstream applications in robotics and closed-loop experimental settings.

5 RESULTS

Table 1: Main results. Embedding Consistency, Equivariance R^2 and Equivariance Acc. (%) are averaged over Pendulum, Reacher and Cartpole per model seed, then reported as mean ± standard deviation over three seeds (selected on validation Equiv. Acc. per environment). Cons. is one value per three-seed ensemble. Reward (dm_control return, maximum 1000) is reported per environment: mean ± standard deviation over model seeds of the selected models' best planner; Reacher uses the hard task. DreamerV3, TD-MPC2 and LeWM are pixel models evaluated with the same metrics (3–4 runs each); they have no retrieval metric. [a,b]Only one of three model seeds planned (a: seed 42, b: seed 789); [c]one run.

					Reward		
Method	Regularizer	Cons.	Equiv. R^2	Equiv. Acc	pendulum	reacher	cartpole
DreamerV3 (Hafner et al., 2025)		51.4	$50.5_{\pm 4.3}$		$668_{\pm 323}$	$792_{\pm 57}$	$899_{\pm 14}$
TD-MPC2 (Hansen et al., 2024)		92.0	$24.5_{\pm 4.9}$		$911_{\pm 4}$	$744_{\pm 94}$	$894_{\pm 21}$
LeWM (Maes et al., 2026)		99.2	0.0		1[c]	$48_{\pm 25}$	$151_{\pm 5}$
Baseline	+ SIGReg	94.2	$11.3_{\pm 1.3}$	$13.7_{\pm 3.9}$	$275_{\pm 213}$	$306_{\pm 60}$	$208_{\pm 9}$
	+ DCL	92.2	$56.7_{\pm 1.7}$	$8.5_{\pm 3.8}$	$0_{\pm 0}$	462[b]	$163_{\pm 51}$
world2vec	+ SIGReg	96.3	$95.0_{\pm 0.6}$	$74.3_{\pm 0.5}$	$656_{\pm 243}$	625[a]	$231_{\pm 3}$
	+ DCL	95.1	$96.4_{\pm 0.0}$	$81.9_{\pm 1.2}$	836[a]	951[a]	$298_{\pm 14}$

Table 2: **Training horizon.** World2vec with Euler ODE dynamics, a three-frame encoder and 128 negatives, trained with 1 step (single-step prediction, the baseline setting) up to 40 steps of latent rollout; the horizon is also given in seconds, since the model step differs between environments. Per environment and horizon, the latent width (and SIGReg weight) with the best validation Equiv. Acc. is shown; mean ± standard deviation over three model seeds (%; Cons. is one value per ensemble). Equiv. R^2 and Equiv. Acc. after a 1 s rollout; state decoding: MLP decoder, mean R^2 over state coordinates. — = not trained.

(a) world2vec-dcl

Environment	k (Δt)	Cons.	Equiv. R^2	Equiv. Acc	State R^2
pendulum	1 (0.02 s)	93.0	$79.9_{\pm 1.1}$	$9.6_{\pm 1.2}$	$93.1_{\pm 0.7}$
	10 (0.2 s)	99.6	$93.6_{\pm 2.0}$	$74.4_{\pm 7.3}$	$98.1_{\pm 0.2}$
	20 (0.4 s)	99.8	$96.7_{\pm 0.3}$	$87.7_{\pm 1.4}$	$98.1_{\pm 0.2}$
	40 (0.8 s)	99.7	$94.6_{\pm 0.2}$	$90.0_{\pm 0.2}$	$98.3_{\pm 0.1}$
reacher	1 (0.1 s)	96.1	$90.3_{\pm 6.1}$	$15.3_{\pm 10.0}$	$70.9_{\pm 4.2}$
	10 (1 s)	91.7	$96.8_{\pm 0.5}$	$95.8_{\pm 0.2}$	$94.2_{\pm 1.7}$
	20 (2 s)	86.3	$95.7_{\pm 0.9}$	$95.3_{\pm 0.2}$	$93.4_{\pm 1.0}$
cartpole	1 (0.02 s)	87.6	0.0	$0.6_{\pm 0.2}$	$72.3_{\pm 3.9}$
	10 (0.2 s)	97.9	$94.1_{\pm 0.3}$	$19.2_{\pm 0.5}$	$69.7_{\pm 2.4}$
	20 (0.4 s)	96.9	$96.0_{\pm 0.6}$	$32.2_{\pm 2.1}$	$90.8_{\pm 0.4}$
	40 (0.8 s)	88.3	$96.6_{\pm 0.5}$	$44.0_{\pm 1.1}$	$87.3_{\pm 1.4}$

(b) world2vec-sigreg

Environment	k (Δt)	Cons.	Equiv. R^2	Equiv. Acc	State R^2
pendulum	1 (0.02 s)	93.6	$0.6_{\pm 1.1}$	$24.9_{\pm 11.2}$	$99.0_{\pm 0.1}$
	10 (0.2 s)	—	—	—	—
	20 (0.4 s)	95.3	$90.7_{\pm 0.9}$	$72.7_{\pm 4.3}$	$98.6_{\pm 0.1}$
	40 (0.8 s)	97.0	$94.3_{\pm 1.3}$	$89.0_{\pm 1.5}$	$98.8_{\pm 0.1}$
reacher	1 (0.1 s)	92.0	$33.3_{\pm 3.8}$	$13.1_{\pm 0.7}$	$75.7_{\pm 1.2}$
	10 (1 s)	—	—	—	—
	20 (2 s)	93.8	$96.9_{\pm 0.4}$	$95.0_{\pm 0.2}$	$91.1_{\pm 0.3}$
cartpole	1 (0.02 s)	97.0	0.0	$3.0_{\pm 0.6}$	$39.6_{\pm 0.0}$
	10 (0.2 s)	—	—	—	—
	20 (0.4 s)	99.2	$90.2_{\pm 0.5}$	$27.5_{\pm 0.1}$	$42.6_{\pm 0.6}$
	40 (0.8 s)	98.1	$93.7_{\pm 0.3}$	$39.0_{\pm 0.4}$	$40.3_{\pm 0.1}$

world2vec faithfully models the world and enables long-range predictions Across all three environments world2vec is able to predict future latent states much better than the SOTA baselines. world2vec achieves an average equivariance R^2 score of $\geq 95\%$ at 1-second ahead predictions, whereas the SOTA baselines are unable to reliably predict future latent states at this horizon (DreamerV3: 50.5%; TD-MPC2: 24.5%; LeWM: 0.0%). Even comparing world2vec's performance on pixel observations against the SOTA baselines' performance (Fig. 6) when receiving the environment state as inputs, world2vec outperforms TD-MPC2 and DreamerV3 by >20% points in equivariance R^2. From the SOTA baselines, only LeWM demonstrates high equivariance R^2 at >98% when receiving environment states as inputs rather than pixel observations.

world2vec achieves state-of-the-art performance in control tasks after being trained from offline data. On both the Reacher and Pendulum environments, we can use model predictive control with world2vec to solve the environments' tasks. We measure cumulative reward on those environments that match the performance of DreamerV3 and TD-MPC2 and we improve over LeWM's task rewards by >800 points. On the cartpole environment, we are still outperforming LeWM, but we don't solve the task sufficiently well yet and lag behind DreamerV3 and TD-MPC2.

Latent world models learn highly consistent representations across multiple model trainings. In addition to producing more faithful models of the environment's future states, our results also demonstrate that latent world models learn representations that stay consistent across model trainings. Both world2vec and LeWM produce embeddings with an average consistency score across environments of > 95%. TD-MPC2 and DreamerV3 on the other hand, only achieve consistency scores of 92.0% and 51.4%, respectively.

A sufficiently large training horizon is essential for latent world modeling. By far the most important differentiator in the training setup that makes world2vec outperform current state-of-the-art latent world models is introducing an autoregressive multi-step rollout as part of the loss. Across all three environments, we observe that increasing the training horizon drastically improves the model's performance (Tab. 2). This holds for both world2vec+SigReg (Tab. 2b) and world2vec+DCL (Tab. 2a). For world2vec+SigReg, increasing the horizon from 1 step to 20 steps improves the equivariance R^2 from 0.6% to 90.7% for pendulum, from 33.3% to 96.9% for reacher, and from 0.0% to 90.2% for cartpole.

Parameterizing the dynamics as an ODE stabilizes training and improves performance. The second important modeling choice to be made is how the dynamics model should be parametrized. We test both a discrete-time parametrization in which we use $z_{t+1} = MLP(z_t, a_t)$ or alternatively a continuous-time parametrization $\frac{dz}{dt} = MLP(z_t, a_t)$ in combination with Euler integration. Across both world2vec+SigReg and world2vec+DCL the Euler ODE formulation clearly improves over

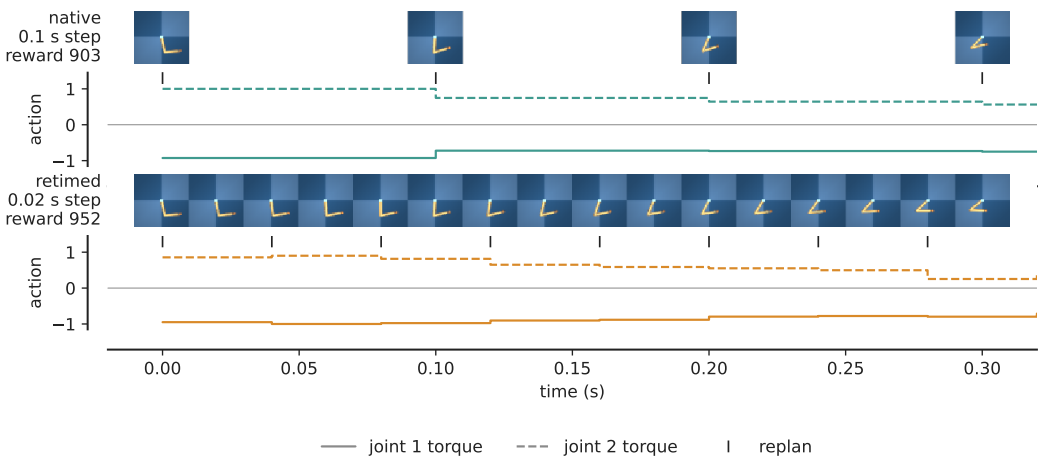

Figure 2: **Control across time scales (Reacher, hard).** The same world2vec-DCL model (20 training steps), start state and target, planned at the 0.1 s step it was trained at (top, *native*) and retimed to 0.02 s by shrinking the Euler step (bottom, *retimed*). Frames are the planner's observations, drawn at the time they were taken (the target is not shown to the planner); under each strip, the two joint torques applied and a tick at every replanning (every 0.1 s native, every 0.04 s retimed). Episode returns: 903 (native) and 952 (retimed).

the discrete MLP dynamics. In the regimes of lower-dimensional latent spaces, MLP dynamics can still be used to learn dynamics that outperform the SOTA baselines in terms of equivariance R^2, achieving an average of 94.6% on world2vec+DCL and 12d latents and 79.2% on world2vec+SigReg at 32 latent dimensionality. But across the board, the Euler ODE formulation results in higher Consistency scores, higher equivariance R^2, higher equivariance Accuracy, and better state decoding when comparing the best-performing models across latent dimensionalities.

