Sparse Koopman Autoencoders Identify Local Dynamical Regimes in Multibasin Systems

Aidan Li[1,2,3] **Uday Kiran Reddy Tadipatri**[4] **Mahan Fathi**[5]
Sarath Chandar[1,2,6,7] **Ross Goroshin**[2,3]
[1]Chandar Research Lab [2]Mila – Quebec AI Institute [3]Université de Montréal
[4]University of Pennsylvania [5]NVIDIA [6]Polytechnique Montréal [7]Canada CIFAR AI Chair
aidan.li@mila.quebec
github.com/chandar-lab/skae

Abstract

Koopman autoencoders (KAEs) seek a higher-dimensional latent representation in which nonlinear dynamics evolve linearly. However, many interesting systems have multiple basins of attraction, and both theoretical and empirical work has shown these multibasin systems cannot generally admit a single finite-dimensional global Koopman embedding under standard assumptions. We posit that encoders with a sparsity-inducing objective encouraging few active latent coefficients will provide latent supports as an inspectable basin-modeling principle for Koopman autoencoders. We use these encoders producing sparse latents in training Sparse Koopman Autoencoders (SKAEs) without basin labels or other regime annotations, and treat the learned latent supports as model-produced regime variables after training. Across a range of procedurally generated multibasin systems and chaotic flows, we show that SKAEs have superior forecasting performance compared to dense-latent KAEs. We also perform a mechanistic study that shows latent supports produced by SKAEs are both essential for the quality of the representation and useful for identifying basins on basin-interior states, whereas dense-latent KAEs collapse to an uninformative single support family. These results identify sparse latents and their corresponding supports as label-free, interpretable regime variables for Koopman learning in nonlinear systems with multiple local dynamical laws.

1 Introduction

Learning representations for nonlinear dynamical systems remains a central problem in machine learning, scientific computing, and control. A particularly useful goal is to embed nonlinear dynamics into coordinates in which the evolution is linear, or at least locally linear, because linear models can be combined with mature tools for prediction, estimation, and control, including model predictive control and linear quadratic regulation [28, 15, 20, 17, 27].

This goal is also well motivated theoretically. Classical dynamical-systems results provide local topological linearizations near hyperbolic fixed points [14, 16], while Koopman eigenfunction constructions can yield linearizing coordinates on basins of attraction for stable equilibria and periodic orbits [23, 22]. Thus, nonlinear systems may admit linear dynamics with a specific change of coordinates.

Although these results show these useful coordinates exist, they do not provide a finite-data procedure for constructing such coordinates. Koopman operator theory formalizes the linearization idea by lifting nonlinear state evolution to a linear operator acting on observables [18, 19, 29, 6]. This viewpoint has inspired data-driven approximations of Koopman dynamics, such as DMD and EDMD [33, 37, 38], and more recently, Koopman autoencoders, which learn an encoder from state space to a

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

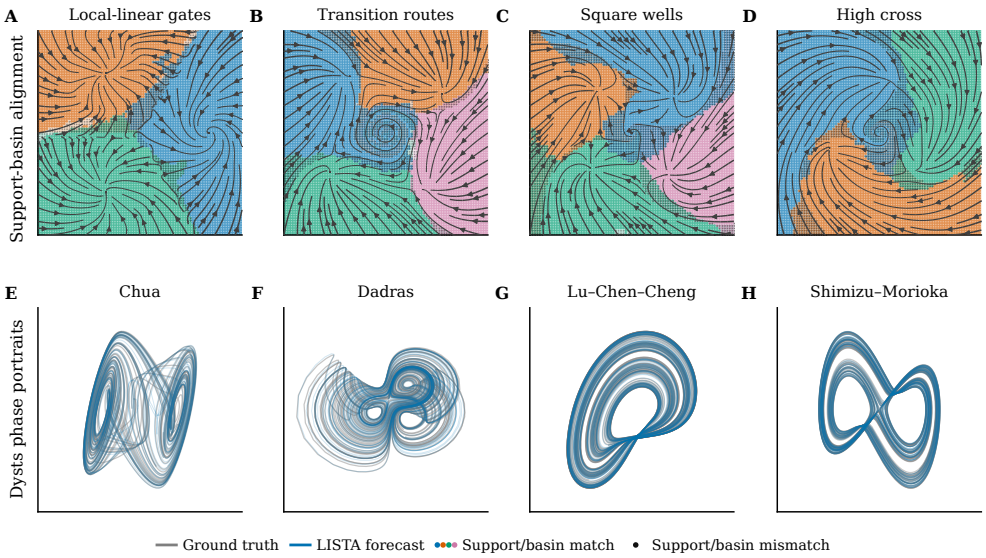


Figure 1: Top: procedurally generated multibasin systems where gray streamlines are the true vector field, colored dots mark support-family/basin matches, and black dots mark support-family/basin mismatches. The learned LISTA (seed 0) support families are mapped post hoc to each family's dominant validation basin, which colors each dot. Bottom: Dysts phase portraits of 20 test starts forecast for 5,000 observation steps, with held-out truth in gray and seed-0 LISTA forecasts in blue.

latent representation, a linear latent transition matrix, and a decoder back to state space [35, 26, 31, 3, 34, 10]. These models provide an appealing route from existence results to practical learning: rather than hand-designing observables, they attempt to learn the embedding directly from trajectory data.

A central challenge for Koopman approximations is that many practically interesting nonlinear systems typically contain multiple equilibria or basins of attraction. For example, a damped mechanical system can settle into different wells, a biological or chemical model can approach different stable equilibria, and a controlled system can move between regions with distinct local dynamics. However, theory shows that these systems cannot generally admit a single finite-dimensional global linearization with a continuous encoder-decoder pair [5, 32, 25], and empirical evidence supports this in practice [10]. In such multistable systems, it is possible to linearize the dynamics within basins, but the linearizations needed in different basins of attraction are generally incompatible with each other. Therefore, for multi-attractor systems, the right objective is not to force all basins through one continuous global linearization, but to discover and use distinct local linear regimes.

This paper proposes inducing sparse latent structure as a practical and interpretable method for Koopman learning of multi-attractor systems (Figure 1). By inducing the state to activate only a small subset of coordinates in lifted latent space, we can incentivize different regions of the original state space to occupy different active coordinates, since nearby states should have similar latent embeddings, and thus enable local linearizations. In this sparse representation, nonzero coefficient values in a latent vector can provide continuous coordinates for prediction within a selected region, while the active indices provide a discrete support object that can be inspected across states and used to select which columns of the latent transition matrix and the decoder act on the code[1].

Sparse coding and LISTA-style encoders [13, 8, 1] provide the right inductive bias for this purpose. They naturally induce piecewise-linear, union-of-subspaces structure [30, 7, 9, 13], which suits this setting where different attractors or basins require different local linearizations in theory.

We hypothesize that sparse-latent Koopman autoencoders will assign local dynamical regimes to distinct latent support families, yielding a learned regime variable without basin supervision. Then,

[1]Concurrent work by Kuang et al. [21] explores a similar separation between latent support and feature magnitudes in action-conditioned world models.

periodically decoding and re-encoding the state [10] can refresh the quality of the latent embedding over time by re-selecting the active support when trajectories move across the state space.[2]

Contributions. We introduce Sparse Koopman Autoencoders (SKAEs), showing that sparse latent supports can serve as label-free regime variables within a single trained model with one global transition matrix. Across multibasin and chaotic benchmarks, SKAEs with either a LISTA or a sparse MLP encoder (i) improve periodically re-encoded long-horizon forecasting over dense-latent KAEs while (ii) producing supports that are functionally necessary for accurate rollouts and (iii) interpretable as emergent local dynamical regimes aligned with basin interiors, where LISTA encoders give the lowest basin-alignment entropy.

2 Problem formulation

The introduction motivates a multibasin version of Koopman learning. Given sampled trajectories of a nonlinear system, the aim is to learn a state-reconstructing latent representation whose evolution is linear, without being told which basin or local dynamical regime generated each observation. We formulate the problem at the cadence of the stored benchmark observations. For each system, let $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ be the observed state space and let

$$x_{k+1} = F_{\Delta t}(x_k), \qquad x_k \in \mathcal{X}, \tag{1}$$

where $x_k \in \mathbb{R}^{d_x}$ is the state at the kth stored sample, $F_{\Delta t}$ denotes the transformation to the next discrete observation step based on underlying dynamics, and Δt is the system-specific observation interval. Forecasting is therefore defined by repeatedly applying the map $F_{\Delta t}$, and a horizon of h means h applications of $F_{\Delta t}$. We denote k compositions as $F_{\Delta t}^{(k)}$. Because Δt may differ across systems, equal values of h may yield different physical time.

Basins of attraction. Many systems of interest are multistable: different initial conditions can approach different long-run behaviors. A fixed point is a state x^* satisfying $F_{\Delta t}(x^*) = x^*$. More generally, a trajectory may approach an attractor A, such as a stable equilibrium, limit cycle, finite set, or chaotic invariant set. The basin of attraction of A is the set of initial conditions whose trajectories converge to A:

$$\mathcal{B}(A) = \left\{ x_0 \in \mathbb{R}^{d_x} : \inf_{a \in A} \left\| F_{\Delta t}^{(k)}(x_0) - a \right\|_2 \to 0 \text{ as } k \to \infty \right\}. \tag{2}$$

When multiple attractors coexist, their basins typically partition the state space up to basin boundaries. This defines the setup for this paper. The task is to learn a representation that can support linear prediction where the appropriate linear law depends on where the state lies in the multibasin geometry.

Koopman theory. The stored-step map $F_{\Delta t}$ induces a Koopman operator $\mathcal{K}$ on scalar observables $g : \mathcal{X} \to \mathbb{R}$ by composition, $(\mathcal{K}g)(x) = g(F_{\Delta t}(x))$. Although $F_{\Delta t}$ may be nonlinear, $\mathcal{K}$ is linear on the space of observables. A finite-dimensional Koopman approximation searches for a vector observable $\Phi : \mathcal{X} \to \mathbb{R}^{d_z}$ and a matrix K such that $\Phi(F_{\Delta t}(x)) \approx K\Phi(x)$, while retaining enough information to reconstruct the state.

Koopman autoencoders. We study Koopman autoencoders (KAEs) in this setting. The model learns an encoder $E_\theta : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$, a decoder $D_\phi : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$, and a latent linear transition matrix $K \in \mathbb{R}^{d_z \times d_z}$. For observation x_k at arbitrary timestep k, the corresponding open-loop h-step forecast is

$$\hat{x}_{k+h} = D_\phi\big(K^h E_\theta(x_k)\big). \tag{3}$$

The same matrix K is applied at every step of the rollout, so any state-dependent structure required for prediction must be encoded in the learned representation rather than in a supplied regime label.

[2] Fathi et al. [10] introduced periodic re-encoding and also sketched an idea on sparse, block-structured latents for switching dynamics in their Appendix D. In comparison, we examine sparsity-inducing effects with LISTA encoders and make the idea of supports as regime variables falsifiable with concrete experiments.

Basin labels are withheld from training. In the intended training and deployment setting, the model is not given basin labels or trajectory-to-basin assignments. When benchmark basin labels are available, they are reserved for post-hoc evaluation. They are not used to train the encoder or transition, or to select re-encoding periods.

3 Method

3.1 Sparse supports as local regime variables

A finite-dimensional Koopman representation for the multibasin systems in Section 2 should encode two complementary quantities: (i) continuous coordinates that linearize the dynamics within a basin, and (ii) which local linear regime is active. We use sparsity to couple these two roles. If the state x is represented by a sparse latent code z, then the support of z can act as a regime variable, while the nonzero coefficient values provide coordinates for the corresponding local linear representation.

Given an overcomplete dictionary D, sparse coding estimates z by solving the Lasso objective

$$z^\star(x) = \arg\min_{z \in \mathbb{R}^{d_z}} \frac{1}{2} \|x - Dz\|_2^2 + \lambda_{\text{sc}} \|z\|_1 , \qquad \lambda_{\text{sc}} > 0, \tag{4}$$

which finds a code z with few non-zero coefficients to reconstruct the input x. In other words, it encourages a sparse solution to the under-specified problem (classically, ℓ_0 instead of ℓ_1 will find the sparsest solution with the fewest non-zero coefficients).

The classical solver to this problem (ISTA [4]) can be unrolled into a depth-Q network of ISTA iterations, yielding learned ISTA (LISTA) [13]. In our setting, LISTA is an encoder yielding the sparse code $z = E_\theta(x)$, where the *support* is the set of active z coordinates. With a linear decoder, the support determines which decoder columns participate in reconstructing x, and the associated nonzero coefficients specify coordinates within that selected reconstruction subspace. When D and z are learned jointly, as in sparse coding [30], this induces a data-adaptive partition of the input state space where each region can be locally reconstructed with a code having the same support.

We therefore posit that combining Koopman representations and sparse coding objectives in a sparse Koopman autoencoder (SKAE) implicitly captures multibasin attractor dynamics without explicit basin labels in training. The Koopman objective encourages the continuous coefficients on each support to evolve linearly, while the sparsity objective encourages different local regimes to use different active coordinates. Thus, the model can represent a multibasin system using a discrete support pattern to identify the active basin and continuous coefficients for its localized linear representation. Essentially, when using only one transition matrix K and one decoder, the support selects which of their columns act on the code. When ground-truth basin labels are available in benchmarks, we use them only after training to assess support–basin alignment or to construct diagnostic interventions.

3.2 Koopman autoencoder and sparse encoder

The predictor follows the KAE formulation in Section 2: an encoder, linear decoder, and latent transition, all of which are optimized jointly, produce the rollout in eq. (3). As in sparse coding [13], we constrain the columns of the decoder $D_\phi = D$ to be unit norm to avoid degenerate solutions. We experiment with K that is dense, block diagonal, or softly block-regularized.

Sparse encoding. Our main sparse encoder is LISTA [13]. It first computes a dense pre-code $c_t = f_\theta(x_t)$, then applies learned shrinkage refinements for $q = 0, \ldots, Q-1$:

$$u_t^{(0)} = \text{shrink}(c_t, \tau), \qquad u_t^{(q+1)} = \text{shrink}(S u_t^{(q)} + c_t, \tau), \qquad z_t = u_t^{(Q)}. \tag{5}$$

Here, $\text{shrink}(a, \tau) = \text{sign}(a) \max(|a| - \tau, 0)$ is applied element-wise, S is learned, and τ controls the threshold scale for shrinkage. The thresholding step creates exact zeros, so the encoder explicitly selects active latent coordinates while learning the selection rule from the prediction objective. Appendix B gives the benchmark-specific output and threshold choices.

Transition families. We vary the structure of K separately from the encoder. The dense transition leaves K unrestricted. The block-diagonal transition partitions latent coordinates into M fixed groups (using the basin count of each system in the multibasin benchmark) and constrains $K_{\text{block}} = \text{blkdiag}(K_1, \ldots, K_M)$. We also use a soft-block LISTA ablation that keeps K dense but penalizes cross-group entries, $\mathcal{L}_{\text{offblock}} = \sum_{i,j:\, g(i)\neq g(j)} |K_{ij}|$, where $g(i)$ is the fixed architectural group containing coordinate i.

3.3 Training objective

For each training window $x_0, \ldots, x_L$ of adjacent stored states, let $z_t = E_\theta(x_t)$ be the encoded states and $\hat{z}_t = K^t z_0$ the latent rollout from the first state. Each application of K is one stored step (eq. (1)). We use standard KAE losses and an ℓ_1 sparsity-inducing penalty on the encoded states, each averaged over the windows of a batch:

$$\begin{aligned} \mathcal{L}_{\text{pred}} &= \frac{1}{L\sqrt{d_x}} \sum_{t=1}^{L} \|D_\phi(\hat{z}_t) - x_t\|_2 , & \mathcal{L}_{\text{rec}} &= \frac{1}{L\sqrt{d_x}} \sum_{t=1}^{L} \|D_\phi(z_t) - x_t\|_2 , \\ \mathcal{L}_{\text{lin}} &= \frac{1}{L} \sum_{t=1}^{L} \|\hat{z}_t - z_t\|_2 , & \mathcal{L}_{\text{sp}} &= \frac{1}{L} \sum_{t=1}^{L} \|z_t\|_1 . \end{aligned} \tag{6}$$

The observation-space terms are normalized by $\sqrt{d_x}$ so that their scale is comparable across systems of different state dimension. The latent terms are left in their native scale because the latent dimension is fixed within each comparison. Appendix B gives the implemented loss scaling on each benchmark. The total objective is

$$\mathcal{L} = (\lambda_{\text{pred}}\mathcal{L}_{\text{pred}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{lin}}\mathcal{L}_{\text{lin}} + \lambda_{\text{sp}}\mathcal{L}_{\text{sp}}) + \mathcal{L}_{\text{struct}}. \tag{7}$$

Here $\mathcal{L}_{\text{struct}} = 0$ for the dense and block-diagonal transition variants. For the soft-block transition, $\mathcal{L}_{\text{struct}} = \lambda_{\text{off}}\mathcal{L}_{\text{offblock}}$, which probes whether a soft transition grouping is sufficient to induce useful support structure. The prediction and latent-linearity terms make the representation useful for Koopman rollout, the reconstruction term keeps the code tied to the observed state, and the sparsity term encourages selective activation.

