# LEARNING HAMILTONIAN DYNAMICS FROM IMAGES FOR LONG-HORIZON PREDICTION

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

## ABSTRACT

Neural networks can generate visually plausible frames while failing to preserve physical consistency over time. Physics-based models introduce physical inductive biases to address this limitation, but inferring a useful state representation from images while simultaneously learning the dynamics remains challenging. We propose **ViHaD** (**Vi**sual **Ha**miltonian **D**ynamics), a framework that jointly learns a low-dimensional state representation from images and a scalar Hamiltonian governing its evolution. To support accurate long-horizon prediction, we penalize latent collapse and minimize prediction error over autonomous rollouts. We evaluate ViHaD on a pendulum spanning libration and rotation, a mass–spring oscillator, and a planar two-body system. Across these systems, ViHaD substantially improves visual prediction accuracy compared with PixelHNN.

## 1 INTRODUCTION

Learning physical dynamics directly from images is an important problem in machine learning, with applications in prediction, simulation, planning, and control. Latent dynamics models have shown that latent state representations and predictive dynamics can be learned from visual observations (Watter et al., 2015; Hafner et al., 2019). Such models can learn the latent representation and its dynamics directly from data without requiring the dynamics to be explicitly constrained by principles from classical mechanics.

A complementary line of work incorporates inductive biases from classical mechanics. Hamiltonian Neural Networks (HNNs), in particular, parameterize a scalar Hamiltonian over canonical states and learn the dynamics by matching the corresponding Hamiltonian vector field to the time derivatives of the state (Greydanus et al., 2019). PixelHNN (Greydanus et al., 2019) and Hamiltonian Generative Networks (Toth et al., 2020) extend this idea to visual observations, showing that Hamiltonian dynamics can be learned in latent spaces inferred from image sequences.

Learning Hamiltonian dynamics from images remains challenging, particularly for long-horizon prediction. Because the physical state is not directly observed, the latent state representation and its dynamics must be learned jointly; accurate reconstruction alone does not ensure that the learned representation is dynamically useful. Supervision based on local dynamics does not by itself ensure a state representation suitable for autonomous prediction, and errors in state inference and learned dynamics can compound under repeated integration. Reliable long-horizon prediction therefore requires both a suitable latent representation and learned dynamics that remain accurate throughout autonomous rollout.

In this work, we introduce **ViHaD** (**Vi**sual **Ha**miltonian **D**ynamics), a framework for long-horizon autonomous prediction from image sequences. ViHaD jointly learns a latent state representation and a scalar Hamiltonian governing its dynamics. We improve local dynamics supervision by using a centered five-point finite-difference estimate of the time derivative of the latent state instead of a one-step forward difference. We also introduce non-collapse constraints to keep the latent coordinates informative and well scaled. Finally, we train with progressively longer closed-loop rollouts to improve autonomous prediction over long horizons.

We evaluate ViHaD on pendulum, mass–spring, and two-body systems with different state dimensions and phase-space structure, using foreground intersection-over-union and pixel mean-squared error on autonomous rollouts.



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.

**Contributions.**

- **Learning Hamiltonian dynamics across motion regimes and state dimensions.** We jointly infer low-dimensional states from images and learn Hamiltonian dynamics in the inferred state space across pendulum, mass–spring, and two-body systems. The pendulum dataset spans libration, bidirectional rotation, and near-separatrix trajectories[1], while the two-body model uses an eight-dimensional latent state.
- **Higher-order local dynamics supervision.** We replace the one-step forward-difference target used in PixelHNN with a centered five-point estimate of the instantaneous latent derivative, matched to the Hamiltonian vector field $J\nabla_z \mathcal{H}_\theta(z_t)$ at the current inferred state.
- **Non-collapse constraints for latent-state formation.** We introduce aggregate-scale and per-coordinate barrier losses that discourage severe contraction of the inferred latent state during joint reconstruction and Hamiltonian learning.
- **Progressive closed-loop rollout training.** We supervise recursively integrated latent and pixel trajectories while progressively increasing the training horizon, separating short-horizon representation formation from longer-horizon refinement.

## 2 Related Work

Modeling visual dynamics from observations has received considerable attention. Interaction Networks and Visual Interaction Networks use object-centric relational structure (Battaglia et al., 2016; Watters et al., 2017), while Embed to Control, Deep Variational Bayes Filters, world models, and PlaNet infer compact latent dynamics from sequential observations (Watter et al., 2015; Karl et al., 2017; Ha & Schmidhuber, 2018; Hafner et al., 2019). Neural ODEs and Latent ODEs provide continuous-time latent evolution (Chen et al., 2018; Rubanova et al., 2019), and Graph Network-based Simulators extend relational modeling to larger interacting systems (Sanchez-Gonzalez et al., 2020). These methods provide flexible frameworks for learning dynamics from data, but generally do not explicitly incorporate physical laws.

Other methods incorporate physics directly into the learning process. Differentiable physics and physics-informed neural networks introduce physical constraints through simulation or differential-equation residuals (Belbute-Peres et al., 2018; Raissi et al., 2019), while Deep Lagrangian Networks and Lagrangian Neural Networks embed Lagrangian mechanics directly into learned dynamics (Lutter et al., 2019; Cranmer et al., 2020). Hamiltonian Neural Networks (HNNs) parameterize a scalar Hamiltonian and derive the vector field through Hamilton's equations (Greydanus et al., 2019). This idea has been extended to interacting systems, controlled dynamics, and non-conservative settings through Hamiltonian Graph Networks, Symplectic ODE-Net, port-Hamiltonian networks, and dissipative Hamiltonian models (Sanchez-Gonzalez et al., 2019; Zhong et al., 2020; Desai et al., 2021; Sosanya & Greydanus, 2022).

A related line of work focuses on structure preservation and generalization. Symplectic Recurrent Neural Networks combine learned Hamiltonian dynamics with symplectic integration (Chen et al., 2020), while SympNets enforce symplectic structure by construction (Jin et al., 2020). Meta-learning approaches study generalization across families of Hamiltonian systems (Lee et al., 2021; Jing et al., 2026). Recent analysis of Hamiltonian Generative Networks in forced and dissipative settings further shows that numerical integration and latent-state behavior can affect temporal generalization (Laird & Clark, 2026).

The closest predecessors to our work are PixelHNN, Hamiltonian Generative Networks (HGN), and HGN++, which learn Hamiltonian dynamics directly from visual observations (Greydanus et al., 2019; Toth et al., 2020; Higgins et al., 2021). PixelHNN combines an autoencoder with a low-dimensional Hamiltonian latent state inferred from consecutive frames. HGN learns an abstract phase-space representation that is evolved through a learned Hamiltonian and decoded back to observations. HGN++ predicts future observations from a short causal context. Higgins et al. (2021) also introduce SyMetric to assess recovery of physical Hamiltonian dynamics. They report improved aggregate pendulum prediction metrics over HGN, but only models for the two mass–spring variants

[1]Libration is bounded oscillatory motion in which the pendulum reverses direction rather than completing full rotations. The separatrix is the phase-space boundary between libration and rotation.

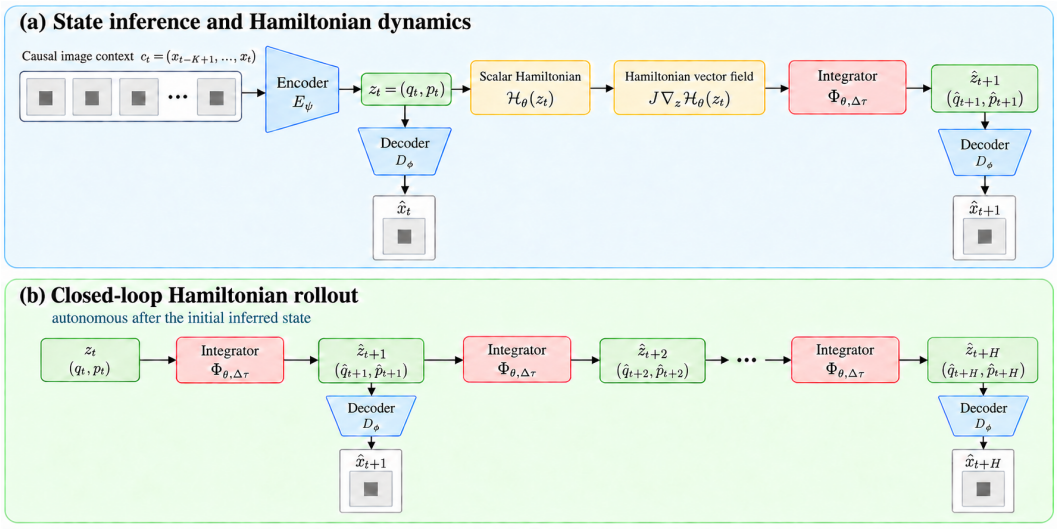



Figure 1: Overview of the ViHaD architecture. (a) A causal image context is encoded into a latent phase-space state, evolved under the learned Hamiltonian dynamics, and decoded into image space. (b) A closed-loop rollout recursively integrates predicted latent states and decodes them into images.

satisfy the proposed recovery criterion; their qualitative rollout examples in image space are likewise limited to mass–spring.