Table 3: **Discrete-time vs. continuous-time dynamics.** World2vec trained with 20 rollout steps and a three-frame encoder, with discrete MLP transitions ($z_{t+1} = \mathrm{MLP}(z_t, a_t)$) or Euler-integrated ODE dynamics, at latent width $d \in \{12, 32, 128\}$. Per environment, the number of negatives (DCL) or the SIGReg weight with the best validation Equiv. Acc. is kept. Scores are averaged over the three environments per model seed, then mean $\pm$ standard deviation over three seeds (%). — = not trained.

(a) world2vec-dcl

Dynamics	d	Cons.	Equiv. R^2	Equiv. Acc	State R^2
Discrete-time	12	90.0	$94.6_{\pm 0.5}$	$65.9_{\pm 0.8}$	$75.5_{\pm 0.7}$
	32	28.1	$53.1_{\pm 23.9}$	$15.8_{\pm 15.0}$	$39.4_{\pm 23.6}$
	128	2.3	$0.8_{\pm 0.1}$	$0.3_{\pm 0.0}$	$11.1_{\pm 1.7}$
Continuous-time	12	94.8	$96.9_{\pm 0.1}$	$70.8_{\pm 0.1}$	$88.6_{\pm 1.3}$
	32	98.4	$96.4_{\pm 0.1}$	$76.6_{\pm 0.3}$	$95.3_{\pm 0.1}$
	128	97.3	$96.2_{\pm 0.2}$	$77.1_{\pm 0.5}$	$95.8_{\pm 0.1}$

(b) world2vec-sigreg

Dynamics	d	Cons.	Equiv. R^2	Equiv. Acc	State R^2
Discrete-time	12	—	—	—	—
	32	86.8	$79.2_{\pm 16.3}$	$36.8_{\pm 10.9}$	$64.7_{\pm 5.7}$
	128	—	—	—	—
Continuous-time	12	—	—	—	—
	32	96.0	$93.4_{\pm 0.3}$	$65.0_{\pm 1.0}$	$76.5_{\pm 0.1}$
	128	96.4	$91.0_{\pm 1.5}$	$46.9_{\pm 8.9}$	$72.7_{\pm 0.5}$

ODE parameterization allows flexible sampling. An additional benefit of using a continuous-time dynamics model is that this allows us to switch to a different sampling rate at inference time. We demonstrate this on the reacher environment (Fig. 2): The world2vec model for this environment was originally trained on a sampling rate of 10Hz (0.1 seconds between any two observations). We can use this model at its native sampling rate to drive the environment with a new action every 0.1 seconds. However, given an ODE dynamics model, we can also sample observations at a higher frequency of 50Hz (the native frequency of the MuJoCo environment) when performing Model Predictive Control and decrease the dt of the Euler Integrator accordingly. Repeating the same action for two steps allows us to drive the environment with a new control step at 25Hz, which enables us to match the control frequencies that were used to evaluate LeWM, TD-MPC2 and DreamerV3. In addition to providing more flexibility in specifying the MPC configuration, this also improves the average cumulative reward from 947 ± 26 to 955 ± 15.

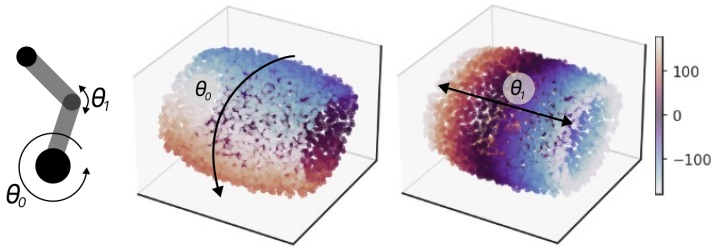

Figure 3: First three principal components of the world2vec representation on the reacher task.

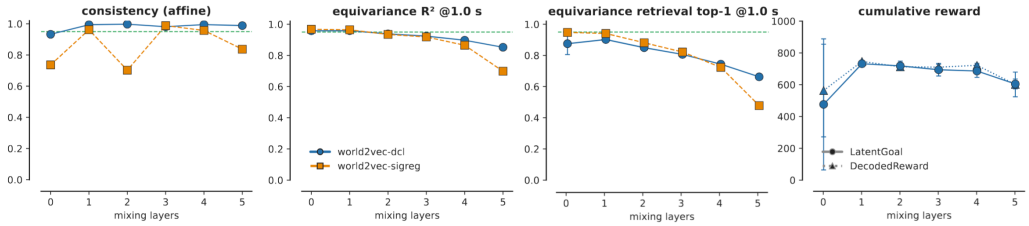

Figure 4: Comparison of world2vec across increasingly non-linear representations of state of CartPole.

Embedding topology qualitatively reflects environment structure As a qualitative assessment of the coordinates learned by world2vec, we visualized a PCA projection of the embeddings obtained on the Reacher environment (Fig 3). This environment can be transcribed through two angles θ_0, θ_1 describing the current configuration of the two arms. While θ_0 is executing full 360 degree movements through data collection, θ_1 is constrained to moving left and right relative to the first arm; full circular movements are not observed with our data selection process. This is reflected in the embedding, where θ_0 is encoded with circular boundaries, while θ_1 is represented linearly with open boundaries.

Empirical identifiability of world2vec Identifiable representation learning denotes the property where two models converge to a representation. This property has been studied extensively in the context of representation learning and world models, and for the given objective, we conjecture that the representations derived from training will be consistent up to an affine transformation; and dynamics will be modeled faithfully over time.

We analyzed the performance of the two best performing models from Table 1 on the most challenging CartPole task, which was not solvable from pixels alone for any of the considered models or baselines. Under a systematic increase in task difficulty, we observe a decline in different metrics. Equivariance R^2 and retrieval performance measured via Equivariane Accuracy both decline from 100% when the state is known (0 mixing layers) to values around 50% to 85% as we increase the difficulty of the nonlinear mixing to 5 nonlinear layers. world2vec-dcl performs best and achieves an Eq. R^2 above 80% at 5 mixing layers (Fig. 4). Cumulative reward slightly declines, but on this synthetic experiments we are generally able to solve the cartpole task using world2vec up to a reward of 600.

6 WORLD2VEC FOR MECHANISTIC DYNAMICAL INFERENCE

By integrating SINDy into world2vec, we enable our framework to learn interpretable models of latent dynamics, rather than using a black-box latent state forecaster based on an MLP vector field. We present a proof-of-concept evaluation of world2vec-SINDy and compare it with an identity baseline and an MLP vector field across four of the metrics described in Appendix A.

world2vec-SINDy outperforms the identity baseline and recovers the same performance as full MLP parameterization across 71 dynamical systems (Fig. 5a). For most systems, the learned SINDy and MLP rollouts agree with both the encoder's future embeddings and the affinely aligned ground-truth rollouts, as measured by equivariance R^2_{50} and iDynR^2_{50}, respectively.

In Figure 5b, we provide ablations across four noise levels, $\alpha \in \{0, 0.1, 0.5, 1.0\}$, showing medians and interquartile ranges (25th–75th percentiles) across systems for the two rollout metrics as well

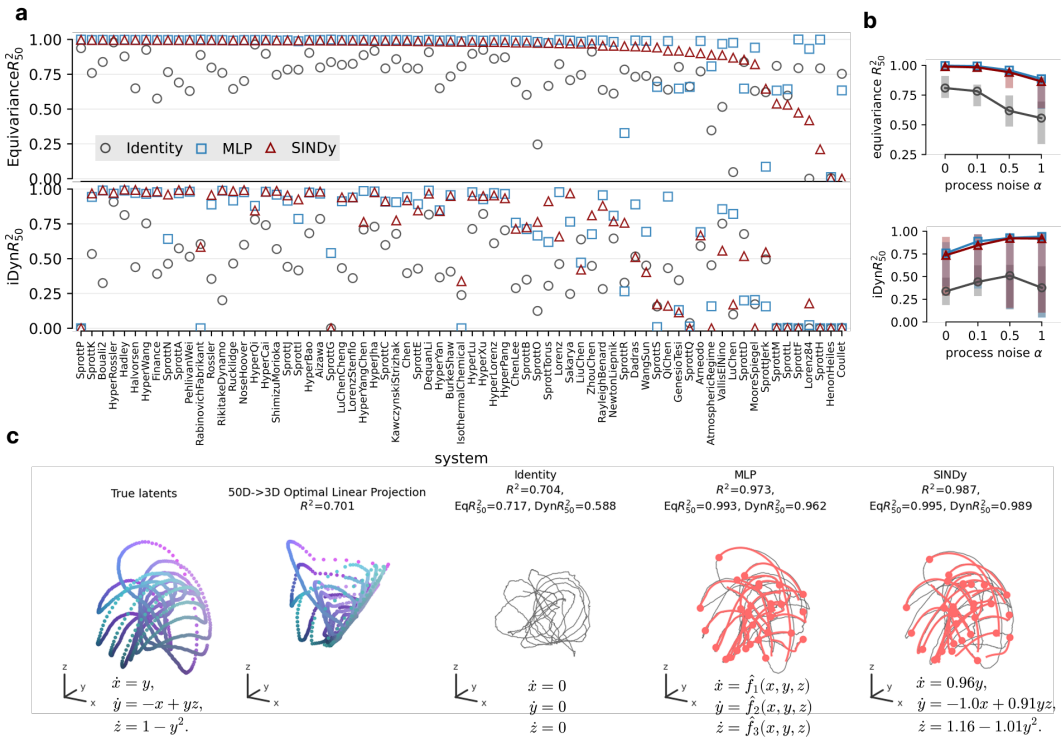

Figure 5: Latent dynamics recovery across all 71 dysts systems (Gilpin, 2021). **a**, Equivariance R^2_{50} ($\mathrm{Eq}R^2_{50}$) and $\mathrm{iDyn}R^2_{50}$ of 50-step predictions with encoded futures and true rollouts after affine alignment, respectively (mean over three seeds at process noise $\alpha = 0.1$). **b**, Lines and shaded bars show medians and 25th–75th percentiles across different noise levels. **(c)** Qualitative result on SprottA, true states and their optimal linear reconstruction from 50-dimensional non-linear observations; de-mixing with identity baseline, MLP and SINDy dynamics.

as for all four metrics in Figure 7b. The ablations show that increasing process noise improves representation recovery (see Fig.7b) but reduces rollout equivariance.