3.4 Support objects

In this subsection, we introduce several notions of support that are used at evaluation time.

Let $z = E_\theta(x) \in \mathbb{R}^{d_z}$. A *support rule* first converts the latent code z into a Boolean active-coordinate vector $m(x) \in \{0,1\}^{d_z}$, where $m_i(x) = 1$ means that latent coordinate i is active for state x. Equivalently, the exact support is the index set $S(x) = \{i : m_i(x) = 1\}$.

Exact supports defined by an arbitrary threshold can be unstable: a small perturbation in x-space can lead to a different support in z, so a single basin may be covered by several nearby or partially overlapping active-coordinate vectors. We therefore distinguish exact supports from *support families*, which group nearby Boolean support vectors and test whether these groups form basin-scale objects. Support families in general are more robust and lead to less basin fragmentation.

We use two support constructions:

- **Absolute-threshold support.** The absolute support vector is the Boolean mask denoted $m_{\text{abs}}(x)$, where $m_{\text{abs},i}(x) = \mathbf{1}\{|z_i| > 10^{-3}\}$ for indices i. $S_{\text{abs}}(x) = \{i : m_{\text{abs},i}(x) = 1\}$ is the equivalent active-index set. $m_{\text{abs}}(x)$ is the input to the support family F_{abs} (defined next).
- **Absolute-threshold support family.** F_{abs} is a group label for exact supports, and every exact support m_{abs} is assigned an F_{abs} label. F_{abs} groups exact supports whose active-index sets substantially overlap. We define the overlap score as the Jaccard index, or intersection-over-union, between the active coordinates of two Boolean support vectors $m_{\text{abs}}(x)$.

  On an evaluation collection $\mathcal{X}$, the algorithm first counts all distinct Boolean vectors and then visits those distinct vectors from most to least frequent, following a leader-style threshold clustering rule. Each family f stores one fixed representative vector r_f: the exact support vector that created that family. For the next distinct vector m_{abs}, the algorithm computes the Jaccard index

$$J(m_{\text{abs}}, r_f) = \frac{\sum_i \mathbf{1}\{m_{\text{abs},i} = 1 \text{ and } r_{f,i} = 1\}}{\sum_i \mathbf{1}\{m_{\text{abs},i} = 1 \text{ or } r_{f,i} = 1\}},$$

with $J(0,0) = 1$ when both vectors are all zero. If the best existing representative has Jaccard similarity at least $\tau = 0.5$, every occurrence of m_{abs} receives that family label. Otherwise, m_{abs} starts a new family and becomes the representative r_f for that family. Pseudocode is provided in Appendix A. We write $F_{\text{abs}}(x)$ for the family assigned to $m_{\text{abs}}(x)$.

3.5 Periodic re-encoding

If a support indicates the currently relevant local dynamics, a long rollout should be able to update that support. We therefore evaluate periodic re-encoding, introduced by Fathi et al. [10]. Starting from $z_k = E_\theta(x_k)$ and a re-encoding period m, the model advances for m Koopman steps, decodes the predicted state, and encodes that prediction again:

$$\tilde{z}_{k+m} = K^m z_k, \qquad \tilde{x}_{k+m} = D_\phi(\tilde{z}_{k+m}), \qquad z_{k+m} = E_\theta(\tilde{x}_{k+m}). \tag{8}$$

The same procedure is repeated over the rollout horizon. Each rollout uses only the model's predicted state, not the true future state, a basin label, or an oracle for transitions between basins. For each trained model, we tune a separate re-encoding period m from a fixed grid on validation rollouts[3].

4 Experiments

Our experiments aim to evaluate if finite-dimensional Koopman representations can be learned for multi-attractor systems implicitly without resorting to side information such as basin labels. We first discuss the quality of learned Koopman representations from sparse KAEs via a forecasting-based evaluation. Next, in the context of sparse inference, we evaluate the importance of the specific active set of coordinates for forecasting within a basin of attraction. Finally, we evaluate alignment of support families F_{abs} with withheld basin labels.

Benchmark systems. The main benchmark for our experiments contains 15 procedurally generated two-dimensional multibasin systems. These systems are constructed by specifying potential wells around known attractor coordinates, and superimposing dynamics that encourage transitions to different far-away regimes. Because we specify the locations of the potential wells, the basin labels (the attractor that a state's trajectory reaches) are well-defined and used only for evaluation. We also use a subset of 10 chaotic systems from the Dysts repository [11, 12] with about 1,000 stored steps per dominant period[4] as an additional forecasting test. Analytic expressions, per-system timesteps, and other details are in Appendix G, H.

Training models. We compare six neural models shown in Table 1 by varying the encoder and latent transition structure and studying their effects in this setting. We test three LISTA sparse encoders (where LISTA-BD denotes block-diagonal latent transitions and LISTA-SB denotes soft-block regularization), two sparse-latent MLPs applying the sparsity penalty $\mathcal{L}_{\text{sp}}$ (6) with dense [10] or block-diagonal (BD) transitions, and a dense-latent MLP control baseline with no sparsity incentive. All models use a linear decoder with columns having unit norm constraint.

Most of these architectures are biased toward sparse inference by varying degrees. Our contribution is in studying how different ways of inducing sparsity can be useful. For LISTA, encoded states are sparse by construction via thresholding, and encoded latents get sparsity pressure from (6). For sparse MLPs, the encoded states have induced sparsity through the ReLU activations and ℓ_1 regularization [39, 24]. The dense MLP baseline has no intended zero-producing mechanism: it uses tanh activations instead of ReLU and has no ℓ_1 penalty. Each model is trained for 15 random seeds on each system before evaluation. Training details are reported in Appendix B.

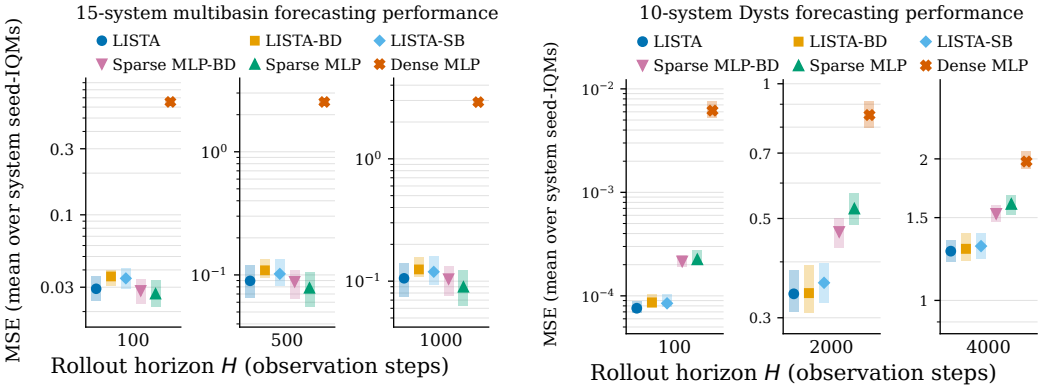
Statistical details. Point estimates are interquartile means (IQM) over seeds within each system to reduce seed uncertainty [2], then arithmetic means across systems. The markers of Figure 2 use the same point estimate as the tables, with seed-bootstrap intervals that summarize uncertainty over seeds for an average system. Statistical significance is rigorously tested with one-sided paired Wilcoxon signed-rank tests and Holm corrections (Appendix C).

[3] In general, there is no universal re-encoding period that is best for all systems.

[4] The dominant period is given by each system's Dysts metadata: the dominant Fourier timescale, i.e. the period of the strongest significant peak in the power spectrum of a long trajectory.

Table 1: Forecasting performance and basin-support diagnostics. $H(B \mid F_{\mathrm{abs}})$ is the entropy of the basin label given the support family and $\overline{|F_{\mathrm{abs}}|}$ is the number of support families. Values are means over systems of per-system seed IQMs. Direct baselines lift the state once whereas re-lifted ones re-lift it every m steps (all neural models use periodic re-encoding). $**$: lower than Dense MLP at $p < 0.05$, Holm-corrected, in every system. $*$ denotes the same significance in all systems but one. **Bold**: best in each $\downarrow$ column.

	Forecasting MSE↓						Support diagnostics	
	Multibasin, 15 systems			Dysts, 10 systems			Multibasin, 15 systems	
Model	H100	H500	H1000	H100	H2000	H4000	$H(B \mid F_{\mathrm{abs}})\downarrow$	$\overline{\lvert F_{\mathrm{abs}}\rvert}$
LISTA	0.0292^{**}	0.0895^{**}	0.106^{**}	$\mathbf{7.59\times10^{-5}}^{**}$	$\mathbf{0.339}^{**}$	$\mathbf{1.27}^{**}$	0.0348^{**}	4.3
LISTA-BD	0.0359^{**}	0.109^{**}	0.125^{**}	$8.64\times10^{-5}{}^{**}$	0.341^{**}	1.29^{**}	0.0560^{**}	4.2
LISTA-SB	0.0348^{**}	0.102^{**}	0.119^{**}	$8.47\times10^{-5}{}^{**}$	0.359^{**}	1.30^{**}	$\mathbf{0.0326}^{**}$	4.3
Sparse MLP-BD	0.0283^{**}	0.0881^{**}	0.104^{**}	$2.15\times10^{-4}{}^{**}$	0.467^{**}	1.53^{**}	0.102^{**}	3.9
Sparse MLP [10]	$\mathbf{0.0269}^{**}$	$\mathbf{0.0784}^{**}$	$\mathbf{0.0897}^{**}$	$2.25\times10^{-4}{}^{*}$	0.526^{**}	1.60^{**}	0.0679^{**}	4.0
Dense MLP *[baseline]*	0.654	2.54	2.91	0.00617	0.851	1.97	1.31	1.0
DMD [33, 36]	1.77	2.81	2.97	0.747	3.01	3.21	–	–
Poly. EDMD [37], direct	1.91	2.90	3.03	0.308	2.40	2.83	–	–
Poly. EDMD [37], re-lifted	1.73	2.71	2.86	0.00164	0.964	1.85	–	–
RBF EDMD [37], direct	1.94	2.86	4.43	0.0194	1.73	2.20	–	–
RBF EDMD [37], re-lifted	1.58	2.52	2.64	0.00171	1.00	2.00	–	–



(a) 15-system multibasin benchmark.

(b) 10-system Dysts benchmark.

Figure 2: Forecasting MSE at the horizons of Table 1, one panel per horizon. Sparse models forecast better than dense models. Markers show arithmetic means across per-system seed-IQM MSEs on a log scale. Shaded bars show fixed-system seed-bootstrap 95% intervals after system-wise log-relative normalization, anchored to the displayed mean, so they summarize seed uncertainty within an average system.

4.1 Forecasting experiments

Sparse encoders improve forecasting. The quality of the representation learned by a KAE is primarily measured by its ability to make accurate, multi-step forecasts. Thus, we evaluate the forecasting performance of trained sparse and dense-latent KAEs on the two benchmarks spanning 25 systems. Forecasting is evaluated on held-out trajectories at the stored observation cadence, with periodic re-encoding at a period chosen on validation, and summarized by mean squared error (MSE).

The results (Figure 2 and Table 1) show that sparse models yield a significant reduction of MSE forecasting error at all horizons on both sets of systems. In particular, at longer horizons, the dense MLP has 23 to 32 times larger MSE at $H = 1000$ on the multibasin systems and about 1.2 to 1.6 times larger MSE at $H = 4000$ on the Dysts systems compared to the sparse models. Figure 3 shows how the errors grow over time.

On the other hand, there are marginal differences between MSE predictions among the sparse-latent encoders on the multibasin benchmark (within a factor of 1.4 at every horizon). These results suggest that having a sparsity-inducing encoder is crucial for good performance over all horizon prediction lengths. The appendix reports per-system results (Appendix F), the classical baselines

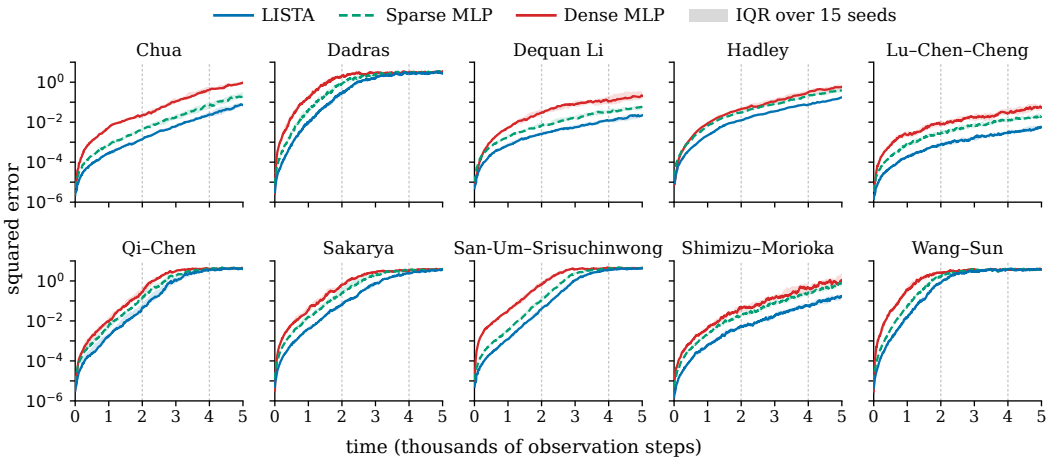
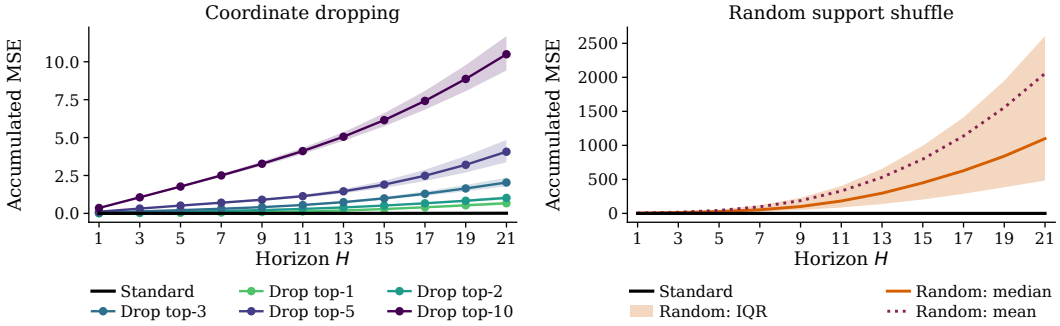


Figure 3: Forecast error over time on the ten Dysts systems, for the re-encoded forecasts of Table 1. At every step, we take the median squared error over the 100 test starts. The lines show the median over the 15 seeds and bands the interquartile range. Dotted lines mark the Table 1 horizons $H = 2000$ and $H = 4000$. Each system's dominant period is about 1,000 steps. In every system, LISTA's and Sparse MLP's errors are below Dense MLP's from step 304 to step 2,000.



(a) Dropping the largest active coordinates.

(b) Moving active values to inactive coordinates.

Figure 4: Support-coordinate interventions on a multibasin system. Interventions severely increase forecasting errors compared to the standard rollout. **(a)** The standard rollout is compared with cumulative dropping of the largest active coordinates of the initial sparse code. Shaded bands are bootstrap 95% intervals over 100 initial conditions. **(b)** Active coefficient values are preserved but reassigned to randomly chosen inactive coordinates. The band is the interquartile range over shuffles and initial conditions (20 shuffles each).

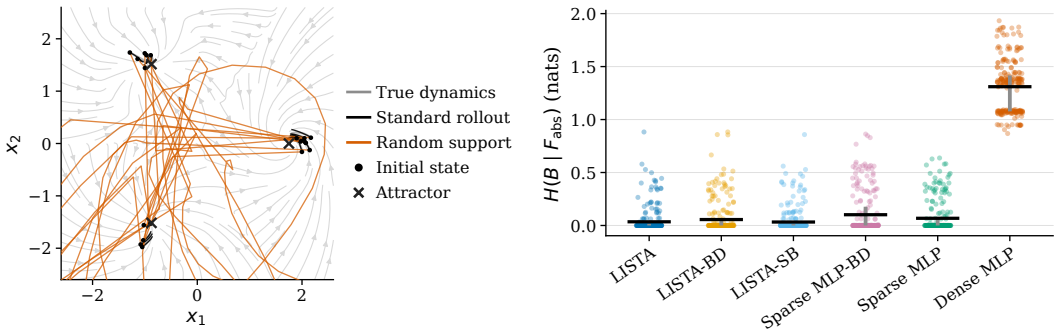
(Appendix D.1), ablations of the encoder's sparsity mechanisms (ReLU versus ℓ_1, Appendix D.2) and of periodic re-encoding (Appendix D.3), and a training-window ablation (Appendix D.4).