## 3 Method

We jointly learn a low-dimensional state representation and Hamiltonian dynamics directly from image sequences. The model consists of a causal state encoder, an observation decoder, and a scalar Hamiltonian neural network. Training combines reconstruction and local dynamics supervision with latent-state regularization and progressively extended closed-loop trajectory losses. An overview of the complete ViHaD architecture is shown in Figure 1.

### 3.1 Visual Hamiltonian formulation

Hamiltonian mechanics describes the evolution of a dynamical system through a scalar Hamiltonian defined over phase space (Goldstein et al., 2002). For a system with $d$ degrees of freedom, the canonical state is $z = (q, p) \in \mathbb{R}^{2d}$, where $q, p \in \mathbb{R}^d$ denote generalized coordinates and conjugate momenta. Its dynamics satisfy Hamilton's equations,

$$\dot{q} = \frac{\partial H}{\partial p}, \qquad \dot{p} = -\frac{\partial H}{\partial q}. \tag{1}$$

For the conservative mechanical systems considered in this work, the physical Hamiltonian corresponds to the total mechanical energy and remains constant along an exact trajectory.

In our setting, the physical state is not directly observed. As illustrated in Figure 1(a), let $x_t$ denote an image observation and $c_t$ a causal image context containing only observations available up to time $t$. The encoder $E_\psi$ maps this context to a $2d$-dimensional latent state,

$$z_t = E_\psi(c_t) = \begin{bmatrix} q_t \\ p_t \end{bmatrix} \in \mathbb{R}^{2d}, \tag{2}$$

where $q_t, p_t \in \mathbb{R}^d$ are learned configuration-like and momentum-like coordinates. These inferred coordinates are not assumed to match the physical simulator coordinates.

A decoder $D_\phi$ maps the inferred latent state to its corresponding visual target $x_t$,

$$\widehat{x}_t = D_\phi(z_t), \qquad \mathcal{L}_{\text{rec}} = \mathbb{E}_t\left[\|D_\phi(E_\psi(c_t)) - x_t\|_2^2\right]. \tag{3}$$

The reconstruction loss keeps the latent state informative about the observed image. Following Hamiltonian Neural Networks (Greydanus et al., 2019), we parameterize the latent Hamiltonian using a scalar neural network $\mathcal{H}_\theta : \mathbb{R}^{2d} \to \mathbb{R}$. Its gradient defines the latent vector field,

$$\dot{z} = f_\theta(z) = J\nabla_z \mathcal{H}_\theta(z), \qquad J = \begin{bmatrix} 0 & I_d \\ -I_d & 0 \end{bmatrix}. \tag{4}$$

Thus, rather than learning an unconstrained state transition, the model learns a scalar Hamiltonian whose gradient determines the instantaneous evolution of the latent state. The encoder, decoder, and Hamiltonian network are optimized jointly, without using ground-truth simulator coordinates as latent-state supervision.

### 3.2 Higher-order local dynamics supervision

PixelHNN uses the one-step forward difference $D_+ z_t = (z_{t+1} - z_t)/\Delta\tau$ as its local dynamics target. This measures the average rate of change over one observation interval, whereas the Hamiltonian vector field specifies the instantaneous derivative at $z_t$. To reduce the truncation error of this derivative approximation on smooth trajectories, we construct the centered five-point estimator from neighboring states independently inferred from their observed contexts:

$$D_5 z_t := \frac{z_{t-2} - 8z_{t-1} + 8z_{t+1} - z_{t+2}}{12\Delta\tau}, \tag{5}$$

where $\Delta\tau = 1$ in frame-normalized time. We match this target to the Hamiltonian vector field at the corresponding inferred state:

$$\mathcal{L}_{\text{dyn}} = \mathbb{E}_t \left[ \| J\nabla_z \mathcal{H}_\theta(z_t) - D_5 z_t \|_2^2 \right]. \tag{6}$$

$\mathcal{L}_{\text{dyn}}$ provides local supervision of the learned vector field using states inferred directly from observed image contexts; recursively predicted states are supervised separately through the rollout losses. Direct-state experiments in Appendix A.4 show that the five-point target closely approximates the analytic pendulum vector field and yields HNN rollouts closer to analytic-target training than forward differences, isolating the effect of derivative estimation from representation learning.

### 3.3 Latent-state regularization

Joint reconstruction and Hamiltonian learning can admit latent representations that reconstruct observations well but do not support accurate learned dynamics. For a training batch $\{z_i\}_{i=1}^N$, let $\bar{z} = \frac{1}{N}\sum_{i=1}^N z_i$ and $r_i = \|z_i - \bar{z}\|_2$. We discourage collapse of the encoded states toward a common point using

$$\mathcal{L}_{\text{rad}} = \frac{1}{N}\sum_{i=1}^{N} \max(-\log(r_i + \epsilon), 0). \tag{7}$$

To maintain variation along each latent coordinate, we define $s_j = \text{Std}_i(z_{ij})$ and use

$$\mathcal{L}_{\text{std}} = \sum_{j=1}^{2d} \omega_j \max(-\log(s_j + \epsilon), 0). \tag{8}$$

We additionally encourage consistency between the configuration-like and momentum-like components through kinetic tangent consistency,

$$\mathcal{L}_{\text{KTC}} = \mathbb{E}_t \left[ \| \alpha p_t - D_5 q_t \|_2^2 \right], \qquad \alpha > 0, \tag{9}$$

where $p_t$ is the momentum-like component of $z_t$, $D_5 q_t$ is the estimated tangent of its configuration-like component, and $\alpha$ is learned. Analogous to PixelHNN's canonical-coordinate consistency loss, this term couples the momentum-like latent coordinates to the configuration-like tangent.

### 3.4 CLOSED-LOOP ROLLOUT TRAINING

Local derivative supervision constrains the vector field only at states inferred from observed sequences. To account for error accumulation under repeated integration, we train on autonomous latent rollouts. Starting from an inferred state $z_t$,

$$\widehat{z}_{t+k} = \Phi^k_{\theta,\Delta\tau}(z_t), \qquad k = 1, \dots, H, \tag{10}$$

where $\Phi_{\theta,\Delta\tau}$ is a differentiable numerical integration step under $f_\theta$, and the superscript denotes repeated composition. The latent rollout loss is

$$\mathcal{L}_{\text{lat}} = \frac{1}{H} \sum_{k=1}^{H} \|\widehat{z}_{t+k} - z_{t+k}\|_2^2, \qquad z_{t+k} = E_\psi(c_{t+k}), \tag{11}$$

and the corresponding observation-space loss is

$$\mathcal{L}_{\text{pix}} = \frac{1}{H} \sum_{k=1}^{H} \|D_\phi(\widehat{z}_{t+k}) - x_{t+k}\|_2^2. \tag{12}$$

The target states $z_{t+k}$ are inferred independently from observed contexts, but do not reset the predicted trajectory; each $\widehat{z}_{t+k}$ is generated from the preceding predicted state.

Rather than training with the longest rollout horizon from the beginning, we progressively increase the closed-loop horizon, $H_1 < H_2 < \cdots < H_S$, initializing each stage from the preceding one while jointly training all model components.

### 3.5 TRAINING OBJECTIVE

The complete objective is

$$\begin{aligned}\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{dyn}}\mathcal{L}_{\text{dyn}} + \lambda_{\text{lat}}\mathcal{L}_{\text{lat}} + \lambda_{\text{pix}}\mathcal{L}_{\text{pix}} \\ + \lambda_{\text{KTC}}\mathcal{L}_{\text{KTC}} + \lambda_{\text{rad}}\mathcal{L}_{\text{rad}} + \mathcal{L}_{\text{std}}.\end{aligned} \tag{13}$$

We optimize the encoder, decoder, and Hamiltonian network jointly. System-specific loss coefficients, rollout schedules, and optimization settings are provided in Appendix A.2.

## 4 EXPERIMENT SETUP

Trajectories are generated from known physical dynamics and rendered as image sequences. The model receives only these images; simulator states and analytic Hamiltonians are not supplied as training targets. Dataset-generation, rendering, and preprocessing details are provided in Appendix A.3.