Unlike the black-box flow field of the MLP, SINDy recovers interpretable equations from the nonlinearly mixed high-dimensional observations that closely match the ground-truth equations after affine alignment and thresholding. Qualitative examples are shown in (Fig. 5c); to obtain the equations, we currently linearly align the representation to the ground truth states, which would be unavailable in practice. This is followed by pruning coefficients with magnitude below 0.1. In extensions of this work, it would be interesting to repeat this experiment with the unsupervised symbolic regression technique discussed by Muratore & Mathis (2026).

7 CONCLUSION

We introduced world2vec, a framework for learning mechanistic world models by jointly fitting representations and latent dynamics from observations. Across the evaluated control environments, multistep rollout training and an ODE parameterization improve long-range prediction and representation consistency while supporting control from pixels using offline data. Experiments with nonlinear observation mixing further demonstrate the recovery of the underlying state, with the choice of representation objective influencing recovery quality. World2vec allows regularization of the training using both contrastive learning and LeJEPA, and extends to mechanistically modeling dynamical systems through explicit parameterization of the dynamics model. These results show how jointly choosing the representation objective and dynamics model can connect world modeling for control with mechanistic inference, providing a foundation for future applications in high stakes applications and scientific discovery. While we are to the best of our knowledge first to present a stable world model that can control systems with autonomous dynamics, studied environments, dynamical systems and observation processes for purely data-driven world models remain limited. Given the success application on the considered benchmarks, we plan to extend world2vec to these more complex environments in future work.

AI USE STATEMENT

We used generative AI tools to assist with drafting and editing the manuscript, implementing and running experimental code, and creating plotting code for the figures. All AI-assisted text and code were reviewed by the authors. Experimental outputs and figures were checked for correctness, and the authors take full responsibility for the final content of this work.

REFERENCES

Randall Balestriero and Yann LeCun. LeJEPA: Provable and Scalable Self-Supervised Learning Without the Heuristics, 2025. URL https://arxiv.org/abs/2511.08544.

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, March 2016. ISSN 1091-6490. doi: 10.1073/pnas.1517384113. URL http://dx.doi.org/10.1073/pnas.1517384113.

Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116 (45):22445–22451, October 2019. ISSN 1091-6490. doi: 10.1073/pnas.1906995116. URL http://dx.doi.org/10.1073/pnas.1906995116.

L Gao and J Nathan Kutz. Bayesian autoencoders for data-driven discovery of coordinates, governing equations and fundamental constants. *arXiv preprint arXiv:2211.10575*, 2022.

William Gilpin. Chaos as an interpretable benchmark for forecasting and data-driven modelling. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/ec5decca5ed3d6b8079e2e7e7bacc9f2-Paper-round2.pdf.

Rodrigo González Laiz, Tobias Schmidt, and Steffen Schneider. Self-supervised contrastive learning performs non-linear system identification, 2025. URL https://arxiv.org/abs/2410.14673.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, April 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2. URL http://dx.doi.org/10.1038/s41586-025-08744-2.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control, 2024. URL https://arxiv.org/abs/2310.16828.

Alan A Kaptanoglu, Lanyue Zhang, Zachary G Nicolaou, Urban Fasel, and Steven L Brunton. Benchmarking sparse system identification with low-dimensional chaos: Aa kaptanoglu et al. *Nonlinear Dynamics*, 111(14):13143–13164, 2023.

David Klindt, Yann LeCun, and Randall Balestriero. When Does LeJEPA Learn a World Model?, 2026. URL https://arxiv.org/abs/2605.26379.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels, 2026. URL https://arxiv.org/abs/2603.19312.

Fabian A Mikulasch and Friedemann Zenke. Understanding self-supervised learning via latent distribution matching. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=schXkGSTus.

Paolo Muratore and Mackenzie Weygandt Mathis. Extracting governing equations from latent dynamics via multi-view contrastive learning. *arXiv preprint arXiv:2606.13260*, 2026.

Ingmar Posner, Anson Lei, and Bernhard Schölkopf. From observation to insight: Mechanistic world models and the quest for autonomous discovery, 2026. URL https://arxiv.org/abs/2607.12474.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite, 2018. URL https://arxiv.org/abs/1801.00690.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, pp. 9929–9939. PMLR, 2020.

Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive Learning Inverts the Data Generating Process. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12979–12990. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/zimmermann21a.html.

A METRICS

Equivariance R^2 measures the agreement of the encoder and dynamics models in an unsupervised way. Given a sequence of observations $\{\boldsymbol{o}_t\}_{t=0}^{T}$, we encode these sequence into latent space using the encoder $\boldsymbol{z}_t = h_\phi(\boldsymbol{o}_t)$. The latent dynamics model predicts $\hat{\boldsymbol{z}_{t+k}} = f_\theta^k(\boldsymbol{z}_{t+k})$ and we compute the equivariance R^2 for horizon k as the score between $\boldsymbol{z}_{t+k}$ and $\hat{\boldsymbol{z}_{t+k}}$.

Equivariance Accuracy (Equiv. Acc.) measures how well we can recover a future observation via KNN decoding in an unsupervised way. Follows the same principle as equivariance R^2. We encode a sequence of observations; we predict the future latents $\hat{\boldsymbol{z}_{t+k}}$ but then we do a KNN lookup to identify the embedding in latent space that is closest to $\hat{\boldsymbol{z}_{t+k}}$. To be able to do this, we randomly sample N=512 observations from the training dataset, we add the observations $\{\boldsymbol{o}_t\}_{t=0}^{T}$ we compute our metric on to the pool of candidates. We encode these into latent space, find the closest latent, compute a Top-k accuracy score on this classification task, and report Top-1 accuracy.

Embedding Consistency (Cons.) measures how well multiple models (different seeds) trained on the same data agree with each other. This is done by taking the same set of observations, encoding them into latent space with each model $\boldsymbol{Z}^{(k)} = h_{\phi_k}(\boldsymbol{O})$. Then we fit a linear regression to map $\boldsymbol{Z}^{(i)} \to \boldsymbol{Z}^{(j)}$ for every $i, j \in [1, ..., K]$, we score how well each of those mappings works via R^2 and average these scores to arrive at the embedding consistency score.

Cumulative Reward measures how well a model does on solving a given task. We use the reward as measured by the MuJoCo-based DeepMind control environments: Pendulum (swingup), Reacher (hard), Cartpole (swingup). We set episode length and dt to a) match the setup for LeWM as used in Maes et al. (2026) and b) constrain per-episode reward to be within 0 and 1000 as maximum cumulative episode reward.

State Decoding measures how well we can decode the state provided by the environments. We split the val dataset into two parts, one to train a decoder (Linear Regression or MLP decoder) and the second part on which we evaluate these.

Identifiable Latents R^2 (iLatR^2) measures how well the encoder recovers the true latent representation up to an affine transform. This is the metric as it is proposed and used in DCL (González Laiz et al., 2025) and corresponds to state decoding via an affine map (linear regression).

Identifiable Dynamics R^2 (iDynR^2) measures how well the recovered dynamics model aligns with the ground truth latent dynamics models. This metric corresponds to the "DynR^2" metric proposed for DCL (González Laiz et al., 2025) at horizon k. As part of this metric, we fit an affine map from the encoder's embeddings into the ground truth latent space. The same map as the Identifiable Latents R^2 metric uses. However, instead of scoring the embeddings, we score how well future embeddings align with the ground truth dynamics.

B DESIGN-CHOICE ABLATIONS

Table 4 summarizes an extended design-choices sweep not yet discussed in the main paper. Besides the sweep over the latent dimensions, we additionally also run experiments with varying observation sequence lengths. We ran world2vec-dcl with varying number of negatives sampled into a given batch. And of course for every world2vec-SIGReg model we trained with varying choices of weighting factors for the regularizer term.

Our ablations on the temporal window (observation history) suggest that for these environments a history of length 3 is optimal. A longer history helps neither on Equiv. R^2 nor on Equiv. accuracy.

For the number of negatives using with the world2vec-dcl objective we can observe a clear trend that suggest that more negatives improve the model. Or at a minimum don't hurt performance. At some level, the benefit gained from more negatives plateaus. This means this parameter doesn't need to be tuned necessarily. These results suggest, just picking the highest possible number of negatives that the compute hardware (and time constraints) allow for is the optimal choice.

Table 4: **Design choices on D1.** Per environment and setting, the configuration with the best validation Equiv. Acc.; scores averaged over environments per model seed, then mean ± standard deviation over seeds (%; Cons. is one value per ensemble). Reward as in Table 1, averaged over environments; – = no model seed planned in all three environments. Blocks use different slices of the grid; compare within a block. Model seeds out of 9: [a]6 (pendulum 2, reacher 1 of 3); [b]6 (cartpole 1, pendulum 2 of 3); [c]8 (reacher 2 of 3).