Sparse supports are essential for good forecasting. If a support formed by sparse encoders is meant to carry basin-relevant dynamical information, then changing only the initial active set should damage forecasts, even when the coefficient values are otherwise kept fixed. We do an ablation study to show whether accurate forecasts within a basin of attraction require the correct coordinates to be active by isolating the selection of sparse coordinates from their values. We do this by forcing a sparse LISTA-based KAE to use an incorrect active set with specific interventions (Figure 4), and evaluating 21-step forecasting performance under these settings. The interventions are the following:

1. **Coordinate dropping:** Force the coordinates of z with the $k \in \{1, 2, 3, 5, 10\}$ largest magnitude coefficients to zero before forecasting.
2. **Random support:** The active coefficient values are preserved, but reassigned to randomly chosen inactive latent coordinates.

Table 2: Accumulated MSE at $H = 21$ for the support interventions (lower is better).

Rollout	Mean ± SD	Median [IQR]
Standard	$\mathbf{0.0112 \pm 0.0140}$	$\mathbf{0.00735}$ $[\mathbf{0.00488}, \mathbf{0.0122}]$
Drop top-1	0.670 ± 0.639	0.542 $[0.189, 0.700]$
Drop top-2	1.02 ± 0.625	0.922 $[0.605, 1.19]$
Drop top-3	2.04 ± 1.49	1.67 $[1.30, 2.02]$
Drop top-5	4.07 ± 3.82	2.33 $[1.77, 4.48]$
Drop top-10	10.5 ± 5.67	8.41 $[6.22, 13.3]$
Random support	2060 ± 2590	1100 $[482, 2610]$



(a) Trajectories after random support shuffling.

(b) Basin-label entropy given the support family.

Figure 5: Support shuffling and support–basin alignment on the multibasin benchmark. **(a)** Rollouts after randomly shuffling the active coordinates (orange) move away from the standard rollouts (black). **(b)** Conditional entropy $H(B \mid F_{\text{abs}})$ of the basin label given the support family: the support families of the sparse models identify basins, whereas those of Dense MLP do not. Dots show system–seed pairs, gray I-bars the first-to-third quartile range of the per-system seed IQMs, and black bars the values of Table 1.

The results show that these interventions have a large detrimental effect on forecast quality, even on very short horizons. At $H = 21$, the standard rollout has mean accumulated MSE 0.0112, compared to 0.67/1.02/2.04/4.07/10.5 for the rollouts after dropping the top 1/2/3/5/10 active coordinates (Table 2), and random support shuffling is much more destructive as seen in Figure 4. The trajectory panel (Figure 5a) shows the qualitative effects of the intervention. So, supports formed by sparse encoders do not merely correlate with basin labels; the active coordinates are key to the quality of the Koopman representation.

4.2 Support–basin alignment experiments

This mechanistic experiment examines the extent to which sparse support families fit after training can behave like basin-interior regime variables. States near separatrices are challenging and may confound the inferences: near these states, support changes can be induced by small perturbations in x, making supports unreliable for basin identification at separatrices. We therefore evaluate this diagnostic on held-out states deep in basins far from basin boundaries.

For a state x in basin b (the attractor that its trajectory reaches), let $d_1(x)$ and $d_2(x)$ be its distances to the center of b and the nearest other benchmark attractor center. The *basin-depth margin* is $d_2(x) - d_1(x)$. This support–basin alignment experiment fits support families on validation trajectories and takes the held-out test states whose margin is positive and in the top quartile of the validation margins of their basin (the *high-margin subset*, Appendix B.4).

If a support family F_{abs} identifies a basin-specific regime used by the model, then on states deep inside a basin, it should leave little uncertainty about the withheld benchmark basin label B, measured by conditional entropy $H(B \mid F_{\text{abs}})$ as the main evaluation metric. $H(B \mid F_{\text{abs}})$ should be small when support families are basin-aligned. Also, different basins should be adequately represented by distinct support families, so we count the observed number of distinct support families, denoted as $|F_{\text{abs}}|$, and compare it with the number of basins represented.

Support families F_{abs} identify basin membership. As seen in Table 1 and Figure 5b, LISTA-based F_{abs} support families have the lowest mean conditional entropy (0.0326 to 0.0560). They are followed by the sparse-latent MLPs, and then finally the dense-latent MLP baseline. Appendix E.2 reports $H(B \mid F_{\mathrm{abs}})$ normalized by the basin-label entropy $H(B)$, which puts systems with different numbers of basins on one scale and leaves this ordering unchanged.

LISTA-based models also expose the most support families, which better matches the benchmark's basin-count mean/median 4.20/4. Ideally, if a basin can truly be recovered by a single support family cluster, then the basin count should be one-to-one with the number of support families, $\overline{|F_{\mathrm{abs}}|}$. In contrast, the dense-latent MLP, which has no mechanism for exact zeros, collapses the representation to one support family.

On the high-margin subset, support families fitted on validation trajectories match the basins of held-out test trajectories one-to-one in 175 of 225 system–seed cells for LISTA, 158 for Sparse MLP, and 0 for Dense MLP (Appendix E.1). This suggests that, on basin-interior states, the support families induced by sparse coding may serve as a good proxy for ground-truth basin labels, and that the sparse models may have the emergent ability to discover basins unsupervised away from basin boundaries.

5 Discussion

In this paper, we show that sparse Koopman autoencoders can learn useful local regime structure in multibasin dynamical systems without observing basin labels. Existing theory and empirical evidence indicate that multibasin systems generally should not be forced through one continuous finite-dimensional global Koopman embedding [5, 32, 10, 25]. Our results show that sparse codes offer a practical alternative: a single trained model *can* find good representations for multiple basins by using different latent supports. This gives practitioners a simple diagnostic to inspect after training.

For model selection, improvements in forecasting performance are more strongly driven by the sparsity incentives provided by the choice of activation functions and ℓ_1 sparsity-inducing regularization. However, the LISTA encoder, which has an inductive bias for finding sparse and specialized representations, can still yield tangible improvements on more complicated systems (such as the chaotic systems), and it can also improve interpretability of the discovered regimes.

We qualify some limitations. First, our interpretability claims are evaluated on benchmark systems where the basin geometry is known, and the support–basin alignment analysis is intentionally restricted to states deep inside basins. Near separatrices, small state changes can imply very different long-term outcomes, and support assignments become fragmented or unstable. This is not a failure mode unique to sparse Koopman models, but we suggest that support families should be interpreted as reliable basin-interior regime variables rather than perfect global basin classifiers. Moreover, support families come from a fixed post-hoc heuristic, not a learned objective. Learning them, e.g. contrastively, is future work.

Second, while the experiments include a diverse set of synthetic multibasin systems and external chaotic flows, they remain deterministic and fully observed low-dimensional benchmarks, because scoring alignment needs known basins and coverage of several basin interiors. Real-world systems may be high-dimensional, contain observation noise, partial observability, nonstationarity, control inputs, or regime changes that are not cleanly basin-like.

More broadly, this work argues that interpretability in dynamical representation learning can come from the structure of the latent code. Sparse supports provide an interface between continuous Koopman coordinates and discrete regime structure, making them a promising tool for multi-regime forecasting, analysis, and eventually control. By showing that these supports improve prediction, are necessary for accurate rollouts, and align with withheld basin information, this paper positions sparse latent structure as a practical step toward Koopman models for multibasin dynamics.

Acknowledgments and Disclosure of Funding

A.L. acknowledges support from the Fonds de recherche du Québec (FRQ) through a Master's Research Scholarship (DOI: https://doi.org/10.69777/2008193). S.C. is supported by the Canada CIFAR AI Chairs program, the Canada Research Chair in Lifelong Machine Learning, and the NSERC Discovery Grant. This research was enabled in part by compute resources, software and technical help provided by Mila (mila.quebec) and by the Digital Research Alliance of Canada (alliancecan.ca). The authors have no competing interests to declare.

References

[1] Aviad Aberdam, Alona Golts, and Michael Elad. Ada-LISTA: Learned Solvers Adaptive to Varying Models. *IEEE transactions on pattern analysis and machine intelligence*, 44(12): 9222–9235, December 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3125041.

[2] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[3] Omri Azencot, N. Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting Sequential Data Using Consistent Koopman Autoencoders. In *Proceedings of the 37th International Conference on Machine Learning*, pages 475–485. PMLR, November 2020.

[4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[5] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control. *PLOS ONE*, 11(2), February 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0150171. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0150171.

[6] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern Koopman Theory for Dynamical Systems. *SIAM Review*, 64(2):229–340, May 2022. ISSN 0036-1445. doi: 10.1137/21M1401243. URL https://doi.org/10.1137/21M1401243.

[7] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998. doi: 10.1137/S1064827596304010. URL https://doi.org/10.1137/S1064827596304010.

[8] Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. Hyperparameter Tuning is All You Need for LISTA. In *Advances in Neural Information Processing Systems*, volume 34, pages 11678–11689. Curran Associates, Inc., 2021.

[9] David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5): 2197–2202, March 2003. doi: 10.1073/pnas.0437847100. URL https://doi.org/10.1073/pnas.0437847100.

[10] Mahan Fathi, Clement Gehring, Jonathan Pilault, David Kanaa, Pierre-Luc Bacon, and Ross Goroshin. Course correcting koopman representations. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=A18gWgc5mi.

[11] William Gilpin. Chaos as an interpretable benchmark for forecasting and data-driven modelling. In *Advances in Neural Information Processing Systems (NeurIPS) 2021*, 2021. URL http://arxiv.org/abs/2110.05266.

[12] William Gilpin. Model scale versus domain knowledge in statistical forecasting of chaotic systems. *Physical Review Research*, 5(4):043252, December 2023. doi: 10.1103/PhysRevResearch.5.043252. URL https://link.aps.org/doi/10.1103/PhysRevResearch.5.043252.

[13] Karol Gregor and Yann LeCun. Learning Fast Approximations of Sparse Coding. In *Proceedings of the 27th International Conference on Machine Learning*, ICML'10, pages 399–406, Haifa, Israel, 2010. Omnipress. ISBN 978-1-60558-907-7. doi: 10.5555/3104322.3104374. URL https://dl.acm.org/doi/abs/10.5555/3104322.3104374.

[14] D. M. Grobman. Homeomorphism of systems of differential equations. *Doklady Akademii Nauk SSSR*, 128:880–881, 1959.

[15] Lars Grüne and Jürgen Pannek. *Nonlinear Model Predictive Control*. Communications and Control Engineering. Springer International Publishing, Cham, 2017. ISBN 978-3-319-46023-9 978-3-319-46024-6. doi: 10.1007/978-3-319-46024-6. URL http://link.springer.com/10.1007/978-3-319-46024-6.

[16] Philip Hartman. A lemma in the theory of structural stability of differential equations. *Proceedings of the American Mathematical Society*, 11(4):610–620, 1960. doi: 10.1090/S0002-9939-1960-0121542-7.

[17] R.E. Kalman. On the general theory of control systems. *1st International IFAC Congress on Automatic and Remote Control, Moscow, USSR, 1960*, 1(1):491–502, August 1960. ISSN 1474-6670. doi: 10.1016/S1474-6670(17)70094-8. URL https://www.sciencedirect.com/science/article/pii/S1474667017700948.

[18] B. O. Koopman. Hamiltonian Systems and Transformation in Hilbert Space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, May 1931. doi: 10.1073/pnas.17.5.315. URL https://www.pnas.org/doi/10.1073/pnas.17.5.315.

[19] B. O. Koopman and J. v. Neumann. Dynamical Systems of Continuous Spectra. *Proceedings of the National Academy of Sciences*, 18(3):255–263, March 1932. doi: 10.1073/pnas.18.3.255. URL https://www.pnas.org/doi/abs/10.1073/pnas.18.3.255.

[20] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, July 2018. ISSN 00051098. doi: 10.1016/j.automatica.2018.03.046. URL http://arxiv.org/abs/1611.03537.

[21] Yilun Kuang, Yash Dagade, Quentin Le Lidec, Lucas Maes, Randall Balestriero, and Yann LeCun. LpWM: A Case for Sparse Representations in World Models, August 2026. URL http://arxiv.org/abs/2608.22764.

[22] Matthew D. Kvalheim and Shai Revzen. Existence and uniqueness of global Koopman eigenfunctions for stable fixed points and periodic orbits. *Physica D: Nonlinear Phenomena*, 425:132959, November 2021. ISSN 0167-2789. doi: 10.1016/j.physd.2021.132959. URL https://www.sciencedirect.com/science/article/pii/S0167278921001160.

[23] Yueheng Lan and Igor Mezić. Linearization in the large of nonlinear systems and Koopman operator spectrum. *Physica D: Nonlinear Phenomena*, 242(1):42–53, January 2013. ISSN 0167-2789. doi: 10.1016/j.physd.2012.08.017.

[24] Oliver W. Layton, Siyuan Peng, and Scott T. Steinmetz. Relu, sparseness, and the encoding of optic flow in neural networks. *Sensors*, 24(23), 2024. ISSN 1424-8220. doi: 10.3390/s24237453.

[25] Zexiang Liu, Necmiye Ozay, and Eduardo D. Sontag. Properties of immersions for systems with multiple limit sets with implications to learning Koopman embeddings. *Automatica*, 176:112226, June 2025. ISSN 0005-1098. doi: 10.1016/j.automatica.2025.112226. URL https://www.sciencedirect.com/science/article/pii/S0005109825001189.

[26] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, November 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07210-0.

[27] Giorgos Mamakoukas, Maria Castano, Xiaobo Tan, and Todd Murphey. Local Koopman Operators for Data-Driven Control of Robotic Systems. In *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019. ISBN 978-0-9923747-5-4. doi: 10.15607/RSS.2019.XV.054. URL http://www.roboticsproceedings.org/rss15/p54.pdf.

[28] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000. ISSN 0005-1098. doi: 10.1016/S0005-1098(99)00214-9. URL https://www.sciencedirect.com/science/article/pii/S0005109899002149.

[29] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1–3):309–325, 2005. doi: 10.1007/s11071-005-2824-x.

[30] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996. ISSN 1476-4687. doi: 10.1038/381607a0. URL https://doi.org/10.1038/381607a0.

[31] Samuel E. Otto and Clarence W. Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019. doi: 10.1137/18M1177846.

[32] Shaowu Pan and Karthik Duraisamy. On the lifting and reconstruction of nonlinear systems with multiple invariant sets. *Nonlinear Dynamics*, 112(12):10157–10165, June 2024. ISSN 1573-269X. doi: 10.1007/s11071-024-09581-0. URL https://doi.org/10.1007/s11071-024-09581-0.

[33] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010. ISSN 0022-1120. doi: 10.1017/S0022112010001217.

[34] Haojie Shi and Max Q.-H. Meng. Deep Koopman Operator With Control for Nonlinear Systems. *IEEE Robotics and Automation Letters*, 7(3):7700–7707, July 2022. ISSN 2377-3766. doi: 10.1109/LRA.2022.3184036.

[35] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems 30*, volume 30, 2017. URL https://papers.nips.cc/paper/6713-learning-koopman-invariant-subspaces-for-dynamic-mode-decomposition.

[36] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, December 2014. ISSN 2158-2491. doi: 10.3934/jcd.2014.1.391. URL https://www.aimsciences.org/en/article/doi/10.3934/jcd.2014.1.391.

[37] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, December 2015. ISSN 1432-1467. doi: 10.1007/s00332-015-9258-5. URL https://doi.org/10.1007/s00332-015-9258-5.

[38] Matthew O. Williams, Clarence W. Rowley, and Ioannis G. Kevrekidis. A kernel-based method for data-driven koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, May 2016. ISSN 2158-2491. doi: 10.3934/jcd.2015005. URL https://www.aimsciences.org/article/doi/10.3934/jcd.2015005.

[39] Yuchen Zhang, Jason D. Lee, and Michael I. Jordan. L1-regularized neural networks are improperly learnable in polynomial time. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 993–1001, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/zhangd16.html.

A Support families

This appendix gives the pseudocode for the support families of Section 3.4.

> **Input:** validation states $\mathcal{X}$, their masks $m_{\mathrm{abs}}(x)$, and the Jaccard threshold $\tau = 0.5$.
> **Output:** a family label $F_{\mathrm{abs}}(x)$ for each $x \in \mathcal{X}$, and a representative r_f for each family f.
>
> 1. Count how often each distinct mask occurs, and sort the distinct masks from most to least frequent, breaking ties by a fixed lexicographic order of the masks.
> 2. For each distinct mask m_{abs} in this order: if no family exists or $\max_f J(m_{\mathrm{abs}}, r_f) < \tau$, create a new family with representative m_{abs}; otherwise, assign m_{abs} to the family that attains the maximum (the earliest one on ties).
> 3. Give each state x the family of its mask $m_{\mathrm{abs}}(x)$.

Family labels are arbitrary; only the partition they induce matters. Test states are assigned to these families as described in Appendix B.4.

Because the Jaccard index divides by the union of the two masks, the threshold τ adapts to the mask size. A lower τ merges more exact supports and can merge distinct basins; a higher τ separates more masks and can lower $H(B \mid F_{\mathrm{abs}})$ by fragmenting each basin. We therefore fix $\tau = 0.5$ instead of tuning it, read each entropy together with the family count, and report other thresholds in Appendix E.3.