### 4.1 PENDULUM

We consider a frictionless point-mass pendulum with mass $m = 0.5$, length $l = 1$, and gravitational acceleration $g = 3$. Let $q$ denote the angle from the downward equilibrium and $p = ml^2\dot{q}$ the canonical momentum. Its physical Hamiltonian is

$$H_{\text{phys}}(q, p) = \frac{p^2}{2ml^2} + mgl(1 - \cos q) = p^2 + 1.5(1 - \cos q). \tag{14}$$

The separatrix energy is $E_{\text{sep}} = 2mgl = 3$. Energies are sampled uniformly over $[0, 6]$, yielding libration, rotation in both directions, and trajectories near the separatrix. In contrast, PixelHNN was originally evaluated only on small-angle pendulum motion up to $\pi/6$ (Greydanus et al., 2019). We generate $4{,}000$ trajectories with $198$ rendered observations each at $\Delta t = 0.05\,\text{s}$, using $3{,}200$ trajectories for training and $800$ for testing.

### 4.2 MASS–SPRING SYSTEM

We consider a one-dimensional harmonic oscillator with canonical state $(q, p)$ and Hamiltonian

$$H_{\text{phys}}(q, p) = q^2 + p^2. \tag{15}$$

Initial conditions are sampled as $r \sim \mathcal{U}(0.1, 1.0)$ and $\varphi \sim \mathcal{U}(0, 2\pi)$, with $q_0 = r\cos\varphi$ and $p_0 = r\sin\varphi$. We generate $50{,}000$ training trajectories with 30 frames and $10{,}000$ independently sampled test trajectories with 102 frames at $\Delta t = 0.1\,\text{s}$.

### 4.3 TWO-BODY SYSTEM

We consider two equal unit-mass bodies interacting gravitationally in two spatial dimensions with $G = 1$. Let $q_i, p_i \in \mathbb{R}^2$ denote the position and momentum of body $i$. The physical Hamiltonian is

$$H_{\text{phys}}(q, p) = \frac{1}{2}\left(\|p_1\|_2^2 + \|p_2\|_2^2\right) - \frac{1}{\|q_2 - q_1\|_2}. \tag{16}$$

Initial conditions have zero center of mass and total momentum. We independently sample $q_{1x}, q_{1y} \sim \mathcal{U}(0.5, 1.5)$, set $q_2 = -q_1$, and compute equal-and-opposite tangential momenta at the circular-orbit magnitude. Both momenta are jointly scaled by $1 + 0.05\xi$, where $\xi \sim \mathcal{N}(0, 1)$, producing mildly eccentric trajectories similar to those in the HNN two-body experiment (Greydanus et al., 2019). We generate $50{,}000$ training and $10{,}000$ independently sampled test trajectories, each with 345 observations at $\Delta t = 0.125\,\text{s}$.

### 4.4 EVALUATION PROTOCOL

All evaluations are causal: the encoder infers one initial state from the observed context, after which the model receives no additional images. The pendulum and two-body encoders use six observed frames, while the mass–spring encoder uses two. Latent states are integrated with leapfrog for the pendulum and with RK4 for mass–spring and two-body, in both training and evaluation. Pendulum results use all 800 test trajectories; mass–spring and two-body use their complete 10,000-trajectory test splits. We use the same randomly sampled valid start frame for all two-body methods on each test trajectory, with sampling seed 0. We tune PixelHNN's loss weights separately for each system.

For each predicted and ground-truth frame, we define foreground by normalized pixel intensity greater than 0.15 and compute $\text{IoU} = |M_{\text{pred}} \cap M_{\text{gt}}|/|M_{\text{pred}} \cup M_{\text{gt}}|$. Let the per-step IoU, $\text{IoU}_k$, denote the IoU at rollout step $k$, averaged over test trajectories (Figure 2). We report Mean IoU $= \frac{1}{100}\sum_{k=1}^{100} \text{IoU}_k$ and $\text{IoU@100} = \text{IoU}_{100}$; MSE and MSE@100 are defined analogously. The central pendulum pivot region and fixed mass–spring support are excluded from the IoU computation. We use rollout foreground IoU as the primary ranking metric because the foreground occupies only a small fraction of each frame. It directly measures overlap between predicted and ground-truth foreground regions, whereas full-frame MSE can remain low even for phase-shifted, displaced, or blurred predictions because most pixels belong to the static background. Pixel MSE is reported as a complementary image-level measure.

## 5 RESULTS

Table 1 separates performance at the one-step training horizon from the additional effect of progressively longer closed-loop training. ViHaD improves over PixelHNN at $H = 1$ on all three systems, while progressive rollout training provides substantially larger gains on the pendulum and two-body tasks. Figure 2 shows how per-step IoU changes over the 100-step autonomous rollout.

**Pendulum: one model across motion regimes.** The pendulum provides a particularly challenging representation setting: a single two-dimensional latent space must coherently organize trajectories spanning libration, both directions of rotation, and motion near the separatrix. The PixelHNN baseline achieves 0.0741 mean IoU, below the 0.085 mean IoU obtained by repeating the last observed frame for all 100 steps. At the matched $H = 1$ training horizon, ViHaD increases mean IoU to 0.2142, but its IoU@100 is 0.0705, showing that the improvement does not persist over the full

Table 1: Causal 100-step visual rollout performance. $H$ is the training rollout horizon; $H = 1$ compares our objective with PixelHNN at the same one-step training horizon. Progressive models increase the horizon in stages: $2 \to 16 \to 32$ (pendulum) and $16 \to 32 \to 64$ (two-body). The mass–spring ViHaD row reports seed 2, whose mean IoU is closest to the four-seed mean.

| System | Method | Mean IoU ↑ | IoU@100 ↑ | MSE ↓ | MSE@100 ↓ |
|---|---|---|---|---|---|
| Pendulum | PixelHNN | 0.0741 | 0.0424 | $8.43\times10^{-3}$ | $8.75\times10^{-3}$ |
| | ViHaD, $H = 1$ | 0.2142 | 0.0705 | $7.11\times10^{-3}$ | $8.81\times10^{-3}$ |
| | ViHaD, progressive | **0.9117** | **0.8102** | $\mathbf{2.67\times10^{-4}}$ | $\mathbf{8.57\times10^{-4}}$ |
| Mass–spring | PixelHNN | 0.9110 | 0.8618 | $3.41\times10^{-4}$ | $7.23\times10^{-4}$ |
| | ViHaD, $H = 1$ | **0.9824** | **0.9777** | $\mathbf{2.09\times10^{-5}}$ | $\mathbf{2.66\times10^{-5}}$ |
| Two-body | PixelHNN | 0.5734 | 0.2138 | $2.49\times10^{-3}$ | $7.84\times10^{-3}$ |
| | ViHaD, $H = 1$ | 0.7246 | 0.4173 | $1.03\times10^{-3}$ | $3.00\times10^{-3}$ |
| | ViHaD, progressive | **0.9438** | **0.9097** | $\mathbf{1.04\times10^{-4}}$ | $\mathbf{1.46\times10^{-4}}$ |

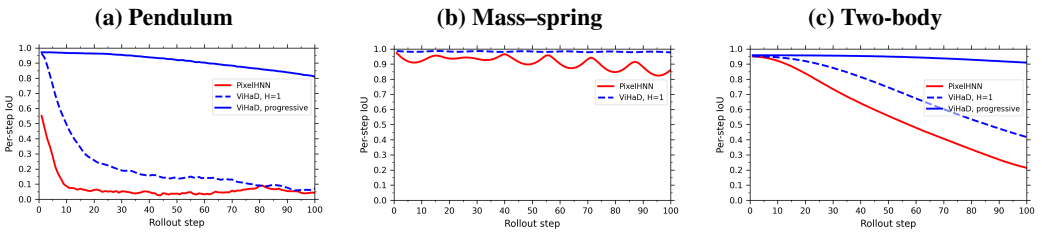


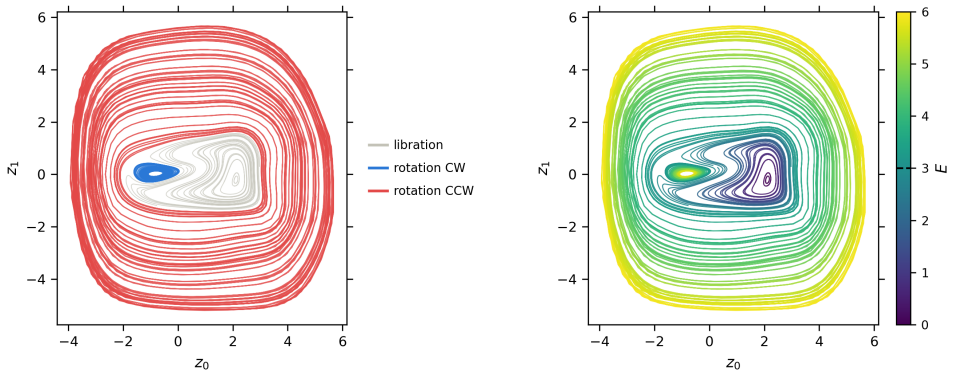

Figure 2: Per-step IoU ($\mathrm{IoU}_k$) over a 100-step autonomous rollout for (a) pendulum, (b) mass–spring, and (c) two-body, averaged over the complete test split. Each model observes only the initial context—six frames for pendulum and two-body, two for mass–spring—and receives no further images. Within each system, all models share the same encoder, decoder, and Hamiltonian-network architectures. PixelHNN (red) uses the PixelHNN objective; ViHaD with $H = 1$ (dashed blue) uses our objective at the same one-step training horizon; and progressive ViHaD (solid blue) additionally trains on progressively longer rollouts (schedules in Table 1).