Axis	Value	Cons.	Equiv. R^2	Equiv. Acc	Reward
latent width	state dim	90.7	$97.1_{\pm 0.1}$	$73.1_{\pm 0.5}$	–[a]
	12	94.8	$96.9_{\pm 0.1}$	$70.8_{\pm 0.1}$	—
	32	96.1	$96.4_{\pm 0.1}$	$75.8_{\pm 0.8}$	—
	128	93.2	$96.0_{\pm 0.0}$	$76.6_{\pm 0.3}$	—
temporal window	1 frame + 3-latent history	77.4	$85.0_{\pm 3.0}$	$32.0_{\pm 0.3}$	—
	3 frames	96.1	$96.4_{\pm 0.1}$	$75.8_{\pm 0.8}$	—
	5 frames	95.3	$95.3_{\pm 0.2}$	$69.7_{\pm 0.6}$	—
	9 frames	96.1	$92.3_{\pm 0.3}$	$67.8_{\pm 1.0}$	—
negatives	128	91.4	$95.6_{\pm 0.2}$	$76.4_{\pm 0.3}$	—
	512	93.3	$96.4_{\pm 0.2}$	$81.6_{\pm 1.2}$	—
	1024	94.9	$96.5_{\pm 0.1}$	$81.4_{\pm 0.8}$	696.7[b]
SIGReg weight	0.01	85.4	$14.0_{\pm 6.6}$	$9.0_{\pm 3.6}$	—
	0.1	89.4	$84.3_{\pm 17.7}$	$48.2_{\pm 11.5}$	—
	1	96.3	$95.0_{\pm 0.6}$	$74.3_{\pm 0.5}$	$507.3_{\pm 96.6}$[c]

Table 5: **Dataset sensitivity.** D1: our environment rollouts; D2: the baselines' data. Averaged over environments per model seed, then mean ± standard deviation over seeds (%; reward as in Table 1); *seeds* = complete seeds. Model steps are 0.02/0.1 s (D1) and 0.02/0.04 s (D2); prediction is compared at 1 s. Model seeds out of 9: [a]8 (reacher 2 of 3); [b]3 (one per environment).

	D1 (our rollouts)					D2 (baselines' data)				
Regularizer	Cons.	Equiv. R^2	Equiv. Acc	Reward	seeds	Cons.	Equiv. R^2	Equiv. Acc	Reward	seeds
+ SIGReg	96.3	$95.0_{\pm 0.6}$	$74.3_{\pm 0.5}$	$507.3_{\pm 96.6}$[a]	3	94.5	$93.1_{\pm 1.0}$	$60.9_{\pm 2.2}$	—	3
+ DCL	95.1	$96.4_{\pm 0.0}$	$81.9_{\pm 1.2}$	$699.1_{\pm 4.8}$	3	89.8	$94.5_{\pm 0.2}$	$81.2_{\pm 0.5}$	178.7[b]	3

C Dataset sensitivity

Table 5 compares the two sources of pixel observations at the same physical prediction horizon.

D Datasets

This section was generated by Python scripts directly from the experiment database logs and the code base used to run the experiments.

All pixel world models are trained on one of two datasets of the same three DeepMind Control Suite environments (Tassa et al., 2018), rendered as 64×64 RGB frames (Table 6). **D1** is our own collection with random policies only (no expert data). The torques are: cartpole, Ornstein–Uhlenbeck noise ($\theta = 0.2$, $\sigma = 0.35$) clipped to [-1, 1], pushing the cart back beyond ±1.6 m; pendulum, Ornstein–Uhlenbeck noise ($\theta = 0.2$, $\sigma = 0.35$) clipped to [-1, 1]; reacher, uniform random torques in [-1, 1]. Episodes start from: cartpole, uniform (position ±1.8 m, any angle, velocities ±5 m/s and ±10 rad/s); pendulum, a uniform angle and an angular velocity uniform in ±7 rad/s; reacher, the environment default, with the target not rendered. **D2** is the training data of our reproduction of the baselines (below), collected following the LeWM protocol (Maes et al., 2026): per environment, 1,000 episodes of a random policy and 1,000 episodes of a TD-MPC2 expert policy (Hansen et al., 2024) trained on the task, 100 steps each. The random policy is an Ornstein–Uhlenbeck process ($\theta = 0.15$, $\sigma = 0.35$) with intermittent high-energy bursts for broad coverage of angles and velocities (in 75% of the cartpole episodes and 40% of the pendulum and reacher episodes); on cartpole a position keeper softly holds the cart within ±0.85 m. Initial states have uniform angles and near-zero

velocities. Episodes are split whole into train / validation / test (70% / 15% / 15% for D1, 85% / 15% / 0% for D2); model selection and all reported model metrics use the validation episodes.

Table 6: Pixel datasets. Rate: frames per second of simulated time (one learned model step per frame).

Dataset	Environment	Frames	Episodes × length	Rate	Train / val / test episodes
D1 (ours)	cartpole (swingup)	200,000	200 × 1,000	50 Hz	140 / 30 / 30
	pendulum (swingup)	200,000	1,000 × 200	50 Hz	700 / 150 / 150
	reacher (easy)	400,000	10,000 × 40	10 Hz	7,000 / 1,500 / 1,500
D2 (reproduction)	cartpole (swingup)	200,000	2,000 × 100	50 Hz	1,700 / 300 / 0
	pendulum (swingup)	200,000	2,000 × 100	25 Hz	1,700 / 300 / 0
	reacher (hard)	200,000	2,000 × 100	25 Hz	1,700 / 300 / 0

The nonlinear-mixing experiments replace the frames by a fixed random map of the simulator state (Table 7), following (González Laiz et al., 2025). A mixing depth ℓ is a frozen, randomly initialised invertible MLP with ℓ layers (leaky ReLU), followed by a frozen random linear map to 50 dimensions; depth 0 observes the state directly. Either the full state is mixed, or only its position coordinates, so that velocities must be inferred from several observations. The data-collection policies are those of D1.

Table 7: Nonlinear-mixing datasets (one dataset per environment, observed coordinates and depth).

Environment	Observed	Episodes × length	Rate	Mixing depths
cartpole	full state	200 × 1,000	50 Hz	0, 1, 2, 3, 4, 5
	positions only	200 × 1,000	50 Hz	0, 1, 2, 3, 4, 5
pendulum	full state	1,000 × 200	50 Hz	0, 1, 2, 3, 4, 5
	positions only	1,000 × 200	50 Hz	0, 1, 2, 3, 4, 5
reacher	full state	10,000 × 40	10 Hz	0, 1, 2, 3, 4, 5
	positions only	10,000 × 40	10 Hz	0, 1, 2, 3, 4, 5

D.1 Baselines

We reproduced TD-MPC2 (Hansen et al., 2024), DreamerV3 (Hafner et al., 2025) and LeWM (Maes et al., 2026) from the authors' implementations and default configurations, on 64×64 frames for TD-MPC2 and DreamerV3 and 224×224 frames with a three-frame context for LeWM (four seeds each). TD-MPC2 and DreamerV3 learn online, for up to one million environment steps with their original optimisation settings and no per-task tuning, and are evaluated with their learned controllers. LeWM is trained on D2 for 50 epochs, with the SIGReg weight selected per environment (0.001 for cartpole, 0.1 for pendulum and reacher); our reproduction reaches an 81.5% success rate on reacher, against 86% reported originally. LeWM is controlled by post-hoc MPC; its original terminal latent cost did not solve our tasks, so it plans with the distance of decoded states to the goal at every predicted step. Rewards are the sum of the native rewards over 500-step episodes. The baselines' equivariance R^2 and consistency are computed on a separate evaluation set of 1,000 episodes of 100 steps per environment, collected with the random policy of D2.

E Model architectures

This section was generated by Python scripts directly from the experiment database logs and the code base used to run the experiments.

Every model has an encoder, which maps a short window of observations to a latent state $\mathbf{z} \in \mathbb{R}^{d_z}$, and a dynamics model, which advances the latent under a control $\mathbf{u}$. The training objectives add no parameters. Table 8 lists the blocks of the main model (three frames, $d_z = 32$, one control): the encoder has **220,769** parameters and the Euler ODE dynamics **42,272**, together **263,041**. The Euler ODE predicts $\hat{\mathbf{z}}_{t+1} = \mathbf{z}_t + \Delta t\, f_\theta([\mathbf{z}_t, \mathbf{u}_t])$ with the model step Δt of the data; its last layer is initialised to zero, so an untrained model predicts no change. The MLP step predicts $\hat{\mathbf{z}}_{t+1} = g_\theta([\mathbf{z}_t, \mathbf{u}_t])$ directly. Counts are exact, from models built from the stored training configurations.

Table 8: Blocks of the main model: three-frame pixel encoder, $d_z = 32$, one control. Output sizes exclude the batch.

Part	Block	Layers	Output	Parameters
Encoder, per frame	patch embedding	16×16 patches, linear to 64, learned position embedding	16 × 64	50,240
	transformer layer 1	pre-norm, 4 heads, feed-forward 256, GELU	16 × 64	49,984
	transformer layer 2	pre-norm, 4 heads, feed-forward 256, GELU	16 × 64	49,984
	final LayerNorm		16 × 64	128
	keypoint readout	per channel: expected (x, y) under a softmax over the 4×4 grid	128	1
Encoder, window	head MLP	concatenated 3 frames → 128 → 128 → 32 (LayerNorm, GELU)	32	70,432
Dynamics	Euler ODE	$[\mathbf{z}, \mathbf{u}]$ → 128 → 128 → 128 → 32 (LayerNorm, ELU)	32	42,272
	MLP step	same widths, Mish	32	42,272
	Euler ODE, latent history	last 3 latents and $\mathbf{u}$ → same widths (ELU)	32	50,464

Table 9 gives the parameter counts of every pixel architecture that was trained. The encoder grows with the number of frames and with d_z; the dynamics with d_z and the number of controls (one for cartpole and pendulum, two for reacher). D1 and D2 models share the architectures.