B Additional experimental details

This appendix gives the settings of the experiments in Section 4; Appendices G and H describe the benchmarks and their data.

B.1 Models

All neural models (Table 3) use $d_z = 256$, MLPs with two hidden layers of width 64, a bias-free linear decoder, and K initialized at the identity. LISTA initializes its decoder dictionary D_0 from random orthonormal bases and S at $I - D_0^\top D_0/\rho$, where ρ is 1.05 times a power-iteration estimate of the largest eigenvalue of $D_0^\top D_0$, and uses $Q = 2$ refinements (eq. (5)). Table 4 gives its benchmark-specific code and threshold.

Table 3: Neural models. The last two rows are the encoder ablations of Appendix D.2.

Model	Encoder	Code sparsity	Latent transition
LISTA	LISTA, ReLU-MLP pre-code	shrinkage, ℓ_1	dense
LISTA-BD	LISTA, ReLU-MLP pre-code	shrinkage, ℓ_1	block-diagonal
LISTA-SB	LISTA, ReLU-MLP pre-code	shrinkage, ℓ_1	dense, off-block penalty
Sparse MLP	MLP, ReLU hidden and output	ReLU, ℓ_1	dense
Sparse MLP-BD	MLP, ReLU hidden and output	ReLU, ℓ_1	block-diagonal
Dense MLP	MLP, tanh hidden, linear output	none	dense
Sparse MLP, $\lambda_{\mathrm{sp}} = 0$	MLP, ReLU hidden and output	ReLU	dense
Dense MLP, ReLU hidden	MLP, ReLU hidden, linear output	none	dense

B.2 Training

All neural models of a benchmark share the settings of Table 4, except in the training-window ablation of Appendix D.4. Every reported result uses the final weights, and no seed is selected. We fixed this recipe after exploratory sweeps over LISTA depth, shrinkage scale, sparsity coefficient, sign-split encodings, transition structure, and MLP sparsity controls, with shorter budgets or broader system lists; they are not pooled with the reported results.

Loss scaling. On the multibasin benchmark, the four terms of eq. (7), but not $\mathcal{L}_{\mathrm{struct}}$, carry a further factor $1/L$, so the off-block penalty weighs $L\lambda_{\mathrm{off}} = 8\times10^{-4}$ relative to them. On Dysts, the terms are exactly those of eq. (6), computed on the standardized states.

Table 4: Settings of the neural models. The encoder and decoder learning rate also applies to the learned LISTA thresholds on Dysts.

	Multibasin	Dysts
Architecture		
LISTA code	128 coordinates, split by sign into 256; decoder initialized as $[D_0, -D_0]$	256 coordinates, nonnegative shrinkage $\max(a-\tau, 0)$; decoder initialized as D_0
LISTA threshold τ	fixed, $0.15/\rho$	learned per coordinate, initialized at $0.15/\rho$, shared by the refinements
BD and SB blocks	one per basin, contiguous, near-equal size	two of 128 coordinates
Training		
Seeds	$0, \ldots, 14$	$0, \ldots, 14$
Optimization steps	200,000	12,000
Minibatch	256 windows	512 windows
Window length	$L = 8$ (9 states)	$L = 63$ (64 states)
Training windows	2,048 per system and seed, fixed and cycled as 8 minibatches	resampled uniformly every step from the $200 \times 29{,}938$ start positions of the training split
Model inputs	raw states	re-standardized per coordinate with training-split statistics
Optimizer	AdamW	AdamW
Learning rate, encoder and decoder	5×10^{-5}	10^{-3}
Learning rate, K	5×10^{-6}	2×10^{-4}
Learning-rate schedule	constant	constant for 75% of the steps, then cosine decay to 1%
Weight decay	10^{-4}, all parameters but K	none
$\lambda_{\mathrm{pred}},\ \lambda_{\mathrm{rec}},\ \lambda_{\mathrm{lin}}$	$1,\ 0.03,\ 1$	$1,\ 1,\ 1$
λ_{sp} (Dense MLP: 0)	3×10^{-3}	0.03
λ_{off} (SB models)	10^{-4}	10^{-4}
Gradient clipping	none	none
Evaluation		
Test starts per system and seed	100 uniform draws in the box, separate seed stream	100 windows of the test split
Validation starts	100 uniform draws in the box, separate seed stream	100 windows of the validation split
Horizons H (observation steps)	100, 500, 1000	100, 2000, 4000
Re-encoding periods m	10, 25, 50, 100	10, 25, 50, 100, 150, 200

Multibasin training windows. Each window starts, with probability 0.5, from a uniform draw in the initial-condition box, and otherwise from a state of a label-free pool plus Gaussian jitter of standard deviation 0.01 times the box width. The pool, built before training to expose basin boundaries and transients, keeps the 1,024 top-scoring of 4,096 uniform draws, each probed for 32 steps and scored by its sensitivity to four Gaussian perturbations of standard deviation 0.04 times the box width (their mean final distance from the unperturbed state, divided by 0.04 times the box diagonal) plus 0.5 times its motion over the last 8 probe steps, divided by the box diagonal.

B.3 Forecast evaluation

The MSE at horizon H is the squared state error, summed over coordinates and averaged over steps $1, \ldots, H$ and over starts; on Dysts, predictions are mapped back to the stored coordinates first. The re-encoding period (Section 3.5) of each trained model minimizes the validation MSE at $H = 500$; if no period gives a finite error, the smallest is used. If a forecast becomes non-finite or its norm exceeds 10 times the largest state norm in the training data, it is replaced by the mean training state from that step on.

B.4 Support alignment and interventions

Basin labels. For each system and seed, we draw 64 validation and 64 test trajectories of 129 stored states from independent uniform draws in the initial-condition box. We continue each trajectory for 5,000 steps of the true system and take the mean of its final 500 states as its limit point. We cluster the validation limit points by k-means into the benchmark's basin count (farthest-first initialization, at most 25 iterations; on every system, no two centers are closer than 5% of the box diagonal), and each state takes the label of the center nearest to its trajectory's limit point. The basin-depth margins of Section 4.2 are measured to these centers, and ties at the high-margin threshold are kept.

Support families. A test support joins the family, fitted on the validation states, whose representative is most similar if their Jaccard similarity is at least 0.5, and is otherwise unmatched. Unmatched states count as one more family in the entropies but not in the family count. Entropies are in nats.

Support-coordinate interventions. The interventions of Figure 4 use one trained LISTA model (Local-linear gates, seed 0) and 100 starts: basin-balanced high-margin states that stay in their basin for the next 21 steps, taken from 512 separate trajectories of 65 stored states, which also give the basin centers and margins. An intervention edits only $z_0 = E_\theta(x_0)$, whose active set is $S_{\mathrm{abs}}(x_0)$ (Section 3.4), and the rollout is not re-encoded. The accumulated MSE at horizon H sums over steps $1, \dots, H$ the per-step squared error averaged over state coordinates, whereas the forecast MSE sums over coordinates and averages over steps.

B.5 Compute

The neural models were trained on NVIDIA L40S or H100 GPUs and the classical baselines on CPUs, for about 90 GPU hours and 200 CPU hours in total. Per-step cost differs between architectures, so equal step counts do not imply equal floating-point operations.

C Statistical testing protocol

The IQM of Section 4 is the mean of the seed values between the 25th and 75th percentiles, inclusive, with linearly interpolated percentiles; for 15 distinct values, it averages the middle 7.

Paired test. Significance marks summarize one-sided paired Wilcoxon signed-rank tests, run within each system over training seeds, with the candidate and the baseline paired by system and seed; we never pool seeds and systems into one sample. For MSE, the paired effect of system s, seed u, and horizon h is

$$\Delta_{s,u,h} = \log_{10} E^{\mathrm{cand}}_{s,u,h} - \log_{10} E^{\mathrm{base}}_{s,u,h},$$

with the alternative $\Delta < 0$, where E is the test MSE of the forecast in Table 1 (re-encoded for the neural models; the row's rollout for the classical baselines). Log ratios match the multiplicative question and reduce the influence of heavy-tailed seed-level MSEs. For $H(B \mid F_{\mathrm{abs}})$, the effect is the raw difference against Dense MLP, with the same alternative. The family count is not tested, because closeness to the basin count is not monotone in either direction. Every reported Wilcoxon test has all 15 paired seeds and no zero differences.

Holm correction. The per-system p-values of one candidate, baseline, metric, and horizon form one family, which is Holm-corrected at $\alpha = 0.05$: if $p_{(1)} \le \cdots \le p_{(m)}$ are the raw p-values, the adjusted value at rank i is $\max_{j \le i} \min\{1, (m - j + 1)\, p_{(j)}\}$. Every comparison is also tested in the reverse direction, as its own family. In Tables 1 and 6, $**$ means that the corrected test passes in every system, and $*$ that it passes in all but one; in the per-system tables of Appendix F, a mark means that the corrected test passes in that system.

Comparisons. The same test compares the sparse models with Dense MLP (Table 1) and with each classical baseline and rollout (Appendix D.1). In the encoder ablation (Appendix D.2), LISTA, Sparse MLP, and its ℓ_1-free variant are each compared with the ReLU-hidden Dense MLP; the ReLU-hidden and ℓ_1-free encoders with Dense MLP; and Sparse MLP with its ℓ_1-free variant. On Dysts, all models of one seed, neural and classical, are evaluated on the same 100 test starts.

Interventions. The intervals of Figure 4 describe the variability of one model on one system and are not hypothesis tests; the coordinate-dropping bands use 2,000 bootstrap resamples of the starts.

Bootstrap intervals. The 95% intervals of Figure 2 resample the seeds within each system 10,000 times; each interval is the 2.5th to 97.5th percentile of the mean over systems of the log change of the seed IQM, applied to the value of Table 1. The paired intervals of Appendix D.2 draw one resample of the seeds per system and use it for every model, 2,000 times.

D Additional forecasting results

D.1 Classical Koopman baselines

We compare the sparse KAEs with three classical Koopman approximations. Each standardizes the state with the training mean and standard deviation, lifts it by a fixed dictionary, fits one linear map between consecutive lifted states by ridge regression, and reads the state back from the degree-one features:

- **DMD** [33, 36]: the state itself; the map is linear on the multibasin benchmark and affine, $x_{t+1} = Ax_t + b$, on Dysts.
- **Polynomial EDMD** [37]: all monomials of the state up to degree three, including the constant (10 features in two dimensions and 20 in three).
- **RBF EDMD** [37]: the constant, the state, and N Gaussian features $\exp(-\|x - c\|^2/(2\sigma^2))$. The run seed draws the centers c from the standardized training states, and σ is the median distance over 8,192 random pairs of a 1,024-state subsample of them.

Direct rollouts lift the initial state once; re-lifted rollouts decode and lift the state again every m steps, the analog of periodic re-encoding (for DMD, the two coincide). Test starts, horizons, and error are those of the neural models (Appendix B.3); Table 5 gives the training data and selection.

Table 5: Training data and selection of the classical baselines. The ridge and, for re-lifted rollouts, the period m are selected jointly, as is N for RBF EDMD. The Dysts grids were fixed before the selection on the ten systems; the last row counts the selections at an edge of the grid.

	Multibasin	Dysts
Training data	the 2,048 windows of the neural run with the same seed	the full training split
Fits	one per system and seed	DMD and polynomial EDMD: one per system, so their seeds replicate only the test starts; RBF EDMD: one per system and seed
Selection starts	validation starts of the same seed	validation starts of a reserved seed (9000) that enters no reported result, per system
Selection criterion	lowest validation MSE at $H = 500$ of the row's own rollout, non-finite last; ties to fewer features, then the larger ridge, then the shorter period	lowest finite validation MSE at $H = 500$ of the row's own rollout
Ridge λ	0, 10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}, 1, 10; added to $\Phi^\top\Phi/n$	10^{-12}, 10^{-10}, …, 10^{-2}; applied as $\lambda\,\mathrm{tr}(G)/p$ for the Gram matrix G of the p features
Period m (re-lifted)	10, 25, 50, 100	10, 25, 50, 100, 150, 200
RBF features N	256, 1024	256, 1024
Edge selections	RBF EDMD: $N = 1024$ in 120 of 225 direct and 98 of 225 re-lifted fits; $\lambda = 0$ in 41 re-lifted fits (four systems) and no direct fit; $m = 10$ in 59 and $m = 100$ in 89 re-lifted fits; polynomial EDMD: $\lambda = 0$ in 3 re-lifted fits (two systems), $m = 10$ in 45 and $m = 100$ in 64 re-lifted fits; DMD: $\lambda = 0$ in 1 fit	$\lambda = 10^{-12}$: re-lifted polynomial and RBF EDMD on 7 of 10 systems each, direct RBF EDMD on 3; $\lambda = 10^{-2}$: DMD on 8, direct polynomial EDMD on 1 (Sakarya); $m = 10$ in 18 of 20 re-lifted choices; $N = 1024$ on every system

Table 1 gives the mean errors of the classical baselines and Appendix F the per-system values; the tests are those of Appendix C. On the multibasin benchmark, every sparse model has significantly lower error than every classical baseline, with either rollout, in all 15 systems at every horizon.

Among the multibasin baselines, DMD and polynomial EDMD are about as accurate as Dense MLP at $H = 1000$, and re-lifted RBF EDMD is the strongest at every horizon. The error of direct RBF EDMD grows after the selection horizon, because selection at $H = 500$ does not penalize slightly unstable fitted maps (spectral radius up to 1.007).

On Dysts, the outcome depends on the system, and the sparse KAEs and re-lifted EDMD win on different systems:

- **DMD.** LISTA and Sparse MLP have significantly lower error in all 10 systems at $H = 100$ and $H = 2000$.
- **Direct EDMD.** LISTA and Sparse MLP have significantly lower error than both direct rollouts in all 10 systems at $H = 2000$. At $H = 100$, LISTA beats direct polynomial EDMD in 9 systems (not Hadley) and Sparse MLP in 8; direct polynomial EDMD is significantly lower than Sparse MLP and Sparse MLP-BD in Hadley, and direct RBF EDMD than every sparse model in Dequan Li, Hadley, Lu–Chen–Cheng, and Shimizu–Morioka.
- **DMD and direct EDMD at $H = 4000$.** All three are significantly lower than every sparse model in Wang–Sun and than Sparse MLP in Dadras (direct RBF EDMD also than Sparse MLP-BD), and LISTA is significantly lower than each of them in the other 9 systems.
- **Re-lifted EDMD.** Re-lifted polynomial and RBF EDMD are significantly lower than every sparse model in Dequan Li, Hadley, Lu–Chen–Cheng, Qi–Chen, and Shimizu–Morioka at every horizon, often by orders of magnitude, and significantly higher than LISTA in the other five.

D.2 Sparsity mechanisms of the encoder

The MLP encoders of Table 3 form a ladder in which each step changes one component: ReLU hidden layers (the ReLU-hidden Dense MLP, whose code is still dense), a ReLU on the code (Sparse MLP with $\lambda_{\mathrm{sp}} = 0$), and the ℓ_1 penalty (Sparse MLP). The two intermediate encoders are trained like the main models (Appendix B); runs with the same seed share their training windows and validation and test starts, and Dense MLP and the two intermediate encoders also share their initial weights.

On this benchmark, the forecasting gain over Dense MLP comes from ReLU encoders, mostly from their hidden layers (Table 6):

- **ReLU hidden layers** make about half of the hidden activations exactly zero and lower the error of Dense MLP in every system. On the log scale, this step closes 79% to 85% of the gap in mean MSE between Dense MLP and the ℓ_1-free Sparse MLP.
- **A ReLU on the code** closes the rest: the ℓ_1-free Sparse MLP has a 1.56, 1.97, and 1.97 times lower mean MSE than the ReLU-hidden Dense MLP at $H = 100$, 500, and 1000 (paired seed-bootstrap 95% intervals $[1.34, 1.89]$, $[1.37, 2.82]$, and $[1.36, 2.85]$; median per-system ratio 2.6 to 2.8), significantly in most systems. No encoder with a sparse code has significantly higher error than the ReLU-hidden Dense MLP in any system.
- **The ℓ_1 penalty** makes the code sparser in all 225 paired runs, but it lowers the error significantly in only one system (SNIC multi-attractor) at each horizon, and raises it significantly in none.

D.3 Re-encoding ablation

Periodic re-encoding (Section 3.5) passes the predicted state back through the encoder, which corrects the latent coefficients and lets the active set change. Each segment between re-encodings is linear in the latent space, but the full forecaster, including the forecasts of Table 1, is an autonomous nonlinear composite of decoder, encoder, and K, not a pure K^h rollout. We separate the two effects with three rollouts of each trained multibasin model:

(a) no re-encoding: the direct K^h rollout;
(b) support-constrained re-encoding: re-encode every m steps, but keep only the coordinates in the initial active set $S_{\mathrm{abs}}(x_0)$;
(c) unrestricted re-encoding.