rollout. Progressive training ($2 \to 16 \to 32$) increases mean IoU to 0.9117 and reaches an IoU@100 of 0.8102. Figure 4 shows that ViHaD's rollouts closely track the ground truth, whereas PixelHNN fails to do so.

Figure 3 provides a complementary view of the inferred state geometry. The physical pendulum has cylindrical phase space $T^*S^1 \cong S^1 \times \mathbb{R}$: libration lies below the separatrix, whereas clockwise and counter-clockwise rotations form two distinct branches above it. Representing these three trajectory families in a single two-dimensional latent space is nontrivial because the rotational branches visit the same configurations with opposite momentum. Without regime labels or physical-state supervision, ViHaD organizes the trajectories into coherent nested regions: clockwise rotations occupy an inner family, librations an intermediate family, and counter-clockwise rotations the outer family. Physical energy varies smoothly within each family, but its radial ordering is branch-dependent: energy increases outward for the outer rotational family and inward, toward smaller latent radius, for the inner rotational family. The learned representation is therefore asymmetric and not a direct copy of physical $(q, p)$, yet it cleanly separates the three motion families and supports accurate autonomous rollouts (Appendix A.5).

Training follows a formation–refinement pattern (Figure 8). Short-horizon training ($H = 2$) forms a usable latent organization (Appendix A.5), which longer horizons refine into the nested families of Figure 3. Formation is the fragile step, since training directly at $H = 16$ failed in all ten runs (Appendix A.2) and two of five progressive-recipe seeds did not form a usable organization at $H = 2$ and were not recovered at $H = 16$ (Figures 9 and 10).

**Mass–spring: accurate rollouts at a one-step training horizon.** PixelHNN is already strong on the mass–spring system, achieving 0.9110 mean IoU, although its IoU oscillates and declines




(a) Colored by motion regime. (b) Colored by physical energy.

Figure 3: Pendulum latent trajectories inferred from held-out image sequences. Each curve spans 120 frames, with each state inferred from six consecutive frames. The same trajectories are colored by (a) motion family and (b) trajectory-averaged physical energy; the dashed marker on the colorbar denotes the separatrix energy $E_{\text{sep}} = 3\,\text{J}$.

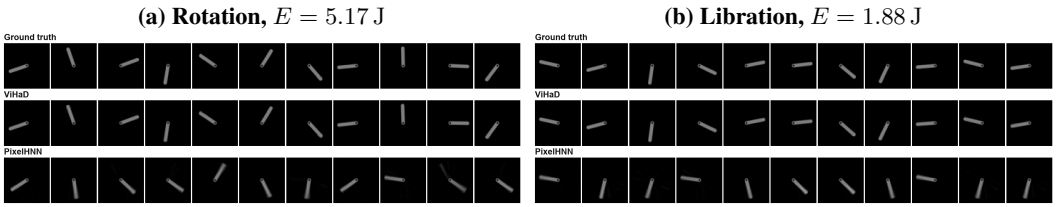



Figure 4: Autonomous 100-step visual rollouts on two held-out pendulum trajectories: (a) rotation above the separatrix ($E = 5.17\,\text{J}$) and (b) libration below it ($E = 1.88\,\text{J}$). Rows show ground truth, ViHaD (progressive), and PixelHNN; every tenth step from $t = 0$ to $t = 100$ is shown. Both models infer the initial state from the same six observed frames and are then integrated autonomously with leapfrog.

overall over the rollout. With a one-step training horizon ($H = 1$), ViHaD reaches 0.9824 mean IoU and an IoU@100 of 0.9777 while reducing pixel MSE by more than an order of magnitude. Its IoU curve remains nearly constant through step 100, demonstrating accurate autonomous prediction over this horizon without progressively extending the training horizon. A representative visual rollout is shown in Figure 5(a).

**Two-body: improved rollouts with longer-horizon training.** The PixelHNN baseline captures meaningful two-body motion, achieving 0.5734 mean IoU. At $H = 1$, ViHaD improves mean IoU to 0.7246 and IoU@100 from 0.2138 to 0.4173. With progressive closed-loop training ($16 \to 32 \to 64$), ViHaD reaches 0.9438 mean IoU and an IoU@100 of 0.9097. Figure 5(b) shows that this model tracks the ground-truth two-body motion more closely than PixelHNN.

## 6 Ablation Study

We perform the main component ablations on the mass–spring system, for which all objective variants can be evaluated under the same training and selection budget. Every model is trained for 17k updates with a one-step rollout horizon using the same four seeds and is evaluated on the complete 10,000-trajectory, 100-step test set. The reference configuration uses the centered five-point target and the independently selected coefficients $\lambda_{\text{dyn}} = 1.0$ and $\lambda_{\text{lat}} = 0.05$. The mass–spring ViHaD row in Table 1 reports seed 2, whose mean IoU is closest to the four-seed mean. Table 2 reports the mean and sample standard deviation over all four seeds trained with that same configuration.

The ablations reveal a clear hierarchy among the objective terms. Reconstruction is indispensable, while the per-coordinate non-collapse loss $\mathcal{L}_{\text{std}}$ is the key protection against latent collapse: remov-

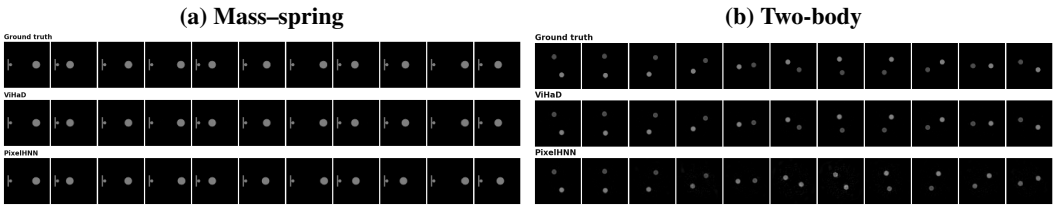



Figure 5: Autonomous 100-step visual rollouts on held-out (a) mass–spring and (b) two-body trajectories. Rows show ground truth, ViHaD ($H = 1$ for mass–spring; progressive for two-body), and PixelHNN, with every tenth step from $t = 0$ to $t = 100$. Initial states are inferred from the same two (mass–spring) or six (two-body) observed frames, and rollouts are integrated autonomously with RK4.

ing it lowers mean IoU from $0.9823$ to $0.3664$, nearly the same outcome as removing both non-collapse losses. The radial loss $\mathcal{L}_{\mathrm{rad}}$ alone has a much smaller effect. Local dynamics supervision $\mathcal{L}_{\mathrm{dyn}}$ improves both accuracy and reliability across seeds, whereas removing the one-step latent rollout loss $\mathcal{L}_{\mathrm{lat}}$ changes mean IoU by less than $0.0003$. That short-horizon result is system-specific; Table 1 shows that longer closed-loop training becomes decisive for the pendulum and two-body tasks. Finally, replacing the centered five-point target with a forward difference at the same coefficients reduces mean IoU to $0.8222$; Appendix A.4 isolates this numerical target comparison from representation learning using direct-state trajectories.