Table 9: Exact parameter counts of all trained pixel architectures (Euler ODE and MLP step have identical counts).

Encoder frames	d_z	Encoder	Dynamics (1 / 2 controls)	Total (1 / 2 controls)
1 + 3-latent history	32	188,001	50,464 / 50,592	238,465 / 238,593
3	3	217,028	34,819 / –	251,847 / –
	5	217,286	35,333 / –	252,619 / –
	6	217,415	– / 35,718	– / 253,133
	12	218,189	37,132 / 37,260	255,321 / 255,449
	32	220,769	42,272 / 42,400	263,041 / 263,169
	128	233,153	66,944 / 67,072	300,097 / 300,225
5	32	253,537	42,272 / 42,400	295,809 / 295,937
9	32	319,073	42,272 / 42,400	361,345 / 361,473

Table 10: Parameters of the selected models behind each row of the main tables, per environment. Several values: the row selects one configuration per training horizon or latent width.

Table	Row	cartpole	pendulum	reacher
1	World2vec + DCL	300,097	263,041	300,225
	World2vec + SIGReg	263,041	263,041	263,169
	Baseline + DCL	252,619	251,847	253,133
	Baseline + SIGReg	263,041	263,041	263,169
2a	World2vec + DCL	252,619 / 300,097	251,847 / 263,041 / 300,097	253,133 / 300,225
2b	World2vec + SIGReg	263,041 / 300,097	263,041	263,169
3	World2vec + DCL	255,321 / 263,041 / 300,097	255,321 / 263,041 / 300,097	255,449 / 263,169 / 300,225
	World2vec + SIGReg	263,041 / 300,097	263,041 / 300,097	263,169 / 300,225

The nonlinear-mixing models use an MLP encoder over the concatenated window of observations (3 hidden layers of width 128, LayerNorm, GELU) whose output width is the true state dimension (cartpole 5, pendulum 3, reacher 6), and the same Euler ODE dynamics (3 hidden layers of width 128). They have between 69,510 and 96,012 parameters, depending on window, state dimension and observation width.

F MODEL PREDICTIVE CONTROL

This section was generated by Python scripts directly from the experiment database logs and the code base used to run the experiments.

Closed-loop control uses model predictive control in the learned latent space. A plan is a sequence of H decisions, each held for k model steps, so it covers $L = Hk$ predicted steps. From the latent of the most recent observation window, every candidate plan is rolled out with the dynamics model, $\hat{\mathbf{z}}_j = F_\theta(\hat{\mathbf{z}}_{j-1}, \mathbf{u}_{j-1})$, and scored by

$$J = \sum_{j=1}^{L} w_j \left\| \mathbf{W}(\hat{\mathbf{z}}_j - \mathbf{z}^\star) \right\|_2^2, \qquad \sum_j w_j = 1, \tag{12}$$

where $\mathbf{z}^\star$ is the goal latent, $\mathbf{W}$ is the identity or, with whitening, the inverse square root of the covariance of the training latents, and the weights w_j either weight all steps equally, only the last step(s), or ramp up linearly from the first to the last step. The planner executes the first e controls of the best plan, observes, encodes the newest window and plans again from scratch. The search is either random search (sample candidates uniformly, keep the best) or the cross-entropy method (CEM: sample from a Gaussian, refit it to the best fraction of candidates, repeat, and return the best candidate seen). **Goals.** For cartpole and pendulum the goal latent is the encoding of a rendered window of the target state (upright, at rest, cart centred). For reacher, whose target moves every episode and is not rendered, the goal is the arm configuration that places the fingertip on the target (closed-form inverse kinematics on a fixed elbow side), rendered and encoded; the two elbow sides are planned separately. These pixel planners are reward-free: they use neither a reward nor a decoded state. The planners of the nonlinear-mixing figure instead use training-set rewards: either as goal the mean latent of the 64 best-rewarded training frames, or a cost equal to the negative of a reward decoded from the latent (a small MLP fitted on training data). The environment reward is used only to score finished episodes (cumulative reward, at most 1000). **Retiming.** The continuous-time model can plan at a finer model step than it was trained at: for reacher (trained at 0.1 s) the Euler step is set to 0.02 s and the encoder's frame window is rescaled to span the same seconds, without retraining (Figure 2).

Table 11 lists the settings of all planners behind a reported reward. Where the planner was selected among several (Table 1: the best planner per environment on its selected models), the table shows the selected ones; for reacher this selection includes the elbow side. The CEM initial standard deviation is half the action range and its minimum 10^{-3} throughout.

Table 11: Planner settings behind every reported reward (one column per table / figure and planner; several values: the selection kept planners that differ).

Setting	Tab. 1 cartpole	Tab. 1 pendulum	Tab. 1 reacher	Fig. 2 native	Fig. 2 retimed	Fig. 4 decoded reward	Fig. 4 latent distance
goal / cost	rendered target state	rendered target state	target configuration (per episode)	target configuration (per episode)	target configuration (per episode)	decoded reward	best-rewarded training frames
step weighting w	linear ramp 0 $\rightarrow$ 1	linear ramp 0 $\rightarrow$ 1	uniform	uniform	uniform	last 5 steps / last step	last 15 steps / last 5 steps / last step
whitening	no	no	no	no	no	–	yes
search	CEM	CEM	CEM	CEM	CEM	random search	random search
actions	continuous	continuous	continuous	continuous	continuous	{-1, 0, 1}	{-1, 0, 1}
candidates per iteration	512	512	512	512	512	4096	4096
CEM iterations / elite fraction	6 / 0.125	6 / 0.125	6 / 0.125	6 / 0.125	6 / 0.125	–	–
decisions H $\times$ hold k	12 $\times$ 4	12 $\times$ 4	10 $\times$ 4	5 $\times$ 2	10 $\times$ 4	25 $\times$ 2	25 $\times$ 2
executed controls e	2	2	2	1	2	1	1
model step (s)	0.02	0.02	0.02	0.1	0.02	0.02	0.02
lookahead (s)	0.96	0.96	0.8	1	0.8	1	1
episodes $\times$ steps	150 $\times$ 500	150 $\times$ 1000 / 50 $\times$ 1000	50 $\times$ 1000	49 $\times$ 200 / 50 $\times$ 200	17 $\times$ 1000 / 46 $\times$ 1000 / 50 $\times$ 1000	5 $\times$ 500	5 $\times$ 500
model seeds	3	1	1	3	3	3	3

G HYPERPARAMETERS

This section was generated by Python scripts directly from the experiment database logs and the code base used to run the experiments.

For every table and figure, this appendix states which model hyperparameters went into it: which were fixed, which are shown, which are averaged over, and which were **selected**, i.e. chosen per table cell by the model-selection rule. It is generated from the exact sets of trained models each table was selected from. Selection always maximises the validation-split mean over a configuration's three training seeds; the selection metric is the equivariance accuracy (Equiv. Acc) at 1 s. Hyperparameters fixed throughout are listed once (Table 12); quantities that follow from others (e.g. the model step from the environment, the input width of the dynamics from d_z) are omitted. Planner settings are listed in Appendix F.

Table 12: Hyperparameters with a single value across all reported models (varies: see the per-table summaries; –: does not apply).

Component	Hyperparameter	Pixel world models	Nonlinear-mixing models
data	environment	varies	cartpole
	observed coordinates	–	positions only
objective	training horizon K	varies	20
	SIGReg projections	1024	64
	temperature τ	1	1
	negatives	varies	1024
encoder	encoder	pixel	MLP
	latent width d_z	varies	5
	encoder frames T	varies	3
	patch size	16	–
	transformer width	64	–
	transformer layers	2	–
	attention heads	4	–
	encoder MLP width	128	128
	encoder MLP hidden layers	2	3
dynamics	dynamics	varies	Euler ODE
	latent history	3	–
	dynamics MLP width	128	128
	dynamics MLP hidden layers	3	3
optimization	optimizer	Adam	Adam
	learning rate, encoder	0.001	0.001
	learning rate, dynamics	0.001	0.001
	learning-rate schedule	cosine decay	cosine decay
	weight decay	0	0
	batch size	128	2048
	training iterations	20000	20000

G.1 TABLE 1: SOTA COMPARISON

- **world2vec + DCL** (D1; objective = DCL, dynamics = Euler ODE, training horizon $K \in \{10, 20, 40\}$, encoder frames $T = 3$): 141 trained models; per environment, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.
- **world2vec + SIGReg** (D1; objective = SIGReg, dynamics = Euler ODE, training horizon $K \in \{10, 20, 40\}$, encoder frames $T = 3$): 90 trained models; per environment, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.
- **Baseline + DCL** (D1; objective = DCL, dynamics = Euler ODE, training horizon $K = 1$, encoder frames $T = 3$): 27 trained models; per environment, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.

- **Baseline + SIGReg** (D1; objective = SIGReg, dynamics = Euler ODE, training horizon K = 1, encoder frames T = 3): 54 trained models; per environment, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.

Table 13: Table 1: SOTA comparison: hyperparameters not fixed across the paper, per row block.

Hyperparameter	world2vec + DCL	world2vec + SIGReg	Baseline + DCL	Baseline + SIGReg
environment	averaged over: cartpole, pendulum, reacher	averaged over: cartpole, pendulum, reacher	averaged over: cartpole, pendulum, reacher	averaged over: cartpole, pendulum, reacher
objective	shown: DCL	shown: SIGReg	shown: DCL	shown: SIGReg
training horizon K	selected from {10, 20, 40}	selected from {20, 40}	1	1
SIGReg weight λ	–	selected from {0.01, 0.1, 1}	–	selected from {0.01, 0.1, 1}
negatives	selected from {128, 512, 1024}	128	128	128
latent width d_z	selected from {3, 5, 6, 12, 32, 128}	selected from {32, 128}	selected from {3, 5, 6, 32, 128}	selected from {32, 128}
encoder frames T	3	3	3	3
dynamics	Euler ODE	Euler ODE	Euler ODE	Euler ODE

Table 14: Table 1: SOTA comparison: the selected values per selection cell.