Table 6: Sparsity mechanisms of the encoder on the multibasin benchmark (15 systems, 15 paired seeds; tests of Appendix C). MSE: means over systems of per-system seed IQMs; $**$: significantly lower than Dense MLP in all 15 systems; $\S[1/15]$: significantly lower than Sparse MLP with $\lambda_{\text{sp}} = 0$ in 1 system at every horizon. Sparsity rows, as means over systems on 64 held-out trajectories per run: exactly zero encoded coordinates; coordinates with $|z_i| \leq 10^{-3}$ in the direct K^h rollout over 8 steps; and exactly zero activations in each hidden layer of the encoder MLP (for LISTA, its pre-code MLP).

	Dense MLP (tanh hidden)	Dense MLP, ReLU hidden	Sparse MLP, $\lambda_{\text{sp}} = 0$	Sparse MLP$^{\S[1/15]}$	LISTA
Forecasting MSE↓					
$H = 100$	0.654	0.0541^{**}	0.0347^{**}	0.0269^{**}	0.0292^{**}
$H = 500$	2.54	0.196^{**}	0.0997^{**}	0.0784^{**}	0.0895^{**}
$H = 1000$	2.91	0.229^{**}	0.116^{**}	0.0897^{**}	0.106^{**}
Systems with significantly lower / higher MSE than the ReLU-hidden Dense MLP ($H = 100/500/1000$)					
lower	0/0/0	–	12/10/11	14/14/14	13/13/13
higher	15/15/15	–	0/0/0	0/0/0	0/0/0
Fraction of inactive code coordinates					
encoded	1.11×10^{-7}	2.73×10^{-8}	0.524	0.634	0.607
rollout	0.0140	0.0114	0.518	0.622	0.615
Fraction of exactly zero hidden activations					
layer 1	8.41×10^{-9}	0.498	0.499	0.499	0.492
layer 2	0	0.538	0.491	0.497	0.559

Table 7: Re-encoding ablation on the multibasin benchmark at $H = 100$: (a) no re-encoding, (b) support-constrained, and (c) unrestricted re-encoding. Cells: system–seed cells in which the first condition has strictly lower test MSE than the second (there are no ties). Ratio: geometric mean over the 225 cells of the test-MSE ratio of (c) to (b). Own period: (b) at the period that minimizes its own validation error.

	Cells with lower test MSE			Ratio	(b) at its own period	
Model	(b) < (a)	(c) < (b)	(c) < (a)	(c)/(b)	(b) < (a)	(c) < (b)
LISTA	204/225	224/225	225/225	0.022	208/225	224/225
Sparse MLP	205/225	225/225	225/225	0.015	208/225	225/225
Dense MLP	213/225	192/225	225/225	0.72	223/225	175/225

Conditions (b) and (c) share one period per model and system, chosen from the grid of Table 4 by the mean over seeds of the log validation MSE of (c) at $H = 500$; test starts never enter the choice. Condition (c) thus differs from the reported forecasts, whose period is chosen per trained model, only in sharing its period across seeds. Dense MLP codes have essentially no exact zeros, and nearly every coordinate clears $|z_i| > 10^{-3}$, so the constraint in (b) removes few Dense MLP coordinates, and (b) stays much closer to (c) than for the sparse models.

Re-encoding helps even when the active set cannot change, so part of its gain is a generic re-projection; letting the support change adds a much larger gain for the sparse models than for Dense MLP (Table 7). Both findings hold when (b) is run at its own validation-selected period.

The validation-selected period of the reported forecasts is the smallest grid value, $m = 10$, in 1,486 of the 1,575 multibasin runs of the six main models and of Sparse MLP with $\lambda_{\text{sp}} = 0$, at the edge of the grid; most Dysts fits select $m = 50$ (662 of 900), and none selects $m = 200$.

D.4 Training-window ablation

Every multibasin training window applies K only $L = 8$ times, while the direct rollouts apply it up to 1,000 times. To test whether the short window limits the direct rollouts, we retrain LISTA, Sparse MLP, and Dense MLP with seeds $0, \ldots, 4$ on windows of 64 states ($L = 63$) and change nothing else. In detail:

- the factor $1/L$ of the multibasin loss rescales all four terms equally;

Table 8: Training-window ablation on the multibasin benchmark (15 systems, seeds $0, \ldots, 4$, paired with the main runs of the same seeds, so the 9-state rows differ from the 15-seed values of Table 1). Test MSE as means over systems of per-system seed IQMs, for the direct rollout and the reported re-encoded forecast; lower is better.

		Direct rollout			Re-encoded (reported)		
Model	Window	H100	H500	H1000	H100	H500	H1000
LISTA	9 states	14.3	64.9	40.1	0.0273	0.0843	0.098
	64 states	0.027	2.16	25	0.0232	0.0704	0.0825
Sparse MLP	9 states	16.3	46.8	25.3	0.0305	0.0804	0.0916
	64 states	0.0234	1.23	14.4	0.0217	0.0653	0.0775
Dense MLP	9 states	21.6	99.7	75.1	0.665	2.53	2.82
	64 states	0.199	2.69	71.2	0.189	0.959	1.49

- seed n shares its initial weights, window start states, and validation and test starts with the main run of seed n.

Longer windows make the sparse direct rollouts more accurate than DMD and polynomial EDMD (fit on the 9-state windows, as in Table 1) up to $H = 500$, but periodic re-encoding remains necessary for the reported accuracy at $H = 500$ and $H = 1000$ (Table 8):

- **Direct rollouts.** The sparse direct test MSE is lower than with 9-state windows in all 15 systems at $H = 100$ and $H = 500$ and in 13 at $H = 1000$ (sign test $p \leq 0.004$).
- **Direct rollouts against the baselines.** The sparse direct rollouts are lower than DMD and both polynomial EDMD rollouts in all 15 systems at $H = 100$ and in 14 at $H = 500$, but higher than DMD in 12 and 11 systems at $H = 1000$. They are lower than the Dense MLP direct rollout in all 15 systems at $H = 100$, in 11 (LISTA) and 14 (Sparse MLP) at $H = 500$, and in 13 at $H = 1000$.
- **Gain from re-encoding.** With 64-state windows, re-encoding still lowers the error in all 15 systems at $H = 500$ and $H = 1000$. At $H = 100$, it does so in only 12 and 10 systems (sign test $p = 0.02$ and 0.15), partly because many of these models select long periods ($m \geq 50$ in 29 of the 75 fits of each sparse model) and so re-encode at most once by $H = 100$.
- **Re-encoded forecasts of the sparse models.** They do not change consistently: they are lower in 6 to 9 of the 15 systems at each horizon (sign test $p \geq 0.3$), with per-system ratios from 0.03 to 48; the largest increase is Sparse MLP on Arrested spiral.
- **Re-encoded forecasts of Dense MLP.** They improve in 11, 14, and 14 systems ($p = 0.06$, 5×10^{-4}, and 5×10^{-4}), so its gap to the sparse models at $H = 1000$ narrows from about 29–31-fold to 18–19-fold; both sparse models still have lower re-encoded error than Dense MLP in all 15 systems at every horizon.

E Additional support–basin alignment results

This appendix extends Section 4.2. All results are on its high-margin subset of held-out test states, with the protocol of Appendix B.4.

E.1 Exact one-family-per-basin counts

A low $H(B \mid F_{\text{abs}})$ does not require one family per basin: it is also low when each basin splits into several families. We therefore count the system–seed cells in which families and basins correspond one-to-one (*exact matches*), that is, in which

- every test state joins one of the families fitted on the validation states;
- the family determines the basin, $H(B \mid F_{\text{abs}}) = 0$; and
- the basin determines the family, $H(F_{\text{abs}} \mid B) = 0$.

Every cell contains test states of at least two basins, so a single family, as in Dense MLP, never matches (Table 9). The five sparse models match in every seed of nine of the 15 systems, and most of their misses are in Arrested spiral, Octagonal eight-well, and Triple-well Duffing.

Table 9: Exact one-family-per-basin matches on the high-margin subset: seeds of each system, out of 15 (last row: system–seed cells, out of 225), whose test support families correspond one-to-one to the basins. Kept: mean fraction of the test states in the high-margin subset, the same for every model.

System	Kept	LISTA	LISTA-BD	LISTA-SB	Sparse MLP-BD	Sparse MLP	Dense MLP
Local-linear gates	0.224	15	15	15	15	15	0
Transfer-gated local-linear	0.202	15	13	15	15	11	0
Arrested spiral	0.349	2	1	0	0	0	0
Asymmetric three-well	0.323	15	15	15	15	15	0
High-cross three-well	0.255	15	15	15	15	15	0
Hexagonal six-well	0.255	14	10	11	7	10	0
Octagonal eight-well	0.309	2	0	1	0	0	0
Pentagonal five-well	0.272	15	15	15	15	15	0
Square four-well	0.400	15	15	15	15	15	0
Triple-well Duffing	0.374	0	0	0	0	0	0
SNIC multi-attractor	0.243	15	15	15	15	15	0
Transition-routes four-well	0.231	15	15	15	15	15	0
Depth-gradient four-well	0.304	15	15	15	15	15	0
Diamond four-well	0.352	15	15	15	15	15	0
L-shaped five-well	0.217	7	4	7	0	2	0
Total (of 225)		175	163	169	157	158	0

Table 10: Conditional entropy of the basin label given the support family on the high-margin subset, unnormalized (as in Table 1) and divided by the basin-label entropy $H(B)$: mean over the 15 systems of per-system seed IQMs; lower is better.

	LISTA	LISTA-BD	LISTA-SB	Sparse MLP-BD	Sparse MLP	Dense MLP
$H(B \mid F_{\mathrm{abs}})$ (nats)	0.0348	0.0560	0.0326	0.102	0.0679	1.31
$H(B \mid F_{\mathrm{abs}})/H(B)$	0.0221	0.0375	0.0206	0.0740	0.0470	1.00

E.2 Normalized conditional entropy

$H(B \mid F_{\mathrm{abs}})$ is at most the basin-label entropy $H(B)$, which grows with the number of basins (3 to 8 per system). Dividing it by $H(B)$, computed on the same states, gives a ratio in $[0, 1]$ (Table 10): 0 when the family determines the basin and 1 when it carries no information about it, as for the single family of Dense MLP.

E.3 Threshold sensitivity

We recompute the alignment over absolute thresholds $\{10^{-4}, 10^{-3}, 10^{-2}\}$ (the bound on $|z_i|$ that defines S_{abs}) and Jaccard thresholds $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ for the family merge and the assignment of test states; the paper uses 10^{-3} and 0.5. For LISTA and Sparse MLP, at every absolute threshold (Table 11):

- Jaccard thresholds below 0.5 merge the supports of different basins: the entropy rises and the number of exact matches falls.
- Thresholds of 0.6 and 0.7 lower the entropy and raise the number of exact matches relative to 0.5, so our fixed choice of 0.5 (Appendix A) is not necessarily the best one.

LISTA has a lower normalized entropy than Sparse MLP at every grid point except the absolute threshold 10^{-2} with Jaccard thresholds 0.6 and 0.7, where the two are within 0.002; the exact-match counts favor neither model consistently. Dense MLP, omitted from the table, forms a single family at every grid point.

Table 11: Threshold sensitivity of the support–basin alignment on the high-margin subset. Each cell gives the mean of $H(B \mid F_{\mathrm{abs}})/H(B)$ over the 225 system–seed cells (lower is better) and, in parentheses, the number of exact matches. Bold: the paper's setting; its entropy is a mean over cells, not the mean of per-system seed IQMs of Table 10.

		Jaccard threshold				
Model	Absolute threshold	0.3	0.4	0.5	0.6	0.7
LISTA	10^{-4}	0.258 (99)	0.107 (149)	0.035 (169)	0.005 (194)	0.001 (177)
	10^{-3}	0.200 (114)	0.092 (151)	**0.032 (175)**	0.004 (195)	0.000 (181)
	10^{-2}	0.127 (134)	0.056 (156)	0.018 (178)	0.006 (193)	0.001 (182)
Sparse MLP	10^{-4}	0.327 (70)	0.130 (135)	0.073 (153)	0.020 (190)	0.005 (192)
	10^{-3}	0.272 (91)	0.109 (142)	**0.055 (158)**	0.014 (193)	0.002 (192)
	10^{-2}	0.150 (132)	0.070 (152)	0.024 (185)	0.005 (199)	0.001 (189)

F Per-system forecasting results

This appendix gives the per-system test MSE behind the forecasting columns of Table 1, one table per benchmark and horizon. Each cell is the IQM over the 15 seeds of one system; the means of the columns over systems are the values in Table 1. The marks on the sparse models are the per-system tests of Appendix C.

F.1 Multibasin benchmark

Every sparse-model cell in Tables 12 to 14 carries all three marks.

F.2 Dysts benchmark

Dense MLP. In Tables 15 to 17, every sparse model has significantly lower error than Dense MLP in all 10 systems at every horizon, except Sparse MLP in Dequan Li at $H = 100$.

Classical baselines. The outcome depends on the system, and polynomial EDMD is a strong baseline on this benchmark: eight of the ten systems have quadratic vector fields (all but Chua and San-Um–Srisuchinwong), so its cubic dictionary contains their one-step map up to an error of third order in the step, which is about a thousandth of a period. As expected, re-lifted polynomial and RBF EDMD have significantly lower error than every sparse model in five of these systems, often by orders of magnitude (Appendix D.1). RBF EDMD, whose dictionary only approximates the one-step map, wins in the same systems by smaller margins.

Where the sparse models win. In the other five systems, all three LISTA models have significantly lower error than both re-lifted baselines at every horizon, and the two sparse MLPs do so in 53 of their 60 comparisons, never significantly higher. These five include three quadratic systems, Dadras, Sakarya, and Wang–Sun, where polynomial EDMD should have the advantage: its cubic dictionary contains the one-step map, whereas LISTA learns its features from data alone. Yet LISTA has significantly lower error in all three, and at $H = 100$ its error is about 37, 3, and 22 times lower than that of re-lifted polynomial EDMD, respectively. Here, a sparse KAE trained only on data outperforms a strong classical baseline whose dictionary matches the form of the dynamics.

System averages. Table 1 averages the per-system errors arithmetically, so each mean is the expected error on a randomly drawn system and emphasizes the hardest systems, where errors are largest.

Table 12: Multibasin test MSE per system at $H = 100$: IQM over 15 seeds; lower is better. Top: sparse models; bottom: baselines. $*$, †, ‡: significantly lower MSE than Dense MLP, DMD, or both polynomial EDMD rollouts in that system (Holm-corrected one-sided paired Wilcoxon test, $p < 0.05$). Tests against RBF EDMD are not marked.