Table 2: Four-seed component ablations on mass–spring at training horizon $H = 1$. Every arm uses the same 17k-update budget and the same $10{,}000$-trajectory, 100-step evaluation set. The forward-target row retains the coefficients selected for the centered five-point reference.

| Configuration | Mean IoU | IoU@100 |
|---|---|---|
| Full model | $\mathbf{0.9823 \pm 0.0020}$ | $\mathbf{0.9769 \pm 0.0039}$ |
| No latent rollout loss ($\mathcal{L}_{\mathrm{lat}}$) | $0.9821 \pm 0.0026$ | $0.9768 \pm 0.0051$ |
| No radial loss ($\mathcal{L}_{\mathrm{rad}}$) | $0.9738 \pm 0.0016$ | $0.9570 \pm 0.0027$ |
| No dynamics loss ($\mathcal{L}_{\mathrm{dyn}}$) | $0.9097 \pm 0.0656$ | $0.8491 \pm 0.1164$ |
| Forward-difference target | $0.8222 \pm 0.0157$ | $0.6994 \pm 0.0299$ |
| No per-coordinate loss ($\mathcal{L}_{\mathrm{std}}$) | $0.3664 \pm 0.0388$ | $0.3504 \pm 0.0750$ |
| No non-collapse losses ($\mathcal{L}_{\mathrm{rad}}, \mathcal{L}_{\mathrm{std}}$) | $0.3735 \pm 0.0358$ | $0.3466 \pm 0.0707$ |
| No reconstruction loss ($\mathcal{L}_{\mathrm{rec}}$) | $0.0229 \pm 0.0063$ | $0.0230 \pm 0.0063$ |

## 7 Conclusions

We presented **ViHaD**, a visual Hamiltonian framework that jointly learns a low-dimensional state representation and Hamiltonian dynamics directly from image sequences. Across pendulum, mass–spring, and two-body systems, ViHaD produces more accurate long-horizon visual predictions than PixelHNN baselines. Our results indicate that reliable long-horizon prediction with visual Hamiltonian models depends not only on imposing Hamiltonian structure, but also on learning a well-behaved latent phase space and training through repeated closed-loop rollouts. We hope ViHaD serves as a foundation for learning more complex dynamics from images and, ultimately, from real-world video.

### AI use statement

In this work, we used generative AI tools to assist with drafting, revising, and polishing the manuscript, including improving wording, grammar, clarity, and organization. We also used these tools to help locate and verify related work and citations, and to generate the architecture diagram in Figure 1. We reviewed and edited the AI-assisted text, verified all citations against the original sources, and checked that the diagram accurately reflects the model. We take responsibility for the final content of this work, including all text, figures, and scientific claims produced with the assistance of generative AI.

## REFERENCES


Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, volume 29, pp. 4502–4510, 2016.

Filipe de Avila Belbute-Peres, Kevin A. Smith, Kelsey R. Allen, Joshua B. Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems*, volume 31, pp. 7178–7189, 2018.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, pp. 6571–6583, 2018.

Zhengdao Chen, Jianyu Zhang, Martín Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. In *International Conference on Learning Representations*, 2020.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Shaan A. Desai, Marios Mattheakis, David Sondak, Pavlos Protopapas, and Stephen J. Roberts. Port-Hamiltonian neural networks for learning explicit time-dependent dynamical systems. *Physical Review E*, 104(3):034312, 2021. doi: 10.1103/PhysRevE.104.034312.

Herbert Goldstein, Charles P. Poole, and John L. Safko. *Classical Mechanics*. Addison-Wesley, San Francisco, 3rd edition, 2002.

Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pp. 15379–15389, 2019.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, pp. 2450–2462. Curran Associates, Inc., 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019.

Irina Higgins, Peter Wirnsberger, Andrew Jaegle, and Aleksandar Botev. SyMetric: Measuring the quality of learnt Hamiltonian dynamics inferred from vision. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Pengzhan Jin, Zhen Zhang, Aiqing Zhu, Yifa Tang, and George Em Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132:166–179, 2020. doi: 10.1016/j.neunet.2020.08.017.

Cheng Jing, Uvini Balasuriya Mudiyanselage, Woojin Cho, Minju Jo, Anthony Gruber, and Kookjin Lee. Meta-learning structure-preserving dynamics. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational Bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2017.

Eli Laird and Corey Clark. Unlocking temporal generalization in Hamiltonian video dynamics models. *arXiv preprint arXiv:2607.07763*, 2026.

Seungjun Lee, Haesang Yang, and Woojae Seong. Identifying physical law of Hamiltonian systems via meta-learning. In *International Conference on Learning Representations*, 2021.

Michael Lutter, Christian Ritter, and Jan Peters. Deep Lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Yulia Rubanova, Ricky T. Q. Chen, and David K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, volume 32, pp. 5320–5330, 2019.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ODE integrators. *arXiv preprint arXiv:1909.12790*, 2019.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8459–8468. PMLR, 2020.

Andrew Sosanya and Sam Greydanus. Dissipative Hamiltonian neural networks: Learning dissipative and conservative dynamics separately. *arXiv preprint arXiv:2201.10085*, 2022.

Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *International Conference on Learning Representations*, 2020.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, volume 28, pp. 2746–2754, 2015.

Nicholas Watters, Daniel Zoran, Théophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4539–4547, 2017.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ODE-Net: Learning Hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020.

# A Appendix

## A.1 Model architectures

Table 3 summarizes the architectures used for the reported models and their matched PixelHNN baselines. Within each physical system, the proposed model and baseline use the same encoder, decoder, latent dimension, and Hamiltonian network. Their training objectives, optimization settings, and horizon exposure are reported in Appendix A.2. All linear weights are initialized orthogonally. The autoencoders use ReLU activations, while the Hamiltonian networks use $\tanh$ activations and produce a scalar Hamiltonian.

Table 3: Architectures used for the reported results. Widths list hidden-layer dimensions; parameter counts include the encoder, decoder, and Hamiltonian network.

| System | Image context | Latent split | AE width(s) | HNN width | Parameters |
|---|---|---|---|---|---|
| Pendulum | $6 \times 64 \times 64$ | $1q + 1p$ | $512, 1024$ | 400 | 9,086,050 |
| Mass–spring | $2 \times 64 \times 64$ | $1q + 1p$ | 200 | 200 | 2,664,698 |
| Two-body | $6 \times 64 \times 64$ | $4q + 4p$ | $512, 1024$ | 400 | 9,094,200 |

**Pendulum.** Each of the six one-channel frames is flattened and passed through a shared per-frame stem,

$$4096 \to 512 \to 512, \tag{17}$$

where the second transformation is an additive residual layer. The six embeddings are concatenated in temporal order and mapped by

$$3072 \to 1024 \to 512 \to 512 \to 2, \tag{18}$$

with an additive residual transformation at width $512$. The selected model uses a one-frame residual MLP decoder

$$2 \to 512 \to 512 \to 512 \to 4096, \tag{19}$$

with a linear output layer. The Hamiltonian branch has two width-$400$ $\tanh$ hidden layers. For compatibility with the original HNN implementation, its final linear layer emits two values; the solenoidal configuration uses only the Hamiltonian channel to construct $J\nabla_z \mathcal{H}_\theta(z)$.

**Mass–spring.** Two adjacent one-channel frames are flattened and concatenated into an $8192$-dimensional input. The residual MLP encoder and decoder are

$$8192 \to 200 \to 200 \to 200 \to 2, \qquad 2 \to 200 \to 200 \to 200 \to 4096. \tag{20}$$

The first hidden transformation uses ReLU and each of the next two is an additive residual transformation. The decoder reconstructs only the newest frame in the two-frame encoder context, which is the visual output associated with the inferred state. The scalar Hamiltonian MLP is $2 \to 200 \to 200 \to 1$, with two $\tanh$ hidden layers and a linear output without bias.

**Two-body.** The two-body model applies a shared residual MLP stem to each of the six one-channel context frames,

$$4096 \to 512 \to 512. \tag{21}$$

The six frame embeddings are concatenated in temporal order and mapped by

$$3072 \to 1024 \to 512 \to 512 \to 8. \tag{22}$$

The decoder reconstructs the single frame associated with the inferred state using

$$8 \to 512 \to 512 \to 512 \to 4096. \tag{23}$$

The latent state is partitioned into four configuration-like and four momentum-like coordinates. The scalar Hamiltonian MLP is $8 \to 400 \to 400 \to 1$, with two $\tanh$ hidden layers and a bias-free linear output.

In all systems, the encoder, decoder, and Hamiltonian network are optimized jointly and remain trainable throughout rollout-horizon continuation.

### A.2 Training configurations

We list the configurations reported in Table 1, together with a direct long-horizon comparison for the pendulum. ViHaD uses the training objective in equation 13. The PixelHNN baseline uses

$$\mathcal{L}_{\mathrm{PixelHNN}} = \lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{CC}}\mathcal{L}_{\mathrm{CC}} + \lambda_{\mathrm{dyn}}\mathcal{L}_{\mathrm{dyn}},$$

where $\mathcal{L}_{\mathrm{rec}}$ is the image reconstruction loss, $\mathcal{L}_{\mathrm{CC}}$ is PixelHNN's auxiliary canonical-coordinate loss, and $\mathcal{L}_{\mathrm{dyn}}$ uses the one-step forward-difference target.