Row	Cell	training horizon K	negatives	latent width d_z	SIGReg weight λ
world2vec + DCL	cartpole	40	512	128	–
	pendulum	40	1024	32	–
	reacher	20	1024	128	–
world2vec + SIGReg	cartpole	40	–	32	1
	pendulum	40	–	32	1
	reacher	20	–	32	1
Baseline + DCL	cartpole	–	–	5	–
	pendulum	–	–	3	–
	reacher	–	–	6	–
Baseline + SIGReg	cartpole	–	–	32	0.1
	pendulum	–	–	32	0.01
	reacher	–	–	32	1

G.2 TABLE 2A: TRAINING HORIZON, DCL

- **world2vec + DCL** (D1; objective = DCL, dynamics = Euler ODE, encoder frames T = 3, negatives = 128): 108 trained models; per environment, training horizon K, the configuration with the best validation Equiv. Acc.

Table 15: Table 2a: training horizon, DCL: hyperparameters not fixed across the paper, per row block.

Hyperparameter	world2vec + DCL
environment	shown: cartpole, pendulum, reacher
objective	DCL
training horizon K	shown: 1, 10, 20, 40
negatives	128
latent width d_z	selected from {3, 5, 6, 12, 32, 128}
encoder frames T	3
dynamics	Euler ODE

G.3 TABLE 2B: TRAINING HORIZON, SIGREG

- **world2vec + SIGReg** (D1; objective = SIGReg, dynamics = Euler ODE, encoder frames T = 3, negatives = 128): 144 trained models; per environment, training horizon K, the configuration with the best validation Equiv. Acc.

G.4 TABLE 3: MLP STEP VS EULER ODE, DCL

- **world2vec + DCL** (D1; objective = DCL, training horizon K = 20, encoder frames T = 3, latent width $d_z \in \{12, 32, 128\}$, dynamics $\in$ {MLP step, Euler ODE}): 108 trained models;

Table 16: Table 2a: training horizon, DCL: the selected values per selection cell.

Row	Cell	latent width d_z
world2vec + DCL	cartpole, 1	5
	cartpole, 10	5
	cartpole, 20	128
	cartpole, 40	128
	pendulum, 1	3
	pendulum, 10	32
	pendulum, 20	32
	pendulum, 40	128
	reacher, 1	6
	reacher, 10	128
	reacher, 20	128

Table 17: Table 2b: training horizon, SIGReg: hyperparameters not fixed across the paper, per row block.

Hyperparameter	world2vec + SIGReg
environment	shown: cartpole, pendulum, reacher
objective	SIGReg
training horizon K	shown: 1, 20, 40
SIGReg weight λ	selected from {0.01, 0.1, 1}
negatives	128
latent width d_z	selected from {32, 128}
encoder frames T	3
dynamics	Euler ODE

per environment, dynamics, latent width d_z, the configuration with the best validation Equiv. Acc.

G.5 Table 3: MLP step vs Euler ODE, SIGReg

- **world2vec + SIGReg** (D1; objective = SIGReg, training horizon $K = 20$, encoder frames $T = 3$, latent width $d_z \in \{12, 32, 128\}$, dynamics $\in$ {MLP step, Euler ODE}): 81 trained models; per environment, dynamics, latent width d_z, the configuration with the best validation Equiv. Acc.

G.6 Figure 2: reacher retiming

- **native 0.1 s step** (D1; latent width $d_z = 32$): 3 trained models; reward over all 10 planners on these models.
- **retimed to 0.02 s** (D1; latent width $d_z = 32$): 3 trained models; reward over all 14 planners on these models.

G.7 Figure 4: nonlinear mixing

- **cartpole, positions only** (mixing data; objective $\in$ {DCL, SIGReg}): 126 trained models; per mixing depth, objective, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.

G.8 Table 4: design choices

- **latent width** (D1; objective = DCL, dynamics = Euler ODE, encoder frames $T = 3$, negatives = 128): 108 trained models; per environment, latent width d_z, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.
- **temporal window** (D1; objective = DCL, dynamics $\in$ {Euler ODE, Euler ODE + latent history}, latent width $d_z = 32$, negatives = 128): 84 trained models; per environment, encoder frames T, dynamics, latent history, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.

Table 18: Table 2b: training horizon, SIGReg: the selected values per selection cell.

Row	Cell	SIGReg weight λ	latent width d_z
world2vec + SIGReg	cartpole, 1	0.1	32
	cartpole, 20	1	128
	cartpole, 40	1	32
	pendulum, 1	0.01	32
	pendulum, 20	1	32
	pendulum, 40	1	32
	reacher, 1	1	32
	reacher, 20	1	32

Table 19: Table 3: MLP step vs Euler ODE, DCL: hyperparameters not fixed across the paper, per row block.

Hyperparameter	world2vec + DCL
environment	averaged over: cartpole, pendulum, reacher
objective	DCL
training horizon K	20
negatives	selected from {128, 512, 1024}
latent width d_z	shown: 12, 32, 128
encoder frames T	3
dynamics	shown: MLP step, Euler ODE

- **negatives** (D1; objective = DCL, dynamics = Euler ODE, encoder frames $T = 3$, latent width $d_z \in \{32, 128\}$, training horizon $K \in \{20, 40\}$): 90 trained models; per environment, negatives, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.

- **SIGReg weight** (D1; objective = SIGReg, dynamics = Euler ODE, encoder frames $T = 3$): 144 trained models; per environment, SIGReg weight λ, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.

G.9 TABLE 5: D1 VS D2

- **D1 (ours)** (D1; dynamics = Euler ODE, training horizon $K \in \{10, 20, 40\}$, encoder frames $T = 3$): 231 trained models; per environment, objective, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.

- **D2 (reproduction)** (D2; dynamics = Euler ODE, training horizon $K \in \{10, 20, 40\}$, encoder frames $T = 3$): 270 trained models; per environment, objective, the configuration with the best validation Equiv. Acc; reward from the best planner on the selected models.

Table 20: Table 3: MLP step vs Euler ODE, DCL: the selected values per selection cell.

Row	Cell	negatives
world2vec + DCL	cartpole, MLP step, 12	128
	cartpole, MLP step, 32	1024
	cartpole, MLP step, 128	128
	cartpole, Euler ODE, 12	128
	cartpole, Euler ODE, 32	1024
	cartpole, Euler ODE, 128	1024
	pendulum, MLP step, 12	128
	pendulum, MLP step, 32	128
	pendulum, MLP step, 128	128
	pendulum, Euler ODE, 12	128
	pendulum, Euler ODE, 32	1024
	pendulum, Euler ODE, 128	1024
	reacher, MLP step, 12	128
	reacher, MLP step, 32	1024
	reacher, MLP step, 128	128
	reacher, Euler ODE, 12	128
	reacher, Euler ODE, 32	1024
	reacher, Euler ODE, 128	1024

Table 21: Table 3: MLP step vs Euler ODE, SIGReg: hyperparameters not fixed across the paper, per row block.

Hyperparameter	world2vec + SIGReg
environment	averaged over: cartpole, pendulum, reacher
objective	SIGReg
training horizon K	20
SIGReg weight λ	selected from {0.01, 0.1, 1}
negatives	128
latent width d_z	shown: 32, 128
encoder frames T	3
dynamics	shown: MLP step, Euler ODE

Table 22: Table 3: MLP step vs Euler ODE, SIGReg: the selected values per selection cell.

Row	Cell	SIGReg weight λ
world2vec + SIGReg	cartpole, MLP step, 32	1
	cartpole, Euler ODE, 32	0.1
	cartpole, Euler ODE, 128	1
	pendulum, MLP step, 32	1
	pendulum, Euler ODE, 32	1
	pendulum, Euler ODE, 128	1
	reacher, MLP step, 32	1
	reacher, Euler ODE, 32	1
	reacher, Euler ODE, 128	1

Table 23: Figure 2: reacher retiming: hyperparameters not fixed across the paper, per row block.

Hyperparameter	native 0.1 s step	retimed to 0.02 s
environment	reacher	reacher
objective	DCL	DCL
training horizon K	20	20
negatives	1024	1024
latent width d_z	32	32
encoder frames T	3	3
dynamics	Euler ODE	Euler ODE

