WATCH YOUR NEIGHBORS: TRAINING STATISTICALLY ACCURATE CHAOTIC SYSTEMS WITH LOCAL PHASE SPACE INFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Chaotic systems pose fundamental challenges for data-driven surrogate modeling, as small modeling errors lead to exponentially growing trajectory discrepancies. Since exact long-term prediction is unattainable, it is natural to ask what a good surrogate model for chaotic dynamics is. Prior work has largely focused either on reproducing the Jacobian of the underlying dynamics, which governs local expansion and contraction rates, or on reproducing the ground-truth dynamics' long-term statistical behavior. We propose a framework that bridges these two paradigms by training surrogate models with accurate Jacobians and long-term statistical properties. Our method constructs a local covering of a chaotic attractor in phase space and analyzes the expansion and contraction of these coverings under the dynamics. The surrogate model is trained by minimizing the maximum mean discrepancy between the pushforward distributions of the coverings under the surrogate and ground-truth dynamics. Experiments show that our method significantly improves Jacobian accuracy while remaining competitive with state-of-the-art statistically accurate dynamics learning methods, at a computational cost that is a constant factor independent of the system dimension. Our code is fully available at `https://anonymous.4open.science/r/neighborwatch`.

1 INTRODUCTION

Modeling dynamical systems and generating accurate forecasts are central problems across the sciences. When governing equations are unavailable or intractable, data-driven discovery offers an alternative by training surrogate models from trajectory data. Typically, model parameters are learned by minimizing discrepancies between observed and predicted trajectories, yielding "black-box" models that can match or even outperform approximate theoretical descriptions (Rackauckas et al., 2021; Brunton & Kutz, 2022; North et al., 2023; Yu & Wang, 2024).

Chaotic systems pose unique challenges for data-driven dynamics as the trajectories of these systems are extremely sensitive to variations in the initial condition and model parameters. Unavoidable errors in measurements or model parameters therefore grow exponentially in time, rendering exact long-term prediction impossible, while models optimized on short-term predictions alone are myopic, quickly outputting unrealistic or diverging solutions when rolled out (Li et al., 2022; Hess et al., 2023; Schiff et al., 2024; Li et al., 2025). This incompatibility between trajectory-based optimization and chaotic dynamics necessitates a reevaluation of what constitutes a good surrogate model for chaos.

Recent works in the machine learning community have focused on training models that are "statistically accurate", meaning that even though the long-time model trajectory predictions cannot match that of the ground truth, expectation values computed along these trajectories must agree (Jiang et al., 2023; Platt et al., 2023; Schiff et al., 2024). The validity of such methods stems from the fact that many chaotic systems possess physical measures that are invariant with respect to their dynamics and are ergodic inside their chaotic attractor. Therefore, long-time trajectory statistics are invariant, global properties of these chaotic dynamics, serving as well-defined targets for the dynamics discovery task.

An alternate line of thought can be found in the dynamical systems literature, where the goal has been to train surrogate models that can accurately reproduce the Jacobian of the ground-truth dynamical equations (Abarbanel et al., 1992; Nychka et al., 1992; McCaffrey et al., 1992; Escot & Sandubete,

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

2023). This was largely motivated by the fact that Jacobians provide information about contracting and repelling dynamics of the phase space, which in turn give rise to the key characteristics of chaos, including the sensitivity to initial trajectory perturbations (Gilmore & Lefranc, 2011). From an alternative standpoint, the recent work of Eisen et al. (2025) has also highlighted the importance of learning accurate Jacobians for chaotic systems, for its pivotal role in identifying interacting subdomains in complex systems such as the brain.

In this work, we aim to bridge the two schools of thought by developing a training method that yields surrogate models for chaotic systems with accurate Jacobians and long-term statistical properties. Our work succeeds that of Park et al. (2025), where the connection between conditioning models with Jacobian information and their capability to reproduce accurate long-term statistical behaviors was also explored. However, their method relied on an oracle supplying the value of the ground-truth Jacobians, making it unsuitable for real-world applications.