N/A denotes a loss that is not used. All reported models use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. Unless noted otherwise, weight decay is $2.1241 \times 10^{-7}$ for encoder–decoder parameters and $3.7286 \times 10^{-9}$ for Hamiltonian parameters. The two-body PixelHNN baseline instead uses weight decay $0.1$ for both parameter blocks. Batch size is 512 for pendulum and two-body and 64 for mass–spring.

Table 4: Pendulum $H = 1$ training configurations.

| Model/stage | $H$ | Updates | Learning rate | $\lambda_{\mathrm{rec}}$ | $\lambda_{\mathrm{CC}}$ | $\lambda_{\mathrm{dyn}}$ | $\lambda_{\mathrm{lat}}$ | $\lambda_{\mathrm{pix}}$ | $\lambda_{\mathrm{KTC}}$ | $\lambda_{\mathrm{rad}}$ | $w_{\mathrm{std,q}}$ | $w_{\mathrm{std,p}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PixelHNN | 1 | 50k | $5.4928\times10^{-4}$ | 0.679498 | 0.416295 | 0.1 | N/A | N/A | N/A | N/A | N/A | N/A |
| ViHaD, $H = 1$ | 1 | 50k | $5.4928\times10^{-4}$ | 6.79498 | N/A | 0.1 | 0.1 | 1 | 0.0416295 | 0.00113205 | 0.0004 | 0.015 |

Table 5: Mass–spring $H = 1$ training configurations.

| Model/stage | $H$ | Updates | Learning rate | $\lambda_{\mathrm{rec}}$ | $\lambda_{\mathrm{CC}}$ | $\lambda_{\mathrm{dyn}}$ | $\lambda_{\mathrm{lat}}$ | $\lambda_{\mathrm{pix}}$ | $\lambda_{\mathrm{KTC}}$ | $\lambda_{\mathrm{rad}}$ | $w_{\mathrm{std,q}}$ | $w_{\mathrm{std,p}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PixelHNN | 1 | 17k | $7.4639\times10^{-5}$ | 1 | 0.1 | 0.1 | N/A | N/A | N/A | N/A | N/A | N/A |
| ViHaD, $H = 1$ | 1 | 17k | $7.4639\times10^{-5}$ | 1.698745 | N/A | 1 | 0.05 | 0 | 0 | 0.00113205 | 0.0004 | 0.015 |

Table 6: Two-body $H = 1$ training configurations.

| Model/stage | $H$ | Updates | Learning rate | $\lambda_{\mathrm{rec}}$ | $\lambda_{\mathrm{CC}}$ | $\lambda_{\mathrm{dyn}}$ | $\lambda_{\mathrm{lat}}$ | $\lambda_{\mathrm{pix}}$ | $\lambda_{\mathrm{KTC}}$ | $\lambda_{\mathrm{rad}}$ | $w_{\mathrm{std,q}}$ | $w_{\mathrm{std,p}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PixelHNN | 1 | 100k | $1\times10^{-4}$ | 2 | 0 | 1 | N/A | N/A | N/A | N/A | N/A | N/A |
| ViHaD, $H = 1$ | 1 | 100k | $1\times10^{-4}$ | 10 | N/A | 1 | 10 | 1 | 0 | 0.00113205 | 0.0004 | 0.015 |

Table 7: Training stages for two-body ViHaD, progressive ($H$: $16\to32\to64$).

| Stage | $H$ | Updates | Learning rate | $\lambda_{\mathrm{rec}}$ | $\lambda_{\mathrm{CC}}$ | $\lambda_{\mathrm{dyn}}$ | $\lambda_{\mathrm{lat}}$ | $\lambda_{\mathrm{pix}}$ | $\lambda_{\mathrm{KTC}}$ | $\lambda_{\mathrm{rad}}$ | $w_{\mathrm{std,q}}$ | $w_{\mathrm{std,p}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Stage 1 | 16 | 85k | $1\times10^{-4}$ | 0.5 | N/A | 0.1 | 0.03 | 1 | 0 | 0.00113205 | 0.0004 | 0.015 |
| Stage 2 | 32 | 90k | $1\times10^{-4}$ | 0.5 | N/A | 0.1 | 0.03 | 1 | 0 | 0.00113205 | 0.0004 | 0.015 |
| Stage 3 | 64 | 125k | $1\times10^{-4}$ | 0.5 | N/A | 0.1 | 0.03 | 1 | 0 | 0.00113205 | 0.0004 | 0.015 |
| Stage 4 | 64 | 32.5k | $1\times10^{-5}$ | 0.5 | N/A | 0.1 | 0.03 | 1 | 0 | 0.00113205 | 0.0004 | 0.015 |
| Stage 5 | 64 | 40k | $5\times10^{-6}$ | 0.5 | N/A | 0.1 | 0.03 | 1 | 0 | 0.00113205 | 0.0004 | 0.015 |

Table 8: Training stages for pendulum ViHaD, progressive ($H$: $2\to16\to32$).

| Stage | $H$ | Updates | Learning rate | $\lambda_{\mathrm{rec}}$ | $\lambda_{\mathrm{CC}}$ | $\lambda_{\mathrm{dyn}}$ | $\lambda_{\mathrm{lat}}$ | $\lambda_{\mathrm{pix}}$ | $\lambda_{\mathrm{KTC}}$ | $\lambda_{\mathrm{rad}}$ | $w_{\mathrm{std,q}}$ | $w_{\mathrm{std,p}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Stage 1 | 2 | 30k | $5.4928\times10^{-4}$ | 6.79498 | N/A | 0.1 | 0.1 | 1 | 0.0416295 | 0.00113205 | 0.0004 | 0.015 |
| Stage 2 | 16 | 195k | $1\times10^{-4}$ | 3.39749 | N/A | 0.05 | 3.2 | 1 | 0.0416295 | 0.00113205 | 0.0004 | 0.015 |
| Stage 3 | 32 | 435k | $3.5\times10^{-5}$ | 3.39749 | N/A | 0.05 | 3.2 | 1 | 0.0416295 | 0.00113205 | 0.0004 | 0.015 |

Each row groups all continuation segments run with the listed horizon and loss weights. "Updates" counts updates from that phase's initialization to the reported phase endpoint, rather than cumulative updates across the full lineage.

In the final pendulum stage, clockwise rotations are oversampled: minibatches target clockwise, counter-clockwise, and librational shares of $0.35$, $0.15$, and $0.50$, respectively. Earlier stages use natural sampling.

**Direct long-horizon training.** To test whether the short-horizon stage is necessary, we trained ten pendulum models directly at $H = 16$ from scratch for 50k updates, using the first-stage recipe of progressive ViHaD with only the horizon changed. All ten fail: mean IoU reaches a median of 0.065 and at best 0.170, and at every checkpoint IoU on rotating trajectories stays at the static-prediction level ($\leq 0.047$), with the runs that improve doing so only on librations. Although the progressive model receives more updates in total, direct $H = 16$ after 50k updates does not reach even the $H = 2$ stage from which progressive training starts (mean IoU 0.204 after 30k updates), so the failure is not explained by training budget.

### A.3 Dataset Generation, Rendering, and Preprocessing

**Common preprocessing.** All models receive $64 \times 64$ single-channel observations in $[0, 1]$. The input channel is the clipped difference between the red and green image channels, and context frames are supplied to the encoder in chronological order. For the reported mass–spring and two-body runs, observations are rendered on demand from the clean simulated trajectories using deterministic signed-distance rasterization.

**Pendulum.** For each trajectory, we sample an energy $E \sim \mathcal{U}(0, 6)$ and choose an initial angle uniformly from the range feasible at that energy. The remaining energy determines the magnitude of the initial angular velocity, whose sign is sampled uniformly, producing both rotational directions in addition to librational and near-separatrix motion. The physical angle is stored unwrapped during simulation and wrapped to $[-\pi, \pi)$ only for rendering.

The physical equations are integrated with adaptive RK45 using $\mathrm{rtol} = 10^{-11}$ and $\mathrm{atol} = 10^{-13}$. Observations are sampled every $0.05\,\mathrm{s}$.

Pendulum frames are rendered using Gym Pendulum-v1 (Brockman et al., 2016), with

$$\theta_{\mathrm{Gym}} = \pi + \mathrm{wrap}_{\pi}(q), \tag{24}$$

so that $q = 0$ corresponds to the visually downward equilibrium. Gym is used only for rendering; its internal dynamics are not stepped. Each frame is rendered on a $2000 \times 2000$ Pygame canvas, cropped to the square region $[400{:}1600, 400{:}1600]$, resized to $64 \times 64$ by bilinear interpolation, and converted using the common red-minus-green preprocessing. Each simulated trajectory contains $200$ time points, of which the $198$ interior rendered frames are retained for training and evaluation.