System	LISTA	LISTA-BD	LISTA-SB	Sparse MLP-BD	Sparse MLP
Local-linear gates	9.78×10^{-4} *†‡	8.28×10^{-4} *†‡	9.41×10^{-4} *†‡	4.45×10^{-4} *†‡	5.38×10^{-4} *†‡
Transfer-gated local-linear	$0.235^{*\dagger\ddagger}$	$0.271^{*\dagger\ddagger}$	$0.236^{*\dagger\ddagger}$	$0.218^{*\dagger\ddagger}$	$0.223^{*\dagger\ddagger}$
Arrested spiral	$0.0122^{*\dagger\ddagger}$	$0.00927^{*\dagger\ddagger}$	$0.0147^{*\dagger\ddagger}$	$0.0125^{*\dagger\ddagger}$	$0.0124^{*\dagger\ddagger}$
Asymmetric three-well	$0.00814^{*\dagger\ddagger}$	$0.0266^{*\dagger\ddagger}$	$0.0157^{*\dagger\ddagger}$	$0.00751^{*\dagger\ddagger}$	$0.0112^{*\dagger\ddagger}$
High-cross three-well	$0.0835^{*\dagger\ddagger}$	$0.0865^{*\dagger\ddagger}$	$0.105^{*\dagger\ddagger}$	$0.0709^{*\dagger\ddagger}$	$0.0771^{*\dagger\ddagger}$
Hexagonal six-well	$0.0210^{*\dagger\ddagger}$	$0.0245^{*\dagger\ddagger}$	$0.0293^{*\dagger\ddagger}$	$0.0235^{*\dagger\ddagger}$	$0.0143^{*\dagger\ddagger}$
Octagonal eight-well	$0.0413^{*\dagger\ddagger}$	$0.0646^{*\dagger\ddagger}$	$0.0645^{*\dagger\ddagger}$	$0.0480^{*\dagger\ddagger}$	$0.0354^{*\dagger\ddagger}$
Pentagonal five-well	$0.0110^{*\dagger\ddagger}$	$0.0171^{*\dagger\ddagger}$	$0.0187^{*\dagger\ddagger}$	$0.00671^{*\dagger\ddagger}$	$0.0114^{*\dagger\ddagger}$
Square four-well	$0.00144^{*\dagger\ddagger}$	$0.00238^{*\dagger\ddagger}$	$0.00229^{*\dagger\ddagger}$	$0.00282^{*\dagger\ddagger}$	$0.00454^{*\dagger\ddagger}$
Triple-well Duffing	$0.00540^{*\dagger\ddagger}$	$0.00830^{*\dagger\ddagger}$	$0.00494^{*\dagger\ddagger}$	$0.00667^{*\dagger\ddagger}$	$0.00560^{*\dagger\ddagger}$
SNIC multi-attractor	2.73×10^{-4} *†‡	3.04×10^{-4} *†‡	3.00×10^{-4} *†‡	2.08×10^{-4} *†‡	2.02×10^{-4} *†‡
Transition-routes four-well	$0.00973^{*\dagger\ddagger}$	$0.00890^{*\dagger\ddagger}$	$0.00874^{*\dagger\ddagger}$	$0.0113^{*\dagger\ddagger}$	$0.00135^{*\dagger\ddagger}$
Depth-gradient four-well	$0.00155^{*\dagger\ddagger}$	$0.00216^{*\dagger\ddagger}$	$0.00409^{*\dagger\ddagger}$	$0.00193^{*\dagger\ddagger}$	$0.00124^{*\dagger\ddagger}$
Diamond four-well	$0.00466^{*\dagger\ddagger}$	$0.00662^{*\dagger\ddagger}$	$0.00460^{*\dagger\ddagger}$	$0.00727^{*\dagger\ddagger}$	$0.00405^{*\dagger\ddagger}$
L-shaped five-well	$0.00227^{*\dagger\ddagger}$	$0.00905^{*\dagger\ddagger}$	$0.0107^{*\dagger\ddagger}$	$0.00634^{*\dagger\ddagger}$	$0.00105^{*\dagger\ddagger}$

			Poly. EDMD		RBF EDMD	
System	Dense MLP	DMD	direct	re-lifted	direct	re-lifted
Local-linear gates	0.161	1.28	1.32	0.714	1.25	0.194
Transfer-gated local-linear	0.534	1.15	2.57	0.821	1.03	0.646
Arrested spiral	0.752	1.82	1.68	1.66	2.05	1.86
Asymmetric three-well	0.613	1.65	1.81	1.81	1.95	1.56
High-cross three-well	2.06	2.37	2.19	2.17	2.92	2.26
Hexagonal six-well	1.11	1.59	1.69	1.69	1.88	1.70
Octagonal eight-well	2.06	2.73	2.90	2.92	3.19	3.02
Pentagonal five-well	1.14	1.87	1.91	1.91	2.11	2.01
Square four-well	0.190	1.96	2.10	2.11	2.14	1.81
Triple-well Duffing	0.0540	1.35	0.404	0.109	0.480	0.0464
SNIC multi-attractor	0.0468	0.465	0.212	0.212	0.822	0.0550
Transition-routes four-well	0.200	2.06	2.98	2.98	2.24	2.24
Depth-gradient four-well	0.144	2.14	2.18	2.19	2.29	2.14
Diamond four-well	0.151	2.39	2.55	2.56	2.18	2.18
L-shaped five-well	0.586	1.80	2.09	2.14	2.63	1.98

Table 13: Multibasin test MSE per system at $H = 500$, as in Table 12.

System	LISTA	LISTA-BD	LISTA-SB	Sparse MLP-BD	Sparse MLP
Local-linear gates	$0.00154^{*\dagger\ddagger}$	$0.00146^{*\dagger\ddagger}$	$0.00109^{*\dagger\ddagger}$	$5.81\times10^{-4\,*\dagger\ddagger}$	$8.28\times10^{-4\,*\dagger\ddagger}$
Transfer-gated local-linear	$0.745^{*\dagger\ddagger}$	$0.862^{*\dagger\ddagger}$	$0.751^{*\dagger\ddagger}$	$0.664^{*\dagger\ddagger}$	$0.662^{*\dagger\ddagger}$
Arrested spiral	$0.0200^{*\dagger\ddagger}$	$0.0168^{*\dagger\ddagger}$	$0.0230^{*\dagger\ddagger}$	$0.0200^{*\dagger\ddagger}$	$0.0165^{*\dagger\ddagger}$
Asymmetric three-well	$0.0304^{*\dagger\ddagger}$	$0.0728^{*\dagger\ddagger}$	$0.0511^{*\dagger\ddagger}$	$0.0401^{*\dagger\ddagger}$	$0.0403^{*\dagger\ddagger}$
High-cross three-well	$0.156^{*\dagger\ddagger}$	$0.157^{*\dagger\ddagger}$	$0.212^{*\dagger\ddagger}$	$0.146^{*\dagger\ddagger}$	$0.155^{*\dagger\ddagger}$
Hexagonal six-well	$0.0353^{*\dagger\ddagger}$	$0.0395^{*\dagger\ddagger}$	$0.0497^{*\dagger\ddagger}$	$0.0392^{*\dagger\ddagger}$	$0.0247^{*\dagger\ddagger}$
Octagonal eight-well	$0.0612^{*\dagger\ddagger}$	$0.0912^{*\dagger\ddagger}$	$0.0907^{*\dagger\ddagger}$	$0.0692^{*\dagger\ddagger}$	$0.0514^{*\dagger\ddagger}$
Pentagonal five-well	$0.0210^{*\dagger\ddagger}$	$0.0315^{*\dagger\ddagger}$	$0.0318^{*\dagger\ddagger}$	$0.0110^{*\dagger\ddagger}$	$0.0210^{*\dagger\ddagger}$
Square four-well	$0.00100^{*\dagger\ddagger}$	$0.00182^{*\dagger\ddagger}$	$0.00182^{*\dagger\ddagger}$	$0.00242^{*\dagger\ddagger}$	$0.00908^{*\dagger\ddagger}$
Triple-well Duffing	$0.0409^{*\dagger\ddagger}$	$0.0453^{*\dagger\ddagger}$	$0.0472^{*\dagger\ddagger}$	$0.0391^{*\dagger\ddagger}$	$0.0434^{*\dagger\ddagger}$
SNIC multi-attractor	$3.79\times10^{-4\,*\dagger\ddagger}$	$3.63\times10^{-4\,*\dagger\ddagger}$	$4.73\times10^{-4\,*\dagger\ddagger}$	$2.97\times10^{-4\,*\dagger\ddagger}$	$2.60\times10^{-4\,*\dagger\ddagger}$
Transition-routes four-well	$0.165^{*\dagger\ddagger}$	$0.225^{*\dagger\ddagger}$	$0.174^{*\dagger\ddagger}$	$0.184^{*\dagger\ddagger}$	$0.111^{*\dagger\ddagger}$
Depth-gradient four-well	$9.41\times10^{-4\,*\dagger\ddagger}$	$0.00163^{*\dagger\ddagger}$	$0.00886^{*\dagger\ddagger}$	$0.00141^{*\dagger\ddagger}$	$7.27\times10^{-4\,*\dagger\ddagger}$
Diamond four-well	$0.0587^{*\dagger\ddagger}$	$0.0674^{*\dagger\ddagger}$	$0.0634^{*\dagger\ddagger}$	$0.0887^{*\dagger\ddagger}$	$0.0384^{*\dagger\ddagger}$
L-shaped five-well	$0.00536^{*\dagger\ddagger}$	$0.0216^{*\dagger\ddagger}$	$0.0240^{*\dagger\ddagger}$	$0.0156^{*\dagger\ddagger}$	$5.81\times10^{-4\,*\dagger\ddagger}$

			Poly. EDMD		RBF EDMD	
System	Dense MLP	DMD	direct	re-lifted	direct	re-lifted
Local-linear gates	0.226	2.66	2.53	1.42	2.70	0.238
Transfer-gated local-linear	1.12	2.25	3.68	2.39	2.01	1.62
Arrested spiral	2.52	2.42	2.37	2.36	2.54	2.47
Asymmetric three-well	2.81	2.64	2.66	2.66	2.68	2.65
High-cross three-well	4.77	2.94	2.90	2.83	3.05	2.79
Hexagonal six-well	4.14	2.42	2.43	2.45	2.60	2.48
Octagonal eight-well	7.47	4.19	4.21	4.24	4.40	4.31
Pentagonal five-well	4.66	2.79	2.79	2.77	2.79	2.76
Square four-well	0.282	2.90	2.92	2.92	2.92	2.90
Triple-well Duffing	0.342	1.07	1.04	0.623	1.02	0.271
SNIC multi-attractor	0.152	0.769	0.763	0.783	0.863	0.175
Transition-routes four-well	2.95	4.90	5.01	5.01	4.90	4.90
Depth-gradient four-well	0.235	3.00	2.99	3.01	2.99	3.00
Diamond four-well	0.975	4.03	4.04	4.04	4.08	4.08
L-shaped five-well	5.41	3.16	3.19	3.17	3.35	3.22

Table 14: Multibasin test MSE per system at $H = 1000$, as in Table 12.

System	LISTA	LISTA-BD	LISTA-SB	Sparse MLP-BD	Sparse MLP
Local-linear gates	$0.00167^{*\dagger\ddagger}$	$0.00161^{*\dagger\ddagger}$	$0.00110^{*\dagger\ddagger}$	$6.34\times10^{-4}{}^{*\dagger\ddagger}$	$8.73\times10^{-4}{}^{*\dagger\ddagger}$
Transfer-gated local-linear	$0.811^{*\dagger\ddagger}$	$0.913^{*\dagger\ddagger}$	$0.824^{*\dagger\ddagger}$	$0.724^{*\dagger\ddagger}$	$0.711^{*\dagger\ddagger}$
Arrested spiral	$0.0210^{*\dagger\ddagger}$	$0.0175^{*\dagger\ddagger}$	$0.0241^{*\dagger\ddagger}$	$0.0209^{*\dagger\ddagger}$	$0.0170^{*\dagger\ddagger}$
Asymmetric three-well	$0.0332^{*\dagger\ddagger}$	$0.0785^{*\dagger\ddagger}$	$0.0555^{*\dagger\ddagger}$	$0.0440^{*\dagger\ddagger}$	$0.0440^{*\dagger\ddagger}$
High-cross three-well	$0.166^{*\dagger\ddagger}$	$0.166^{*\dagger\ddagger}$	$0.225^{*\dagger\ddagger}$	$0.155^{*\dagger\ddagger}$	$0.165^{*\dagger\ddagger}$
Hexagonal six-well	$0.0371^{*\dagger\ddagger}$	$0.0413^{*\dagger\ddagger}$	$0.0522^{*\dagger\ddagger}$	$0.0411^{*\dagger\ddagger}$	$0.0260^{*\dagger\ddagger}$
Octagonal eight-well	$0.0642^{*\dagger\ddagger}$	$0.0943^{*\dagger\ddagger}$	$0.0941^{*\dagger\ddagger}$	$0.0718^{*\dagger\ddagger}$	$0.0534^{*\dagger\ddagger}$
Pentagonal five-well	$0.0223^{*\dagger\ddagger}$	$0.0332^{*\dagger\ddagger}$	$0.0335^{*\dagger\ddagger}$	$0.0115^{*\dagger\ddagger}$	$0.0222^{*\dagger\ddagger}$
Square four-well	$9.37\times10^{-4}{}^{*\dagger\ddagger}$	$0.00174^{*\dagger\ddagger}$	$0.00178^{*\dagger\ddagger}$	$0.00235^{*\dagger\ddagger}$	$0.00963^{*\dagger\ddagger}$
Triple-well Duffing	$0.0708^{*\dagger\ddagger}$	$0.0753^{*\dagger\ddagger}$	$0.0825^{*\dagger\ddagger}$	$0.0709^{*\dagger\ddagger}$	$0.0751^{*\dagger\ddagger}$
SNIC multi-attractor	$4.05\times10^{-4}{}^{*\dagger\ddagger}$	$3.58\times10^{-4}{}^{*\dagger\ddagger}$	$4.71\times10^{-4}{}^{*\dagger\ddagger}$	$3.11\times10^{-4}{}^{*\dagger\ddagger}$	$2.67\times10^{-4}{}^{*\dagger\ddagger}$
Transition-routes four-well	$0.269^{*\dagger\ddagger}$	$0.343^{*\dagger\ddagger}$	$0.270^{*\dagger\ddagger}$	$0.287^{*\dagger\ddagger}$	$0.169^{*\dagger\ddagger}$
Depth-gradient four-well	$8.65\times10^{-4}{}^{*\dagger\ddagger}$	$0.00156^{*\dagger\ddagger}$	$0.00943^{*\dagger\ddagger}$	$0.00131^{*\dagger\ddagger}$	$6.72\times10^{-4}{}^{*\dagger\ddagger}$
Diamond four-well	$0.0797^{*\dagger\ddagger}$	$0.0785^{*\dagger\ddagger}$	$0.0894^{*\dagger\ddagger}$	$0.114^{*\dagger\ddagger}$	$0.0511^{*\dagger\ddagger}$
L-shaped five-well	$0.00577^{*\dagger\ddagger}$	$0.0254^{*\dagger\ddagger}$	$0.0257^{*\dagger\ddagger}$	$0.0167^{*\dagger\ddagger}$	$5.04\times10^{-4}{}^{*\dagger\ddagger}$

			Poly. EDMD		RBF EDMD	
System	Dense MLP	DMD	direct	re-lifted	direct	re-lifted
Local-linear gates	0.236	2.87	2.80	1.57	3.22	0.243
Transfer-gated local-linear	1.21	2.85	3.82	2.75	2.71	1.71
Arrested spiral	2.69	2.48	2.46	2.44	2.87	2.52
Asymmetric three-well	3.15	2.76	2.76	2.76	3.09	2.76
High-cross three-well	5.01	3.02	3.00	2.91	3.07	2.85
Hexagonal six-well	4.82	2.52	2.52	2.54	7.74	2.56
Octagonal eight-well	8.18	4.36	4.37	4.40	6.42	4.46
Pentagonal five-well	4.92	2.90	2.90	2.88	5.75	2.85
Square four-well	0.293	3.01	3.02	3.02	4.94	3.00
Triple-well Duffing	0.431	1.06	1.06	0.747	1.03	0.402
SNIC multi-attractor	0.168	0.810	0.833	0.854	2.57	0.192
Transition-routes four-well	4.06	5.26	5.32	5.32	5.47	5.26
Depth-gradient four-well	0.247	3.09	3.09	3.11	3.64	3.09
Diamond four-well	1.33	4.23	4.23	4.24	4.67	4.27
L-shaped five-well	6.86	3.31	3.33	3.30	9.32	3.37

Table 15: Dysts test MSE per system at $H = 100$ observation steps, as in Table 12.

System	LISTA	LISTA-BD	LISTA-SB	Sparse MLP-BD	Sparse MLP
Chua	$1.64\times10^{-5}{}^{*\dagger\ddagger}$	$1.71\times10^{-5}{}^{*\dagger\ddagger}$	$1.74\times10^{-5}{}^{*\dagger\ddagger}$	$3.19\times10^{-5}{}^{*\dagger\ddagger}$	$3.86\times10^{-5}{}^{*\dagger\ddagger}$
Dadras	$1.70\times10^{-4}{}^{*\dagger\ddagger}$	$1.78\times10^{-4}{}^{*\dagger\ddagger}$	$1.74\times10^{-4}{}^{*\dagger\ddagger}$	$4.24\times10^{-4}{}^{*\dagger\ddagger}$	$4.40\times10^{-4}{}^{*\dagger\ddagger}$
Dequan Li	$5.24\times10^{-5}{}^{*\dagger}$	$5.15\times10^{-5}{}^{*\dagger}$	$5.65\times10^{-5}{}^{*\dagger}$	$1.10\times10^{-4}{}^{*\dagger}$	$1.50\times10^{-4}{}^{\dagger}$
Hadley	$4.68\times10^{-5}{}^{*\dagger}$	$4.79\times10^{-5}{}^{*\dagger}$	$4.50\times10^{-5}{}^{*\dagger}$	$1.06\times10^{-4}{}^{*\dagger}$	$1.34\times10^{-4}{}^{*\dagger}$
Lu–Chen–Cheng	$7.83\times10^{-6}{}^{*\dagger}$	$8.72\times10^{-6}{}^{*\dagger}$	$8.27\times10^{-6}{}^{*\dagger}$	$2.59\times10^{-5}{}^{*\dagger}$	$2.99\times10^{-5}{}^{*\dagger}$
Qi–Chen	$2.33\times10^{-5}{}^{*\dagger}$	$2.73\times10^{-5}{}^{*\dagger}$	$2.64\times10^{-5}{}^{*\dagger}$	$7.43\times10^{-5}{}^{*\dagger}$	$8.97\times10^{-5}{}^{*\dagger}$
Sakarya	$1.82\times10^{-4}{}^{*\dagger\ddagger}$	$2.24\times10^{-4}{}^{*\dagger\ddagger}$	$2.42\times10^{-4}{}^{*\dagger\ddagger}$	$7.26\times10^{-4}{}^{*\dagger}$	$6.11\times10^{-4}{}^{*\dagger}$
San-Um–Srisuchinwong	$3.79\times10^{-5}{}^{*\dagger\ddagger}$	$4.15\times10^{-5}{}^{*\dagger\ddagger}$	$3.98\times10^{-5}{}^{*\dagger\ddagger}$	$1.02\times10^{-4}{}^{*\dagger\ddagger}$	$1.11\times10^{-4}{}^{*\dagger\ddagger}$
Shimizu–Morioka	$9.10\times10^{-6}{}^{*\dagger}$	$9.19\times10^{-6}{}^{*\dagger}$	$1.02\times10^{-5}{}^{*\dagger}$	$2.70\times10^{-5}{}^{*\dagger}$	$3.21\times10^{-5}{}^{*\dagger}$
Wang–Sun	$2.14\times10^{-4}{}^{*\dagger\ddagger}$	$2.59\times10^{-4}{}^{*\dagger\ddagger}$	$2.27\times10^{-4}{}^{*\dagger\ddagger}$	$5.26\times10^{-4}{}^{*\dagger\ddagger}$	$6.14\times10^{-4}{}^{*\dagger\ddagger}$

			Poly. EDMD		RBF EDMD	
System	Dense MLP	DMD	direct	re-lifted	direct	re-lifted
Chua	1.02×10^{-4}	0.00521	2.28×10^{-4}	1.36×10^{-4}	4.95×10^{-5}	4.69×10^{-5}
Dadras	0.0249	1.39	0.671	0.00621	0.0826	0.0108
Dequan Li	1.81×10^{-4}	0.576	1.26×10^{-4}	3.76×10^{-8}	8.66×10^{-6}	1.35×10^{-6}
Hadley	2.57×10^{-4}	0.0831	4.96×10^{-5}	1.09×10^{-8}	4.02×10^{-6}	3.64×10^{-7}
Lu–Chen–Cheng	7.05×10^{-5}	0.513	8.73×10^{-4}	8.13×10^{-8}	1.39×10^{-6}	1.63×10^{-7}
Qi–Chen	8.74×10^{-4}	0.915	0.0524	8.49×10^{-7}	0.00111	7.27×10^{-6}
Sakarya	0.0102	1.13	1.48	6.07×10^{-4}	0.0156	0.00458
San-Um–Srisuchinwong	0.00190	0.721	0.0317	0.00475	6.27×10^{-4}	1.75×10^{-4}
Shimizu–Morioka	8.80×10^{-5}	0.715	0.00555	1.15×10^{-10}	9.39×10^{-7}	6.03×10^{-7}
Wang–Sun	0.0231	1.42	0.836	0.00471	0.0939	0.00146

Table 16: Dysts test MSE per system at $H = 2000$ observation steps, as in Table 12.