Contributions We propose a framework for training Neural ODEs on noisy chaotic data that yields models which are statistically accurate and properly capture the local expansion and contraction of the chaotic attractor. We achieve this by focusing on how local covering sets of the attractor evolve in time and designing a specialized loss that regularizes training with this information. Experiments on three chaotic systems show that the method greatly improves Jacobian accuracy, even for noisy data, while retaining statistical accuracy. By design, the method's overhead relative to standard training is a constant factor independent of the system dimension.

2 Background

We consider dynamical systems of the form

$$\frac{d\boldsymbol{u}}{dt} = f(\boldsymbol{u}(t)), \tag{1}$$

where $\boldsymbol{u}(t) \in \mathbb{R}^d$ is the state vector and $f : \mathbb{R}^d \to \mathbb{R}^d$ is sufficiently smooth. Eq. (1) encompasses autonomous and non-autonomous ODEs,[1] as well as spatial discretizations of PDEs. Associated with Eq. (1) is the time-T map $\phi_T : \mathbb{R}^d \to \mathbb{R}^d$, which sends an initial condition $\boldsymbol{u}_0$ to the solution $\boldsymbol{u}(T)$ of Eq. (1) with $\boldsymbol{u}(0) = \boldsymbol{u}_0$ (Guckenheimer & Holmes, 1983). Since f is smooth, ϕ_T is a diffeomorphism, and $\phi_{nT} = \phi_T^n$ is its n-fold composition.

We are primarily interested in dissipative systems with a compact attracting invariant set $\mathcal{A} \subset \mathbb{R}^d$ on which the dynamics is chaotic, i.e., sensitive to initial conditions and aperiodic (Alligood et al., 1996), and which supports a physical (SRB) measure μ (Young, 2002): a Borel probability measure with support $\mathcal{A}$ that is invariant under every time-T map, $\mu(\phi_T^{-1}(B)) = \mu(B)$ for all Borel B, and for which there exists a set V of positive Lebesgue measure such that for every continuous $g : \mathbb{R}^d \to \mathbb{R}$ and every $x \in V$,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} g(\phi_T^i(x)) = \int_{\mathcal{A}} g \, d\mu.$$

That is, long-time averages along typical trajectories are governed by μ, which is what makes "statistical accuracy" well defined. Establishing the existence of an SRB measure rigorously is nontrivial even for classical systems - for the Lorenz attractor it required a combination of analysis and rigorous numerics (Tucker, 2002). Following previous work (Schiff et al., 2024; Park et al., 2025), we assume throughout that the systems under study admit one, as is known for uniformly hyperbolic attractors and the Lorenz attractor (Young, 2002).

Finally, for a measurable map ϕ, the pushforward of μ is the measure $\phi_{\#}\mu(B) = \mu(\phi^{-1}(B))$. In this notation, invariance of μ is precisely $(\phi_T)_{\#}\mu = \mu$, and $(\phi_{nT})_{\#}\mu = (\phi_T^n)_{\#}\mu = \mu$, $\forall n \in \mathbb{N}$.

3 Learning accurate surrogate dynamics models for chaos

In this work, we focus on the problem of training surrogate dynamics models for chaotic systems using trajectory measurements on the chaotic attractor. One powerful framework for learning continuous-time dynamical systems is the neural ordinary differential equation (Neural ODE) (Chen et al., 2018;

[1] By introducing an augmented state vector $\boldsymbol{v} = [t, \boldsymbol{u}]^\intercal$ (Strogatz, 2024).

Kidger, 2022), where a neural network $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^d$ with learnable parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ models the dynamics as,

$$\frac{d\tilde{\boldsymbol{u}}}{dt} = f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{u}}). \tag{2}$$

Neural ODEs satisfy the differential-equation structure of the ground truth a priori and can be augmented with further prior knowledge, such as conserved quantities (Greydanus et al., 2019; Cranmer et al., 2020) or partially known equations (Rackauckas et al., 2021).