**Mass–spring.** The oscillator satisfies $\dot{q} = 2p$ and $\dot{p} = -2q$. Rather than numerically integrating these equations, trajectories are evaluated analytically as

$$q(t) = q_0 \cos(2t) + p_0 \sin(2t), \qquad p(t) = -q_0 \sin(2t) + p_0 \cos(2t). \tag{25}$$

Initial conditions are generated by sampling $r \sim \mathcal{U}(0.1, 1.0)$ and $\varphi \sim \mathcal{U}(0, 2\pi)$, followed by $q_0 = r\cos\varphi$ and $p_0 = r\sin\varphi$.

Observations are rendered directly at $64 \times 64$. On a normalized image plane, the moving mass is represented by a disk of radius $0.085$ centered at $(0.62 + 0.22q,\ 0.50)$. A fixed support disk of radius $0.035$ is centered at $(0.14, 0.50)$, with a short vertical wall immediately to its left. Disk and wall coverage use a signed-distance antialiasing width of $0.75$ pixel. The foreground color is $(0.8, 77/255, 77/255)$ on a white background.

**Two-body.** For two unit-mass bodies with $G = 1$, the equations of motion are

$$\dot{q}_i = p_i, \qquad \dot{p}_i = \frac{q_j - q_i}{\|q_j - q_i\|_2^3}, \qquad j \neq i. \tag{26}$$

Initial conditions are chosen to give zero center of mass and zero total momentum. We sample $q_{1x}, q_{1y} \sim \mathcal{U}(0.5, 1.5)$ and set $q_2 = -q_1$. The initial momenta are

$$p_1 = s\,\frac{(-q_{1y}, q_{1x})}{2\|q_1\|_2^{3/2}}, \qquad p_2 = -p_1, \tag{27}$$

where $s = 1 + 0.05\xi$ with $\xi \sim \mathcal{N}(0, 1)$. The vector $(-q_{1y}, q_{1x})$ is $q_1$ rotated by $90^\circ$, so both bodies start moving counter-clockwise, perpendicular to the line joining them. For $s = 1$, the momenta have exactly the magnitude required for a circular orbit; $s \neq 1$ makes the orbits mildly eccentric.

Trajectories are integrated using fourth-order Runge–Kutta with ten substeps between consecutive observations. The physical state is ordered as $(q_{1x}, q_{1y}, p_{1x}, p_{1y}, q_{2x}, q_{2y}, p_{2x}, p_{2y})$.

Two-body observations are rendered directly at $64 \times 64$. Each physical position $q_i \in \mathbb{R}^2$ is mapped to image coordinates by $c_i = 0.5 + 0.165 q_i$. Each body is represented by an antialiased disk of radius $0.055$ with a $0.75$-pixel soft edge. The bodies are composited in order over a white background using colors $(0.8, 77/255, 77/255)$ and $(0.8, 0.5, 77/255)$.

### A.4 Direct-state validation of derivative targets

We evaluate the numerical derivative targets directly on the held-out split of the physical pendulum dataset described in Section 4.1. This comparison uses the true state $z_t = (q_t, p_t)$ and therefore contains neither an image encoder nor representation-learning error. The analytic vector field is

$$\dot{q}_t = \frac{p_t}{ml^2}, \qquad \dot{p}_t = -mgl \sin q_t. \tag{28}$$

At the dataset interval $\Delta t = 0.05\,\mathrm{s}$, we compare this vector field with the one-step forward difference and centered five-point estimate computed from the same trajectories. The evaluation contains all 800 held-out trajectories and 154,400 aligned interior states spanning the full 0–6 J energy range.

For target $D \in \{D_+, D_5\}$ and state component $c \in \{q, p\}$, we report

$$\mathrm{MSE}_c(D) = \frac{1}{M}\sum_{j=1}^{M}\left(Dz_{j,c} - f_c(z_j)\right)^2, \qquad M = 154{,}400. \tag{29}$$

Table 9: Direct-state derivative-target error against the analytic pendulum vector field over the complete test split.

| Target | $\dot{q}$ MSE | $\dot{p}$ MSE |
|---|---|---|
| Analytic derivative | 0 | 0 |
| Forward difference | $2.839\times10^{-3}$ | $4.888\times10^{-3}$ |
| Centered five-point | $3.394\times10^{-10}$ | $2.785\times10^{-9}$ |

We also test whether the target difference propagates to learned dynamics. We train three direct-state HNNs for a matched 50k-update budget using identical initialization and minibatches. Each model is trained only by matching its Hamiltonian vector field to the corresponding analytic, forward-difference, or centered-five-point target; no rollout loss is used. Starting from the true initial state of every test trajectory, we then integrate each learned vector field autonomously for 100 steps with the same leapfrog integrator used in the image-based pendulum experiments.

Because $q$ is an angle, its rollout error is the principal-angle difference

$$e^q_{i,k} = \mathrm{atan2}(\sin(\hat{q}_{i,k} - q_{i,k}), \cos(\hat{q}_{i,k} - q_{i,k}))\,, \tag{30}$$

whereas momentum uses $e^p_{i,k} = \hat{p}_{i,k} - p_{i,k}$. The mean rollout metrics are

$$\mathrm{MSE}_q = \frac{1}{NT}\sum_{i=1}^{N}\sum_{k=1}^{T}(e^q_{i,k})^2, \qquad \mathrm{MSE}_p = \frac{1}{NT}\sum_{i=1}^{N}\sum_{k=1}^{T}(e^p_{i,k})^2, \tag{31}$$

with $N = 800$ and $T = 100$; the @100 metrics average only $k = 100$. Angle wrapping is used only for evaluation: all three HNNs are trained and integrated in the raw unwrapped $(q, p)$ coordinates.

Table 10: Direct-state autonomous rollout error over the complete physical pendulum test split. All models are evaluated after the same 50k-update budget and are trained without rollout supervision.

| Training target | Mean wrapped $q$ MSE | Mean $p$ MSE | Wrapped $q$ MSE@100 | $p$ MSE@100 |
|---|---|---|---|---|
| Analytic derivative | $6.997\times10^{-3}$ | $4.110\times10^{-3}$ | $2.254\times10^{-2}$ | $8.811\times10^{-3}$ |
| Centered five-point | $8.458\times10^{-3}$ | $4.482\times10^{-3}$ | $3.333\times10^{-2}$ | $1.486\times10^{-2}$ |
| Forward difference | $5.202\times10^{-2}$ | $1.550\times10^{-2}$ | $2.906\times10^{-1}$ | $8.445\times10^{-2}$ |

The centered five-point target is consequently a close numerical approximation to the analytic instantaneous derivative at the observation rate used by the image experiments. Direct-state HNN training with this target also remains substantially closer to analytic-target training than the forward-difference alternative under matched local-only supervision. Because these comparisons are performed directly on physical states, they isolate derivative discretization from the additional optimization and representation errors present in the joint pixel model.

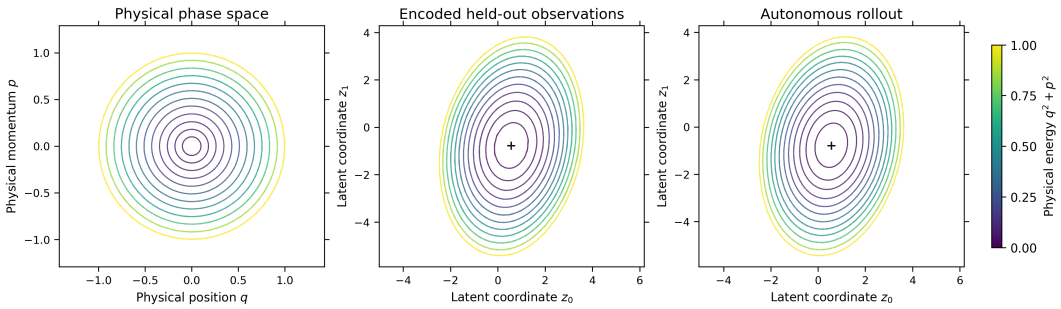



Figure 6: Mass–spring phase-space geometry. Left: twelve held-out trajectories in physical phase space $(q, p)$. Middle: the same trajectories encoded into the two-dimensional latent space. Right: 100-step autonomous rollouts from each trajectory's initial encoded state, with no further observations. Curves are colored by physical energy $q^2+p^2$; the cross marks the latent image of the physical equilibrium. Encoded and autonomous orbits closely coincide and preserve the energy ordering of the physical system.