G.10 Glossary

H Rollouts

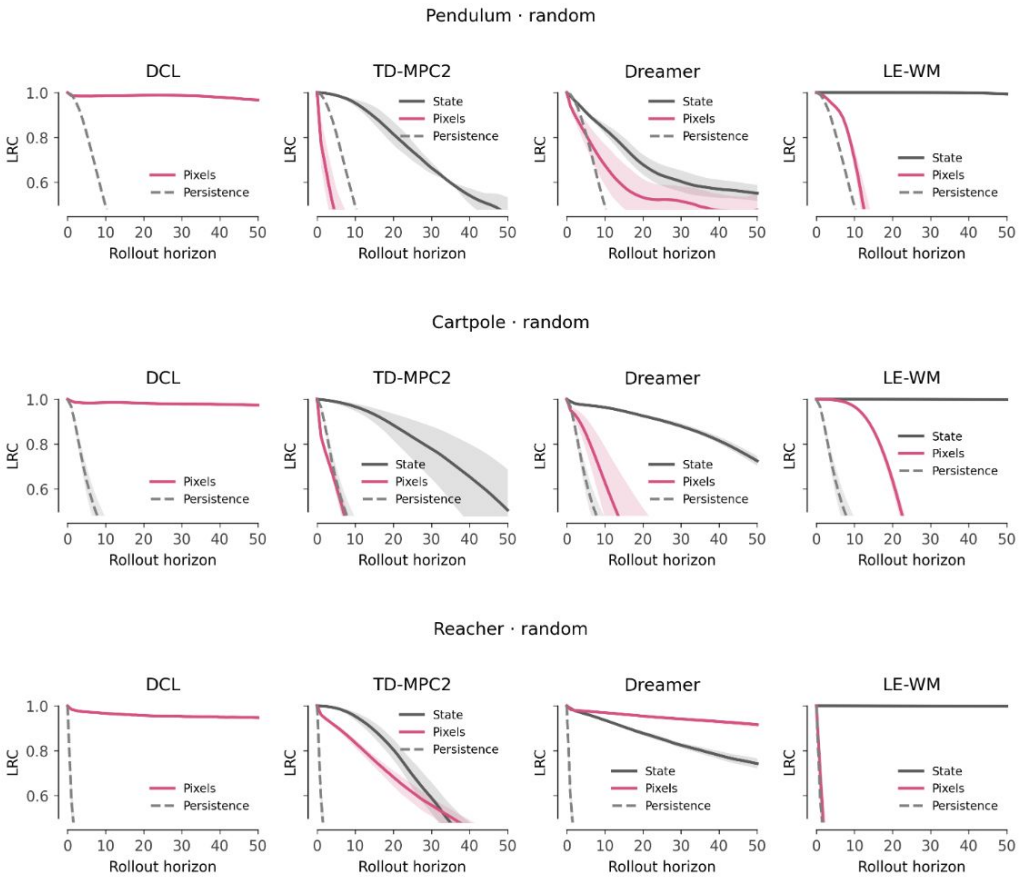

Figure 6: **Latent rollout behavior on Cartpole under random actions.** The comparison shows DCL, TD-MPC2, Dreamer, and LeWM. Pink curves use pixel observations, gray curves use state observations where available, and dashed curves show persistence. The DCL pixel curve remains near its initial score across the displayed 50-step horizon, while the other pixel curves decline rapidly. LRC denotes latent rollout consistency, the equivariance R^2 used in the text. *Figure edit: relabel LRC as Equiv. R^2.*

Table 24: Figure 4: nonlinear mixing: hyperparameters not fixed across the paper, per row block.

Hyperparameter	cartpole, positions only
mixing depth	shown: 0, 1, 2, 3, 4, 5
objective	shown: SIGReg, DCL
SIGReg weight λ	selected from {1e-05, 0.0001, 0.001, 0.01, 0.1, 1}

Table 25: Figure 4: nonlinear mixing: the selected values per selection cell.

Row	Cell	SIGReg weight λ
cartpole, positions only	0, SIGReg	0.01
	0, DCL	–
	1, SIGReg	0.01
	1, DCL	–
	2, SIGReg	0.01
	2, DCL	–
	3, SIGReg	0.001
	3, DCL	–
	4, SIGReg	0.001
	4, DCL	–
	5, SIGReg	0.01
	5, DCL	–

I SINDy

I.1 Methods

The learned latent dynamics are evaluated on 71 systems from the `dysts` collection Gilpin (2021). For each system, we generate 1,000 ground-truth trajectories of 1,000 steps using a fourth-order Runge–Kutta (RK4) integrator with a step size Δt_s equal to ten times the system-specific value defined in Gilpin (2021). To choose the initial conditions, we first simulate a noise-free, 1,000-step reference trajectory from the system's canonical initial state, then discard the first 100 states, determine the minimum and maximum values reached by each state coordinate over the remaining trajectory, and draw each coordinate of a new initial condition uniformly between these bounds. Trajectories containing nonfinite states or exceeding 10^6 in absolute value are resampled.

After each RK4 step, independent Gaussian noise is added to the simulated physical state: $\boldsymbol{x}_{t+1} = \Phi_{\Delta t_s}(\boldsymbol{x}_t) + \boldsymbol{\epsilon}_t$. For state variable $i \in \{1, \ldots, d\}$, the noise has standard deviation $\alpha \sigma_{x,i} \sqrt{\Delta t_s / T_s}$, where $\sigma_{x,i}$ is that variable's attractor standard deviation reported in the `dysts` metadata and T_s is the system's reference period. We use $\alpha \in \{0, 0.1, 0.5, 1.0\}$, with $\alpha = 0$ giving noise-free trajectories.

This is followed by applying a fixed, *injective* mixing function $\boldsymbol{g} : \mathbb{R}^d \mapsto \mathbb{R}^n$ to the true latent trajectory $\boldsymbol{x} \in \mathbb{R}^d$, yielding observed time series $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^n$. Following Zimmermann et al. (2021), the mixing function consists of a fixed four-layer MLP ($\text{MLP}_{l=4} : \mathbb{R}^d \mapsto \mathbb{R}^d$) followed by a linear projection to $n = 50$ dimensions. Entire trajectories, rather than individual time points, are assigned to training, validation, and test sets in a 70:15:15 ratio. All models use a three-layer MLP encoder with 128 hidden units, trained with the one-step DCL objective to learn a representation $\boldsymbol{z} = h(\boldsymbol{y})$.

Three dynamics models are evaluated on the generated datasets. Identity dynamics provide a baseline without a learned dynamics model: $\hat{\boldsymbol{z}}_{t+1} = \boldsymbol{z}_t$. The MLP dynamics model parameterizes a continuous-time flow field, $\dot{\boldsymbol{z}} = f_\theta^{\text{MLP}}(\boldsymbol{z})$, whose one-step prediction is obtained by RK4 integration. Finally, the SINDy model represents the flow field using a polynomial feature library $\boldsymbol{\Theta}$, such that for encoded states across m timepoints, $\mathbf{Z} \in \mathbb{R}^{m \times d}$, $\boldsymbol{\Theta}(\mathbf{Z}) = [\mathbf{1}, \mathbf{Z}, \mathbf{Z}^{[\mathrm{P}_2]}, \ldots, \mathbf{Z}^{[\mathrm{P}_q]}] \in \mathbb{R}^{m \times p}$ and $\dot{\mathbf{Z}} = \boldsymbol{\Theta}(\mathbf{Z})\boldsymbol{\Xi}$. $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_d] \in \mathbb{R}^{p \times d}$, then is a sparse coefficient matrix whose columns correspond to the governing equations of the individual state variables and $\mathbf{Z}^{[\mathrm{P}_k]}$ contains all degree-k monomials, with the `dysts` system-specific maximum degree $q \in \{2, 3\}$ being assumed to be known. Finally, an L_1 penalty, $\lambda_{\text{SINDy}} \|\boldsymbol{\Xi}\|_1$ with $\lambda_{\text{SINDy}} = 10^{-5}$, encourages sparse coefficients, and (unlike for other SINDy implementations Brunton et al. (2016); Champion et al. (2019); Gao & Kutz (2022)), no coefficient thresholding is applied during training.

Table 26: Table 5: D1 vs D2: hyperparameters not fixed across the paper, per row block.

Hyperparameter	D1 (ours)	D2 (reproduction)
environment	averaged over: cartpole, pendulum, reacher	averaged over: cartpole, pendulum, reacher
objective	shown: SIGReg, DCL	shown: SIGReg, DCL
training horizon K	selected from {10, 20, 40}	selected from {10, 20, 40}
SIGReg weight λ	selected from {0.01, 0.1, 1}	selected from {0.01, 0.1, 1}
negatives	selected from {128, 512, 1024}	selected from {128, 512, 1024}
latent width d_z	selected from {3, 5, 6, 12, 32, 128}	selected from {3, 5, 6, 12, 32, 128}
encoder frames T	3	3
dynamics	Euler ODE	Euler ODE

Table 27: Table 5: D1 vs D2: the selected values per selection cell.

Row	Cell	training horizon K	SIGReg weight λ	negatives	latent width d_z
D1 (ours)	cartpole, SIGReg	40	1	128	32
	cartpole, DCL	40	–	512	128
	pendulum, SIGReg	40	1	128	32
	pendulum, DCL	40	–	1024	32
	reacher, SIGReg	20	1	128	32
	reacher, DCL	20	–	1024	128
D2 (reproduction)	cartpole, SIGReg	40	1	128	32
	cartpole, DCL	40	–	1024	32
	pendulum, SIGReg	40	1	128	32
	pendulum, DCL	20	–	1024	128
	reacher, SIGReg	20	0.1	128	32
	reacher, DCL	20	–	1024	128

The performance of the resulting models is assessed across four metrics. Affine recovery R^2 measures how well the physical state can be recovered from the learned representation. Consistency quantifies mean pairwise affine agreement between representations learned from three different model seeds. iDynR^2_{50} compares ground-truth and learned 50-step rollouts after affine alignment, while equivariance R^2_{50} compares learned rollouts with encodings of the observed future.

All models are trained for 10,000 iterations and we tune only the weight of the contrastive log-denominator term λ_{unif} over a grid of $\lambda_{\text{unif}} \in \{0.01, 0.1, 0.5, 1.0\}$. During analysis, for all systems, (except for the qualitative rolllout plots), we select the ensemble across uniformity weights that has the highest mean equivariance R^2_{50} on the validation set and then report the results on the validation set.

For qualitative inspection of rollouts and recovered equations in Fig. 5, we first rank the recovered SINDy systems by equivariance R^2_{50} and retain only the top ten systems. Among these, we select the system whose learned SINDy models are most consistent across seeds. Because each seed learns its own latent coordinate system, we first express all three SINDy models in a common space, choosing the seed r with the highest validation equivariance R^2_{50} as the reference. For each seed $j \neq r$, we fit the affine transformation $\boldsymbol{z}^{(r)} \approx L_{j\to r}\boldsymbol{z}^{(j)} + \boldsymbol{b}_{j\to r}$ by least squares on paired embeddings, as motivated by the affine identifiability result in Theorem 1 of González Laiz et al. (2025). We then transform the SINDy coefficient matrix learned by seed j into the reference coordinates according to $\widehat{\boldsymbol{\Xi}}_{j\to r} = M_{j\to r}^{\top}\widehat{\boldsymbol{\Xi}}_j L_{j\to r}^{\top}$, where $M_{j\to r}$ describes how the polynomial feature library transforms under the affine coordinate change, $\boldsymbol{\theta}\big(L_{j\to r}^{-1}\left(\boldsymbol{z}^{(r)} - \boldsymbol{b}_{j\to r}\right)\big) = M_{j\to r}\boldsymbol{\theta}\big(\boldsymbol{z}^{(r)}\big)$, and $\boldsymbol{\theta}$ denotes the column vector of polynomial features evaluated at a single state. The reference model remains unchanged, $\widehat{\boldsymbol{\Xi}}_{r\to r} = \widehat{\boldsymbol{\Xi}}_r$.