Given a training dataset consisting of n trajectories sampled at discrete measurement times $\{t_j\}_{j=0}^m$, such that $\mathcal{D}_{train} = \{\{\boldsymbol{u}_j^{(i)}\}_{j=0}^m\}_{i=1}^n$, Neural ODE training is typically formulated as solving the following unconstrained optimization problem:

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{traj}(\boldsymbol{\theta}); \quad \mathcal{L}_{traj}(\boldsymbol{\theta}) = \frac{1}{S \cdot |\mathcal{D}_{train}|} \sum_{(i,j)\in\mathcal{D}_{train}} \sum_{s=1}^{S} ||\phi_{s\Delta t;\boldsymbol{\theta}}(\boldsymbol{u}_j^i) - \boldsymbol{u}_{j+s}^i||_2^2, \tag{3}$$

where $\phi_{T;\boldsymbol{\theta}}$ is the time-T map of the Neural ODE, calculated by integrating Eq. (2) with a differentiable numerical ODE solver, and $\phi_{\Delta t}(\boldsymbol{u}_j^i) = \boldsymbol{u}_{j+1}^i$. In practice, $\mathcal{L}_{traj}$ is minimized by mini-batch gradient descent (Goodfellow et al., 2016) over randomly sampled length-S trajectory segments.

Statistically accurate learning methods for chaotic dynamics attempt to regularize the unstable learning problem of Eq. (3) by solving the constrained problem

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{traj}(\boldsymbol{\theta}) \quad s.t. \quad \mu_{\boldsymbol{\theta}} = \mu, \tag{4}$$

where $\mu_{\boldsymbol{\theta}}$ denotes the physical measure of the Neural ODE dynamics. Since the learned dynamics are not guaranteed to admit such a measure, Eq. (4) is to be read as the ideal objective: the learned flow should preserve μ whenever it admits an invariant measure. This constraint is difficult to enforce strictly in practice, so Eq. (4) is often relaxed into the following Lagrange multipliers form:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}); \quad \mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{traj}(\boldsymbol{\theta}) + \lambda \cdot \ell(\mu_{\boldsymbol{\theta}}, \mu), \tag{5}$$

where ℓ is a pairwise loss between measures. However, losses pertaining to the ground truth measure are intractable to compute directly, and one instead compares observables or statistics induced by the measures, such as Lyapunov exponents (Platt et al., 2023), domain-specific summary statistics (Jiang et al., 2023), or distances between empirical trajectory distributions in phase space, e.g., the maximum mean discrepancy (MMD) (Schiff et al., 2024).

4 Neighborhood-based training of neural ODEs

Concretely, while Eq. (5) enforces agreement between invariant measures, the local features of how the flow transports and deforms sets can be lost. We therefore shift from a global to a local perspective and study how the pushforward measure distributes over the images of the sets forming a cover of the attractor, i.e., we solve

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{traj}(\boldsymbol{\theta}) \quad s.t. \quad (\phi_{\Delta t;\boldsymbol{\theta}})_{\#}\, \mu|_{\phi_{\Delta t}(U_i)} = (\phi_{\Delta t})_{\#}\, \mu|_{\phi_{\Delta t}(U_i)}\,, \ \forall i, \tag{6}$$

where $\{U_i\}_{i=1}^P$ is a finite cover of the chaotic attractor $\mathcal{A}$ by Borel measurable sets, and $(\phi_{\Delta t})_{\#}\mu|_{\phi_{\Delta t}(U_i)}$, $(\phi_{\Delta t;\boldsymbol{\theta}})_{\#}\mu|_{\phi_{\Delta t}(U_i)}$ denote the restrictions of the pushforward measures to $\phi_{\Delta t}(U_i)$ ($\{\phi_{\Delta t}(U_i)\}_{i=1}^P$ is also a cover of $\mathcal{A}$, since $\phi_{\Delta t}$ is a diffeomorphism and $\mathcal{A}$ is invariant). The constraint requires the Neural ODE to reproduce how the ground truth transports μ restricted to each neighborhood over time Δt.