### A.5 Phase-space analysis

Pixel metrics measure visual agreement but not whether the learned latent dynamics reproduce the physical phase-space structure. We therefore inspect latent trajectories directly: in the latent space for the mass–spring system, and in physical coordinates via a readout for the pendulum.

**Mass–spring.** Figure 6 shows that the encoder maps the circular physical orbits to nested, energy-ordered, approximately elliptical closed orbits in the latent space, and that autonomous rollouts remain on the encoded orbits. This analysis uses the same seed-2 ViHaD checkpoint reported in Table 1.

**Pendulum.** ViHaD's latent coordinates form a learned, nonlinear chart of phase space and cannot be compared with $(q, p)$ directly. To visualize them in physical coordinates, we fit a readout MLP $B : z \mapsto (\hat{q}, \hat{p})$ after training, using encoded training-split trajectories; it is used only for evaluation and never during training. $B$ predicts $(\cos q, \sin q, p)$ to respect the periodicity of $q$, and a single readout is shared across all motion families. Encoded observations and rolled-out latent states pass through the same fixed readout, so differences between them in Figure 7 reflect the learned dynamics rather than the readout. We roll out for 192 steps because the pendulum's period becomes arbitrarily long as trajectories approach the separatrix, so a 100-step rollout would not complete a full period for all trajectories shown; 192 steps covers at least one full period of each trajectory shown here.

**Latent-chart formation.** Figure 8 shows the latent chart of progressive ViHaD after each training stage. Under a time-independent Hamiltonian, exactly one trajectory passes through each point of phase space, so distinct trajectories cannot cross. At $H = 2$, the clockwise and counter-clockwise families are already disjoint, and longer horizons refine this organization. Of five seeds trained with the same recipe, three formed such a chart at $H = 2$. Figures 9 and 10 show the other two, in which the two rotation directions cross at $H = 2$; subsequent training at $H = 16$ does not recover a usable chart.

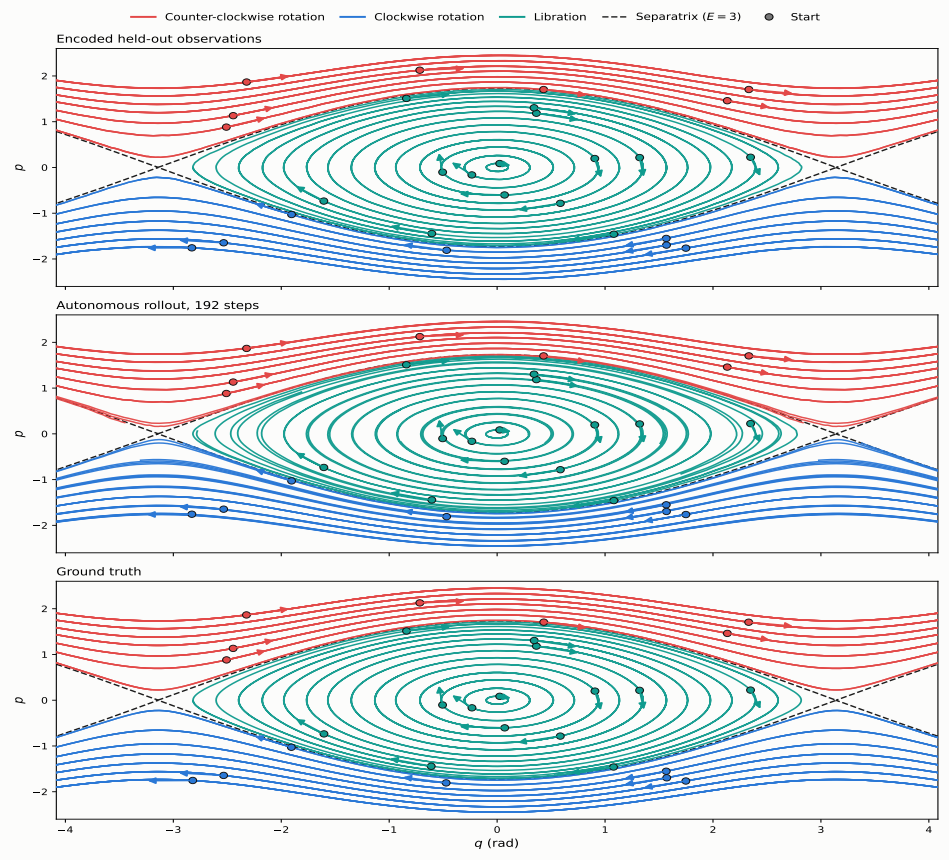



Figure 7: Pendulum phase portraits for 28 held-out test trajectories over 192 steps. Top: encoded held-out observations mapped to $(q, p)$ by the readout $B$. Middle: autonomous rollouts from the first encoded state only, mapped through the same readout. Bottom: ground truth. Dots mark each trajectory's start and arrows the direction of motion; $q$ is drawn beyond $\pm\pi$ so rotations are not cut at the boundary. No rollout leaves its motion family. Over repeated laps, clockwise rotations and orbits closest to the separatrix drift slightly across energy levels, appearing as broadened bands, while the encoded and ground-truth curves remain single lines.

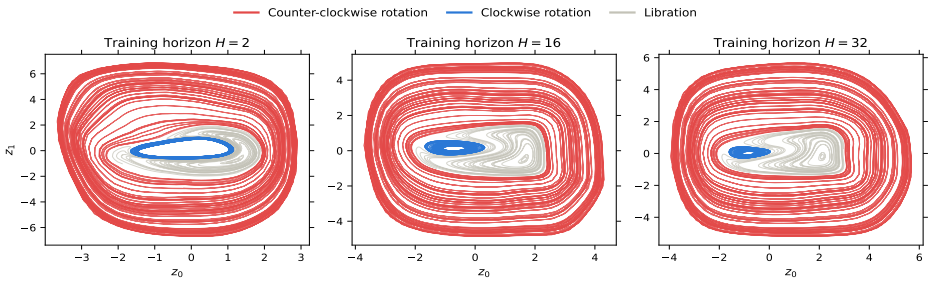



Figure 8: Latent charts of progressive ViHaD after the $H = 2$, $H = 16$, and final $H = 32$ stages (mean IoU 0.204, 0.775, and 0.912). Each curve is an encoded held-out trajectory over 120 frames, colored by its ground-truth motion family (librations in grey, as in Figure 3); no readout is involved. At $H = 2$, the clockwise and counter-clockwise families already occupy disjoint regions; $H = 16$ and $H = 32$ refine this organization.

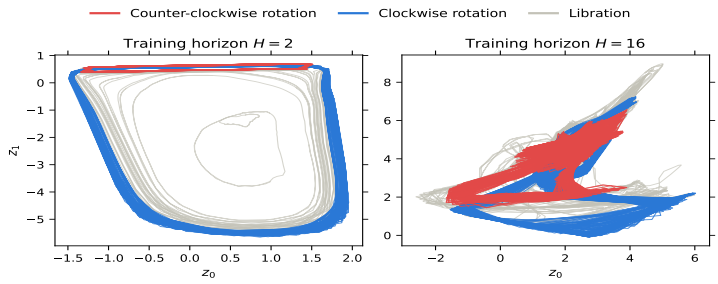



Figure 9: Latent charts for a seed that fails under the same recipe as progressive ViHaD (seed 3), after $H = 2$ and after 30k updates at $H = 16$; trajectories and colors as in Figure 8. At $H = 2$, the counter-clockwise and clockwise trajectories cross. At $H = 16$, the motion families fold into overlapping regions (mean IoU 0.016).

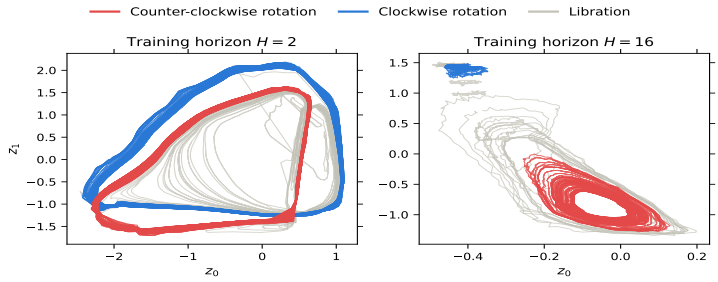



Figure 10: Latent charts for a second seed that fails under the same recipe (seed 5), after $H = 2$ and after 30k updates at $H = 16$. At $H = 2$, the counter-clockwise ring crosses both the librations and the clockwise ring. At $H = 16$, the clockwise rotations form a small tangled cluster (mean IoU 0.117).