Once all three models are expressed in the same coordinate system, we quantify their agreement using the mean pairwise Frobenius distance between their coefficient matrices, $D = \frac{1}{3}\sum_{1\le j<k\le 3}\left\|\widehat{\boldsymbol{\Xi}}_{j\to r} - \widehat{\boldsymbol{\Xi}}_{k\to r}\right\|_F$. Among the candidate systems, we select the one with the smallest D, corresponding to the SINDy models with the highes agreement across seeds. Importantly, both the alignment and the coefficient-distance criterion depend only on the learned embeddings and learned SINDy coefficients and therefore do not require access to the ground-truth latent states or dynamics.

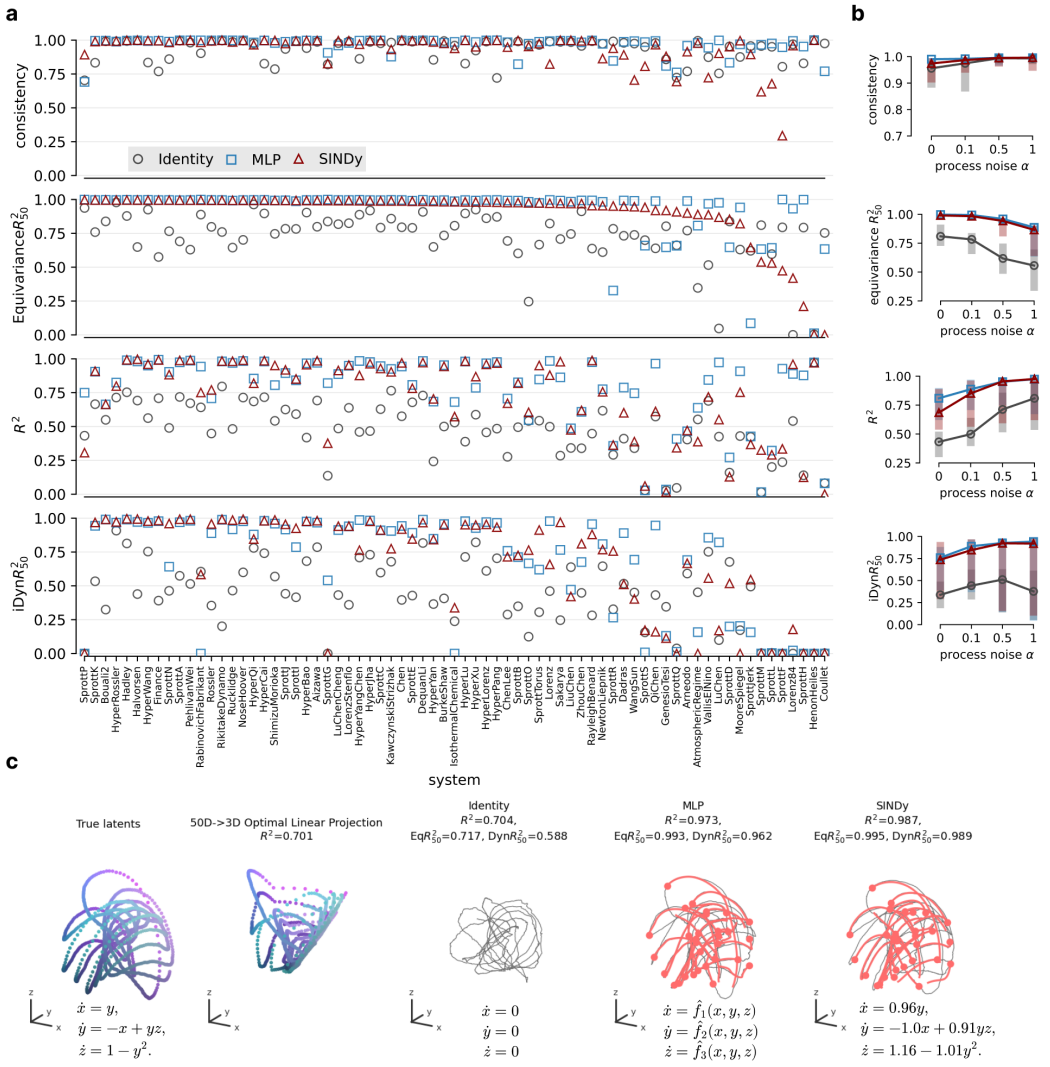

Figure 7: Latent dynamics recovery, representation consistency, and affine state recovery across all 71 `dysts` systems. (**a**) Consistency of ensembles is displayed on top, followed by equivariance R^2_{50} (EqR^2_{50}), affine recovery R^2, and iDynR^2_{50}. EqR^2_{50} and iDynR^2_{50} compare 50-step predictions with encoded futures and true rollouts after affine alignment, respectively. The points show means over three seeds for 71 `dysts` systems at process-noise level $\alpha = 0.1$, ordered by decreasing EqR^2_{50} of SINDy models. Identity, MLP and SINDy dynamics models are shown as grey circles, blue squares and red triangles, respectively. (**b**) Lines and shaded bars show medians and 25th–75th percentiles across systems for $\alpha \in \{0, 0.1, 0.5, 1.0\}$ for consistency, EqR^2_{50}, R^2, and iDynR^2_{50}. (**c**) For the selected system, SprottA, true states and their optimal linear reconstruction from 50-dimensional observations share corresponding point colours; grey curves show learned embeddings and red curves show predicted rollouts, with dots marking initial states, all in ground-truth coordinates. For each dynamics model, as well as the true system, the equations governing the dynamics are displayed in the bottom of the rollout figures.

Table 28: Hyperparameters used in this appendix.

Hyperparameter	Meaning
environment	cartpole, pendulum or reacher
observed coordinates	nonlinear mixing of the full state or of the positions only
mixing depth	layers of the random mixing network (0: no mixing)
objective	DCL (contrastive, InfoNCE) or SIGReg (JEPA prediction loss with the SIGReg regularizer)
training horizon K	model steps of the autoregressive rollout in the training loss
SIGReg weight λ	weight of the SIGReg regularizer
SIGReg projections	random 1-D projections tested by SIGReg
temperature τ	temperature of the InfoNCE scores
negatives	negative samples per batch in the InfoNCE denominator
encoder	pixel patch transformer, or MLP (mixing experiments)
latent width d_z	dimension of the latent state
encoder frames T	consecutive observations the encoder reads
patch size	side of the square image patches (pixels)
transformer width	patch embedding and transformer width
encoder MLP width	hidden width of the MLP from the frames' readouts (pixels) or observations (mixing) to the latent
dynamics	Euler ODE (continuous-time vector field), MLP step (discrete-time), or Euler ODE on a latent history
latent history	latents the history-conditioned vector field reads
learning rate, encoder	peak learning rate
learning rate, dynamics	peak learning rate
batch size	rollout start points per batch

I.2 Full Results

world2vec-SINDy outperforms the identity baseline overall and performs comparably to an MLP vector field. In Figure 5 (**a**), world2vec-SINDy (red) generally outperforms the identity predictor (grey) and performs comparably to the MLP model (blue) across metrics of latent state and dynamics recovery. For latent process noise $\alpha = 0.1$, we recover highly consistent ensembles across all three dynamics model families. For SINDy and MLP, this consistency is accompanied by high affine recovery R^2 for many of the 71 systems. Furthermore, for multiple systems, the learned SINDy and MLP rollouts agree with both the encoder's future embeddings and the affinely aligned ground-truth rollouts, as measured by equivariance R^2_{50} and iDynR^2_{50}, respectively.

Increasing process noise improves representation recovery but reduces rollout equivariance. In Figure 5B, we provide ablations across four noise levels, $\alpha \in \{0, 0.1, 0.5, 1.0\}$, showing medians and interquartile ranges (25th–75th percentiles) across systems for all four metrics. As α increases from 0 to 1.0, median affine recovery R^2 rises from approximately 0.8 to 0.98 for MLP, from 0.7 to 0.97 for SINDy, and from 0.4 to 0.8 for the identity model. Similarly, consistency increases, reaching approximately 0.99 for all three models at $\alpha = 1.0$. In contrast, median equivariance R^2_{50} decreases, while for SINDy and MLP, median iDynR^2_{50} increases, although its interquartile range widens. These results are consistent with prior findings that contrastive learning can benefit from increased noise Wang & Isola (2020), while noise can complicate dynamical inference Brunton et al. (2016).

world2vec-SINDy provides a proof of concept by outperforming MLP dynamics while yielding analytical equations. To provide a proof-of-concept demonstration of how world2vec can recover governing dynamics, we select a system from the `dysts` collection without using ground-truth metrics for selection. Among the ten systems with the highest validation equivariance R^2_{50} for SINDy, the one with the smallest mean pairwise Frobenius distance between affinely aligned coefficient matrices across ensemble members, is chosen as described in Appendix I. In Figure 5C, both the MLP and SINDy models outperform the optimal linear projection and identity baselines in R^2. SINDy also achieves slightly higher scores than MLP on all three metrics: $R^2 = 0.987$ versus 0.973, equivariance $R^2_{50} = 0.995$ versus 0.993, and iDyn$R^2_{50} = 0.989$ versus 0.962, respectively.

Most importantly for our purposes, unlike the black-box flow field of the MLP, SINDy recovers interpretable equations from the nonlinearly mixed high-dimensional observations that closely match

the ground-truth equations after affine alignment and thresholding. We emphasize, however, that this is a proof of concept, as we use knowledge of the ground-truth system, generally unavailable in practice, to transform the discovered equations from embedding space into ground-truth coordinates using an affine map fitted to the true latent states and then set coefficients with magnitude below 0.1 to zero for display clarity. We note that related work (Muratore & Mathis, 2026) explores recovering sparse SINDy models without ground-truth knowledge by optimizing a sparsity objective over affine coordinate transformations, however, here we omit this additional optimization as it is outside the scope of our proof-of-concept demonstration.