System	LISTA	LISTA-BD	LISTA-SB	Sparse MLP-BD	Sparse MLP
Chua	$0.00625^{*\dagger\ddagger}$	$0.00738^{*\dagger\ddagger}$	$0.0116^{*\dagger\ddagger}$	$0.0181^{*\dagger\ddagger}$	$0.0248^{*\dagger\ddagger}$
Dadras	$0.777^{*\dagger\ddagger}$	$0.803^{*\dagger\ddagger}$	$0.927^{*\dagger\ddagger}$	$1.16^{*\dagger\ddagger}$	$1.26^{*\dagger\ddagger}$
Dequan Li	$0.0505^{*\dagger}$	$0.0446^{*\dagger}$	$0.0559^{*\dagger}$	$0.0781^{*\dagger}$	$0.0826^{*\dagger}$
Hadley	$0.0117^{*\dagger}$	$0.0111^{*\dagger}$	$0.0124^{*\dagger}$	$0.0234^{*\dagger}$	$0.0294^{*\dagger}$
Lu–Chen–Cheng	$0.00140^{*\dagger}$	$0.00176^{*\dagger}$	$0.00170^{*\dagger}$	$0.00490^{*\dagger}$	$0.00629^{*\dagger}$
Qi–Chen	$0.395^{*\dagger}$	$0.329^{*\dagger}$	$0.325^{*\dagger}$	$0.456^{*\dagger}$	$0.581^{*\dagger}$
Sakarya	$0.499^{*\dagger\ddagger}$	$0.491^{*\dagger\ddagger}$	$0.543^{*\dagger\ddagger}$	$0.670^{*\dagger}$	$0.769^{*\dagger}$
San-Um–Srisuchinwong	$0.270^{*\dagger\ddagger}$	$0.300^{*\dagger\ddagger}$	$0.354^{*\dagger\ddagger}$	$0.399^{*\dagger\ddagger}$	$0.481^{*\dagger\ddagger}$
Shimizu–Morioka	$0.0894^{*\dagger}$	$0.0847^{*\dagger}$	$0.0863^{*\dagger}$	$0.171^{*\dagger}$	$0.170^{*\dagger}$
Wang–Sun	$1.29^{*\dagger\ddagger}$	$1.33^{*\dagger\ddagger}$	$1.27^{*\dagger\ddagger}$	$1.68^{*\dagger\ddagger}$	$1.85^{*\dagger\ddagger}$

			Poly. EDMD		RBF EDMD	
System	Dense MLP	DMD	direct	re-lifted	direct	re-lifted
Chua	0.0627	1.95	0.899	0.140	0.725	0.0768
Dadras	2.00	2.79	2.72	2.36	2.40	4.18
Dequan Li	0.168	2.82	2.02	3.27×10^{-5}	0.699	0.00161
Hadley	0.0783	4.72	2.84	1.34×10^{-5}	1.15	1.29×10^{-4}
Lu–Chen–Cheng	0.0173	2.76	1.12	5.34×10^{-5}	0.592	1.19×10^{-4}
Qi–Chen	0.783	2.97	2.81	0.0587	2.38	0.175
Sakarya	1.17	2.77	2.79	0.861	2.20	1.98
San-Um–Srisuchinwong	1.11	3.15	2.98	3.83	2.39	0.605
Shimizu–Morioka	0.245	2.79	2.48	7.82×10^{-7}	1.71	9.52×10^{-4}
Wang–Sun	2.88	3.40	3.37	2.39	3.03	3.00

Table 17: Dysts test MSE per system at $H = 4000$ observation steps, as in Table 12.

System	LISTA	LISTA-BD	LISTA-SB	Sparse MLP-BD	Sparse MLP
Chua	$0.117^{*\dagger\ddagger}$	$0.105^{*\dagger\ddagger}$	$0.129^{*\dagger\ddagger}$	$0.209^{*\dagger\ddagger}$	$0.274^{*\dagger\ddagger}$
Dadras	$2.50^{*\dagger\ddagger}$	$2.54^{*\dagger\ddagger}$	2.58^{*}	2.92^{*}	3.03^{*}
Dequan Li	$0.127^{*\dagger}$	$0.126^{*\dagger}$	$0.148^{*\dagger}$	$0.189^{*\dagger}$	$0.233^{*\dagger}$
Hadley	$0.108^{*\dagger}$	$0.102^{*\dagger}$	$0.121^{*\dagger}$	$0.179^{*\dagger}$	$0.216^{*\dagger}$
Lu–Chen–Cheng	$0.00822^{*\dagger}$	$0.00999^{*\dagger}$	$0.00988^{*\dagger}$	$0.0321^{*\dagger}$	$0.0355^{*\dagger}$
Qi–Chen	$2.03^{*\dagger}$	$1.92^{*\dagger}$	$1.95^{*\dagger}$	$2.31^{*\dagger}$	$2.46^{*\dagger}$
Sakarya	$1.75^{*\dagger\ddagger}$	$1.79^{*\dagger\ddagger}$	$1.80^{*\dagger\ddagger}$	$2.23^{*\dagger\ddagger}$	$2.33^{*\dagger\ddagger}$
San-Um–Srisuchinwong	$2.07^{*\dagger\ddagger}$	$2.23^{*\dagger\ddagger}$	$2.25^{*\dagger\ddagger}$	$2.51^{*\dagger\ddagger}$	$2.61^{*\dagger\ddagger}$
Shimizu–Morioka	$0.361^{*\dagger}$	$0.311^{*\dagger}$	$0.395^{*\dagger}$	$0.687^{*\dagger}$	$0.676^{*\dagger}$
Wang–Sun	3.66^{*}	3.73^{*}	3.65^{*}	4.02^{*}	4.15^{*}

			Poly. EDMD		RBF EDMD	
System	Dense MLP	DMD	direct	re-lifted	direct	re-lifted
Chua	0.518	3.52	1.66	0.958	1.51	0.344
Dadras	3.64	2.83	2.79	3.94	2.63	6.56
Dequan Li	0.388	2.75	2.84	3.44×10^{-4}	1.10	0.0152
Hadley	0.399	4.70	4.23	1.30×10^{-4}	2.03	0.00121
Lu–Chen–Cheng	0.0819	2.88	1.67	6.96×10^{-4}	1.11	9.22×10^{-4}
Qi–Chen	2.94	3.05	2.96	0.917	2.75	1.38
Sakarya	2.79	2.80	2.81	2.90	2.50	3.82
San-Um–Srisuchinwong	3.36	3.27	3.19	5.22	2.90	2.88
Shimizu–Morioka	0.929	2.86	2.70	1.87×10^{-5}	2.25	0.0407
Wang–Sun	4.70	3.45	3.44	4.59	3.27	4.91

G Multibasin benchmark inventory and system definitions

Table 18 lists the 15 two-dimensional multibasin systems $\dot{x} = f_s(x)$ used in this paper, and Figure 1 shows four of them. Claude Opus 4.5 in Claude Code provided substantial help in procedurally generating candidates given initial specifications, from which these 15 systems were chosen.

Ground-truth vector-field visualizations. Figure 6 shows the continuous-time ground-truth vector fields for all 15 retained two-dimensional multibasin systems. Streamlines show the direction of $f_s(x)$, and black crosses mark the generator design centers (well, trap, or gate centers; for SNIC multi-attractor, the equilibria of its polar part without the added rotation and confinement terms), which can lie up to about 0.5 from the attractors; evaluation labels come from trajectory fates (Appendix B.4). The streamline colors use a panel-local $\log_{10} \|f_s(x)\|_2$ scale to reveal structure within each system; they should not be read as comparable speed scales across panels.

Gaussian-well family. Most systems use a common two-dimensional Gaussian potential with an independent rotational drift. For $x = (x_1, x_2) \in \mathbb{R}^2$, let $Rx = (x_2, -x_1)$ and

$$V_s(x) = \sum_{i=1}^{M_s} -a_{s,i} \exp\left(-\frac{\|x - c_{s,i}\|_2^2}{2\sigma_{s,i}^2}\right) + \gamma_s(x_1^4 + x_2^4).$$

The implemented vector field is

$$\dot{x} = -\nabla V_s(x) + \omega_s R x. \tag{9}$$

Define $p_i^{(M)}(r, \varphi) = r(\cos(2\pi i/M + \varphi), \sin(2\pi i/M + \varphi))$, $i = 0, \ldots, M-1$. The retained Gaussian-well systems instantiate eq. (9) with the parameters in Table 19.

The transition-routes system uses the same Gaussian-well form with centers

$$\{c_i\} = \{(-1.8, 1.8), (1.8, 1.8), (-1.8, -1.8), (1.8, -1.8)\},$$

$a_i = 3.0$, $\sigma_i = 0.6$, $\omega = 1.0$, and $\gamma = 0.03$, and adds a corridor-dependent rotation term:

$$\dot{x} = -\nabla V_s(x) + \big(1.0 + 0.3\, C_{\text{route}}(x_2)\big) R x, \qquad C_{\text{route}}(x_2) = e^{-(x_2-1.8)^2/0.3} + e^{-(x_2+1.8)^2/0.3}. \tag{10}$$

Native gated local-linear systems. Both native gated systems place three centers on a circular layout, but with different radii and region definitions. Let $Q(\alpha) = \left(\begin{smallmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{smallmatrix}\right)$ and $\alpha_b = 2\pi b/3$, $b \in \{0, 1, 2\}$.

For `gated_local_linear`, $c_b = 1.75(\cos\alpha_b, \sin\alpha_b)$. Inside the radius-1.05 core of basin b,

$$\dot{x} = A_b(x - c_b), \qquad A_b = Q(\alpha_b)\widetilde{A}_b Q(\alpha_b)^\top,$$

with

$$\widetilde{A}_0 = \left(\begin{smallmatrix} -0.9 & -1.2 \\ 1.2 & -0.9 \end{smallmatrix}\right), \quad \widetilde{A}_1 = \left(\begin{smallmatrix} -1.35 & 0.2 \\ -0.3 & -0.7 \end{smallmatrix}\right), \quad \widetilde{A}_2 = \left(\begin{smallmatrix} -0.7 & -0.1 \\ 0.5 & -1.2 \end{smallmatrix}\right).$$

Outside the cores, the active sector is the nearest angular sector $s(x)$ and the shared gate matrix

$$G = \left(\begin{smallmatrix} -1.35 & -0.9 \\ 0.9 & -1.35 \end{smallmatrix}\right)$$

drives the state as $\dot{x} = G(x - c_{s(x)})$.

For `gated_transfer_linear`, $c_b = 1.85(\cos\alpha_b, \sin\alpha_b)$. The implemented radii and widths are

$$\rho_{\text{core}} = 0.30, \quad \rho_{\text{source}} = 0.80, \quad \rho_{\text{exit}} = 0.60, \quad \beta_{\text{exit}} = 0.72,$$
$$w_{\text{chan}} = 0.22, \quad \delta_{\text{lane}} = 0.28, \quad \rho_{\text{hand}} = 0.45.$$

The core matrices use the same rotated form with templates

$$\left(\begin{smallmatrix} -1.0 & -1.1 \\ 1.1 & -1.0 \end{smallmatrix}\right), \quad \left(\begin{smallmatrix} -1.4 & 0.2 \\ -0.2 & -0.7 \end{smallmatrix}\right), \quad \left(\begin{smallmatrix} -0.8 & -0.3 \\ 0.5 & -1.3 \end{smallmatrix}\right).$$

For each ordered pair $p = (s, t)$, $s \neq t$, define

$$d_{s\to t} = \frac{c_t - c_s}{\|c_t - c_s\|_2}, \qquad n_{s\to t} = (-d_{s\to t,2}, d_{s\to t,1}), \qquad o_t = \frac{c_t}{\|c_t\|_2},$$

Table 18: The 15 multibasin benchmark systems. Every system has state dimension $d = 2$. Each stored observation is one fourth-order Runge–Kutta step of size Δt of $\dot{x} = f_s(x)$ from the previous stored state, and initial conditions are drawn uniformly from the listed box. Parameters of the Gaussian-well systems are in Table 19; the other systems are defined in the text. "Basins" is the number of designed attractors and of trajectory-fate labels (Appendix B.4). On Transfer-gated local-linear, about 40% of uniform starts instead stall at region boundaries, and the labeling groups these stall points with the designed attractors.

Display name and system key	Generator family	Parameters	d	Δt	Initial-condition box	Basins
Local-linear gates `gated_local_linear`	piecewise local-linear gates	$c_b = 1.75(\cos\alpha_b, \sin\alpha_b)$, core radius 1.05; $\widetilde{A}_b$, G in text	2	0.04	$[-2.6, 2.6]^2$	3
Transfer-gated local-linear `gated_transfer_linear`	piecewise transfer gates	$c_b = 1.85(\cos\alpha_b, \sin\alpha_b)$; radii, widths, speeds, and templates in text	2	0.04	$[-2.8, 2.8]^2$	3
Arrested spiral `arrested_spiral`	spiral capture with Gaussian traps	damping 0.3, rotation 2.0; traps at $(1.5, 0)$, $(0, 1.8)$, $(-1, -1)$, $(0.8, -1.5)$, and the origin	2	0.02	$[-3, 3]^2$	5
Asymmetric three-well `cal_asymmetric_3`	Gaussian wells with independent rotation	Table 19	2	0.03	$[-3, 3] \times [-2.5, 3]$	3
High-cross three-well `cal_high_cross_3`	Gaussian wells with stronger rotation	Table 19	2	0.03	$[-3.3, 3.3]^2$	3
Hexagonal six-well `cal_hexagon_6`	Gaussian wells with independent rotation	Table 19	2	0.03	$[-3.3, 3.3]^2$	6
Octagonal eight-well `cal_octagon_8`	Gaussian wells with independent rotation	Table 19	2	0.03	$[-4, 4]^2$	8
Pentagonal five-well `cal_pentagon_5`	Gaussian wells with independent rotation	Table 19	2	0.03	$[-3.3, 3.3]^2$	5
Square four-well `cal_square_4`	Gaussian wells with independent rotation	Table 19	2	0.03	$[-3.3, 3.3]^2$	4
Triple-well Duffing `duffing_triple_well_no_rotation`	triple-well Duffing oscillator without rotation	$a = 0.3$, $\delta = 0.5$, $\omega = 0$, $\epsilon = 0.003$, $s = 4.0$ (eq. (11))	2	0.02	$[-2, 2]^2$	3
SNIC multi-attractor `snic_multi`	polar SNIC-like phase dynamics	$\dot{\theta} = 1 - 1.2\cos 3\theta$, radial forcing 0.3, rotation 0.5, confinement 0.01	2	0.02	$[-2, 2]^2$	3
Transition-routes four-well `transition_routes_4`	route-augmented Gaussian wells	centers $(\pm 1.8, \pm 1.8)$, $(a, \sigma) = (3.0, 0.6)$, $(\omega, \gamma) = (1.0, 0.03)$, route gain 0.3 (eq. (10))	2	0.02	$[-3.5, 3.5]^2$	4
Depth-gradient four-well `var_depth_gradient_4`	Gaussian wells with depth gradient	Table 19	2	0.03	$[-3, 3]^2$	4
Diamond four-well `var_diamond_4`	Gaussian wells on a diamond layout	Table 19	2	0.03	$[-3.5, 3.5]^2$	4
L-shaped five-well `var_l_shape_5`	Gaussian wells on an L-shaped layout	Table 19	2	0.03	$[-3.5, 3.5]^2$	5

Table 19: Parameters for retained Gaussian-well systems. Repeated pairs in the (a_i, σ_i) column apply to every listed center.

System	$\{c_i\}$	(a_i, σ_i)	(ω, γ)
High-cross three-well	$\{p_i^{(3)}(1.8, \pi/2)\}$	$(3.0, 0.5)$	$(2.0, 0.03)$
Hexagonal six-well	$\{p_i^{(6)}(1.7, 0)\}$	$(2.0, 0.5)$	$(1.0, 0.03)$
Octagonal eight-well	$\{p_i^{(8)}(2.2, 0)\}$	$(3.0, 0.5)$	$(0.90, 0.02)$
Pentagonal five-well	$\{p_i^{(5)}(1.8, \pi/2)\}$	$(2.0, 0.5)$	$(1.1, 0.03)$
Square four-well	$\{p_i^{(4)}(1.8, \pi/4)\}$	$(3.0, 0.5)$	$(1.0, 0.03)$
Diamond four-well	$\{(0, 2.2), (2.2, 0), (0, -2.2), (-2.2, 0)\}$	$(2.5, 0.5)$	$(1.0, 0.02)$
L-shaped five-well	$\{(-1.5, 1.5), (-1.5, 0), (-1.5, -1.5), (0, -1.5), (1.5, -1.5)\}$	$(2.5, 0.5)$	$(1.0, 0.03)$
Asymmetric three-well	$\{(0, 1.8), (-1.5, -0.9), (1.5, -0.9)\}$	$\{(2.5, 0.55), (1.5, 0.4), (2.0, 0.5)\}$	$(1.0, 0.03)$
Depth-gradient four-well	$\{(-1.3, 1.3), (1.3, 1.3), (1.3, -1.3), (-1.3, -1.3)\}$	$\{(2.2, 0.55), (2.5, 0.5), (3.0, 0.5), (3.5, 0.5)\}$	$(1.3, 0.03)$

and $\chi_{s,t} = 1$ when $\sin(\alpha_s - \alpha_t) \geq 0$, otherwise $\chi_{s,t} = -1$. The implemented channel entry and handoff points are

$$e_p = c_s + \rho_{\text{source}} d_{s\to t} + \chi_{s,t}\delta_{\text{lane}} n_{s\to t},$$
$$h_p = c_t + \rho_{\text{hand}} o_t + \chi_{s,t}\delta_{\text{lane}}(-o_{t,2}, o_{t,1}),$$

with channel direction $d_p = (h_p - e_p)/\left\|h_p - e_p\right\|_2$, channel normal $n_p = (-d_{p,2}, d_{p,1})$, and channel length $\ell_p = (h_p - e_p) \cdot d_p$.

The vector field is piecewise analytic. Core regions $\left\|x - c_b\right\|_2 \leq \rho_{\text{core}}$ use $A_b(x - c_b)$. Source annuli use $0.65A_b(x - c_b)$, and other non-channel background regions use $0.50A_b(x - c_b)$ for the nearest center c_b. Exit wedges for $p = (s, t)$ are source-neighborhood states outside the core with $\left\|x - c_s\right\|_2 \geq \rho_{\text{exit}}$ and $(x - c_s) \cdot d_{s\to t} \geq \left\|x - c_s\right\|_2 \cos\beta_{\text{exit}}$; they use

$$\dot{x} = v_{\text{exit}} d_{s\to t} - \lambda_{\text{exit}}\big((x - c_s) \cdot n_{s\to t}\big) n_{s\to t}, \qquad v_{\text{exit}} = 1.0, \quad \lambda_{\text{exit}} = 1.8;$$

and channel rectangles satisfying

$$0 \leq (x - e_p) \cdot d_p \leq \ell_p, \qquad |(x - e_p) \cdot n_p| \leq w_{\text{chan}}$$

use

$$\dot{x} = v_{\text{chan}} d_p - \lambda_{\text{chan}}\big((x - e_p) \cdot n_p\big) n_p, \qquad v_{\text{chan}} = 1.55, \quad \lambda_{\text{chan}} = 2.8.$$

Channel rectangles take precedence over all other regions, including the cores.

Other specialized systems. The arrested spiral system has a background spiral flow plus four Gaussian traps and an origin trap:

$$\dot{x} = -0.3x + 2.0Rx - 4.0\sum_{i=1}^{4} e^{-\|x-c_i\|_2^2/(2\cdot 0.25)} \frac{x - c_i}{0.25} - 1.5e^{-\|x\|_2^2/0.3} \frac{x}{0.15} - 0.02(x_1^3, x_2^3),$$

with $c_i \in \{(1.5, 0), (0, 1.8), (-1, -1), (0.8, -1.5)\}$. Its fifth basin is the origin trap.

The triple-well Duffing system uses $x = (q, p)$ and

$$V(q) = q^6/6 - q^4/2 + aq^2, \qquad V'(q) = q^5 - 2q^3 + 2aq,$$

with $a = 0.3$, damping $\delta = 0.5$, rotation strength $\omega = 0$ (no rotation, so that each of the three wells holds an attractor), and soft cubic confinement $-\epsilon u^3/s^3$ with $\epsilon = 0.003$ and $s = 4.0$:

$$\begin{aligned} \dot{q} &= (1 + \omega)p - \epsilon q^3/s^3, \\ \dot{p} &= -V'(q) - \delta p - \omega q - \epsilon p^3/s^3. \end{aligned} \tag{11}$$

The SNIC system is implemented in polar form and then converted to Cartesian coordinates. With $\varepsilon_{\text{num}} = 10^{-8}$, $r^2 = x_1^2 + x_2^2 + \varepsilon_{\text{num}}$, $r = \sqrt{r^2}$, $\theta = \operatorname{atan2}(x_2, x_1)$, $c_\theta = x_1/(r + \varepsilon_{\text{num}})$, and $s_\theta = x_2/(r + \varepsilon_{\text{num}})$,

$$\dot{r} = r(1 - r^2) - 0.3\cos(3\theta), \qquad \dot{\theta} = 1 - 1.2\cos(3\theta).$$

The Cartesian vector field is

$$\dot{x}_1 = \dot{r}c_\theta - r\dot{\theta}s_\theta - 0.01x_1(x_1^2 + x_2^2) + 0.5x_2,$$
$$\dot{x}_2 = \dot{r}s_\theta + r\dot{\theta}c_\theta - 0.01x_2(x_1^2 + x_2^2) - 0.5x_1.$$

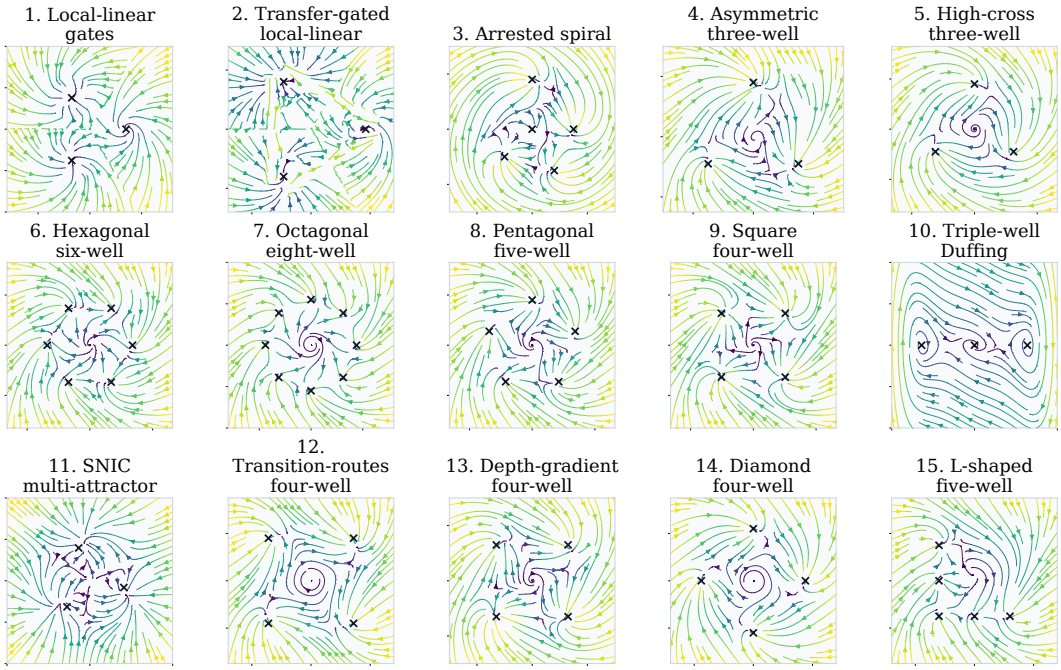


Figure 6: Ground-truth vector fields for the retained 15-system multibasin benchmark. The models are trained only on stored state trajectories; these continuous-time vector fields and design-center annotations are shown here only to document the benchmark geometry and are not provided to the learner.

H Dysts benchmark inventory and system definitions

Table 20 lists the ten Dysts systems [11, 12] of the forecasting benchmark, with right-hand sides and metadata from `dysts` 0.96 (Apache-2.0 license). The equations below are in native coordinates (x, y, z); trajectories are integrated and stored in Dysts' standardized coordinates (per-coordinate mean and standard deviation from the Dysts metadata). All right-hand sides are closed-form; most are polynomial, and the Chua and San-Um–Srisuchinwong systems include absolute values, so they are only piecewise smooth.

Trajectories and splits. Trajectories are integrated with DOP853 (relative and absolute tolerances 10^{-12}). Each split holds 200 trajectories per system of 30,001 stored states, after a discarded warm-up of 2,000 stored steps. The first training trajectory starts from the Dysts default initial condition; every other trajectory perturbs it with Gaussian noise of 0.2 times the coordinate-wise Dysts standard deviation, drawn from a separate deterministic seed stream per split, so no validation or test trajectory is a training trajectory.

Sakarya trajectories. Sakarya's flow is unbounded along the z-axis, so a Sakarya trajectory whose standardized state norm exceeds ten times the largest one in the training split is replaced by the next initial condition of its split's stream. This replaced four of the 600 Sakarya trajectories, two in validation and two in test.

Chua. The Chua system uses the piecewise-linear diode nonlinearity

$$h(x) = m_1 x + \frac{1}{2}(m_0 - m_1)\big(|x+1| - |x-1|\big),$$

with parameters

$$\alpha = 15.6, \qquad \beta = 28, \qquad m_0 = -1.142857, \qquad m_1 = -0.71429.$$

The vector field is

$$\begin{aligned} \dot{x} &= \alpha\big(y - x - h(x)\big), \\ \dot{y} &= x - y + z, \\ \dot{z} &= -\beta y. \end{aligned} \tag{12}$$

Dadras. With parameters

$$c = 2, \qquad e = 9, \qquad o = 2.7, \qquad p = 3, \qquad r = 1.7,$$

the Dadras system is

$$\begin{aligned} \dot{x} &= y - px + oyz, \\ \dot{y} &= ry - xz + z, \\ \dot{z} &= cxy - ez. \end{aligned} \tag{13}$$

Table 20: The ten Dysts systems used in the forecasting benchmark. The "stored step" column is 5 times the native Dysts integration interval, which gives 999.8 stored steps per dominant period for every system.

Display name	System key	Dimension	Native dt	Stored step $5dt$
Chua	`Chua`	3	$2.8474745791\times10^{-4}$	$1.4237372895\times10^{-3}$
Dadras	`Dadras`	3	$6.5782963827\times10^{-4}$	$3.2891481914\times10^{-3}$
Dequan Li	`DequanLi`	3	$1.6763993999\times10^{-5}$	$8.3819969995\times10^{-5}$
Hadley	`Hadley`	3	$2.9086847808\times10^{-4}$	$1.4543423904\times10^{-3}$
Lu–Chen–Cheng	`LuChenCheng`	3	$1.8469678280\times10^{-4}$	$9.2348391399\times10^{-4}$
Qi–Chen	`QiChen`	3	$7.8371061844\times10^{-5}$	$3.9185530922\times10^{-4}$
Sakarya	`Sakarya`	3	$9.9704617436\times10^{-4}$	$4.9852308718\times10^{-3}$
San-Um–Srisuchinwong	`SanUmSrisuchinwong`	3	$1.4933288881\times10^{-3}$	$7.4666444407\times10^{-3}$
Shimizu–Morioka	`ShimizuMorioka`	3	$2.4080013336\times10^{-3}$	$1.2040006668\times10^{-2}$
Wang–Sun	`WangSun`	3	$5.3924987498\times10^{-3}$	$2.6962493749\times10^{-2}$

Dequan Li. With parameters

$$a = 40, \qquad c = 1.833, \qquad d = 0.16, \qquad \varepsilon = 0.65, \qquad f = 20, \qquad k = 55,$$

the Dequan Li system is

$$\begin{aligned} \dot{x} &= a(y - x) + dxz, \\ \dot{y} &= kx + fy - xz, \\ \dot{z} &= cz + xy - \varepsilon x^2. \end{aligned} \tag{14}$$

Hadley. With parameters

$$a = 0.2, \qquad b = 4, \qquad f = 9, \qquad g = 1,$$

the Hadley system is

$$\begin{aligned} \dot{x} &= -y^2 - z^2 - ax + af, \\ \dot{y} &= xy - bxz - y + g, \\ \dot{z} &= bxy + xz - z. \end{aligned} \tag{15}$$

Lu–Chen–Cheng. With parameters

$$a = -10, \qquad b = -4, \qquad c = 18.1,$$

the Lu–Chen–Cheng system is

$$\begin{aligned} \dot{x} &= -\frac{ab}{a+b}x - yz + c, \\ \dot{y} &= ay + xz, \\ \dot{z} &= bz + xy. \end{aligned} \tag{16}$$

Qi–Chen. With parameters

$$a = 38, \qquad b = 2.666, \qquad c = 80,$$

the Qi–Chen system is

$$\begin{aligned} \dot{x} &= a(y - x) + yz, \\ \dot{y} &= cx + y - xz, \\ \dot{z} &= xy - bz. \end{aligned} \tag{17}$$

Sakarya. With parameters

$$a = -1, \qquad b = 1, \qquad c = 1, \qquad h = 1, \qquad p = 1, \qquad q = 0.4, \qquad r = 0.3, \qquad s = 1,$$

the Sakarya system is

$$\begin{aligned} \dot{x} &= ax + hy + syz, \\ \dot{y} &= -by - px + qxz, \\ \dot{z} &= cz - rxy. \end{aligned} \tag{18}$$

San-Um–Srisuchinwong. With parameter $a = 2$, the San-Um–Srisuchinwong system is

$$\begin{aligned} \dot{x} &= y - x, \\ \dot{y} &= -z\tanh(x), \\ \dot{z} &= -a + xy + |y|. \end{aligned} \tag{19}$$

Shimizu–Morioka. With parameters

$$a = 0.85, \qquad b = 0.5,$$

the Shimizu–Morioka system is

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= x - ay - xz, \\ \dot{z} &= -bz + x^2. \end{aligned} \tag{20}$$

Wang–Sun. With parameters

$$a = 0.2, \qquad b = -0.01, \qquad d = -0.4, \qquad e = -1, \qquad f = -1, \qquad q = 1,$$

the Wang–Sun system is

$$\begin{aligned} \dot{x} &= ax + qyz, \\ \dot{y} &= bx + dy - xz, \\ \dot{z} &= ez + fxy. \end{aligned} \tag{21}$$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have checked this and feel that our claims reflect the paper's contribution and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 explicitly identifies what the experiments do and do not establish.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: No original theorems or proofs or lemmas were included in this work.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 and Appendixes B, C, D.1, G, H contain benchmark populations, model families, training budgets, evaluation horizons, statistical units, and seed aggregation rules.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code, including the benchmark data generators, is available at `https://github.com/chandar-lab/skae`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendixes B, D.1, G, and H contain these details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The forecasting and support-alignment comparisons in Section 4 have statistical significance tests and more details provided in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Mentioned in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the code of ethics and believe our research complies.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The question being studied is slightly more theoretical, and it is unclear how this question would have societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper poses no such risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The dysts package from Gilpin [11, 12] is explicitly cited in the main text, and license is mentioned in Appendix H.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Details of the generated multibasin systems are provided in Section 4 and Appendix G.

    Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Claude Opus 4.5 in Claude Code helped procedurally generate candidate multibasin systems, from which the 15 benchmark systems were chosen (Appendix G); agent LLMs also assisted with code and the generation of figures.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.