At the same time, matching the pushforward measures on every set of a cover pins down the global measure (see proof in Appendix E):

Proposition 1. *Let ν_1, ν_2 be finite measures on a measurable space $(X, \mathcal{C})$, both supported on a set $\mathcal{A} \in \mathcal{C}$, and let $\{B_j\}_{j\in\mathbb{N}} \subset \mathcal{C}$ be a countable cover of $\mathcal{A}$. If $\nu_1|_{B_j} = \nu_2|_{B_j}$ for all j, then $\nu_1 = \nu_2$ on $\mathcal{C}$.*

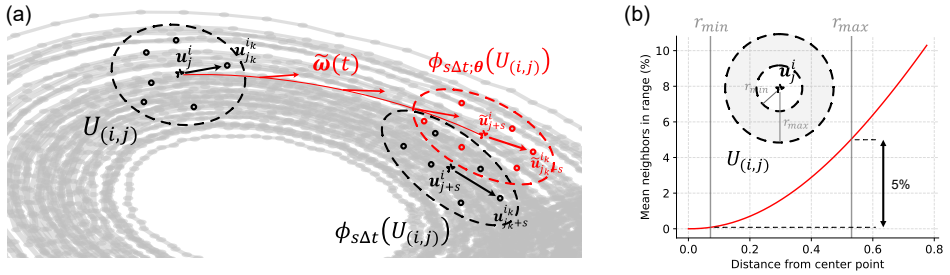

Figure 1: Schematic of our Neighborhood method. **(a)** We focus on finite cover of the attractor $U_{(i,j)}$ and regularize training so that the model deforms the cover in the same manner as the ground truth dynamics. **(b)** We design our cover sets to be annuli to mitigate the effects of noise.

With $\nu_1 = (\phi_{\Delta t})_{\#}\mu$, $\nu_2 = (\phi_{\Delta t;\boldsymbol{\theta}})_{\#}\mu$ and $B_j = \phi_{\Delta t}(U_j)$, and since μ is invariant under the ground-truth dynamics, Proposition 1 implies $(\phi_{\Delta t;\boldsymbol{\theta}})_{\#}\mu = (\phi_{\Delta t})_{\#}\ \mu = \mu$. Therefore, the Neural ODE preserves μ under iterates of its time-Δt map and, for small Δt, approximately in continuous time as well.

Turning Eq. (6) into an algorithm requires three ingredients: (i) a construction of a finite cover $\{U_i\}$ of the attractor from data, together with estimates of the ground-truth pushforward measures on the images $\phi_{\Delta t}(U_i)$; (ii) a procedure for evolving the cover sets under the Neural ODE and estimating the corresponding pushforward measures on $\phi_{\Delta t;\boldsymbol{\theta}}(U_i)$; and (iii) a pairwise loss ℓ quantifying the discrepancy between the two pushforward measures on each evolved neighborhood. Note that (ii) and (i) produce two different evolved covers, $\{\phi_{\Delta t;\boldsymbol{\theta}}(U_i)\}$ and $\{\phi_{\Delta t}(U_i)\}$, which is at first sight inconsistent with Proposition 1. However, ℓ is chosen so that $\ell = 0$ implies $\phi_{\Delta t}(U_i) = \phi_{\Delta t;\boldsymbol{\theta}}(U_i)$; whenever the loss is small, the two evolved covers nearly coincide and the restricted pushforward measures are compared on (approximately) the same sets. We detail each ingredient below.

Estimating the time-evolution of attractor covers from data Since our dataset consists of discrete trajectory measurements on the attractor, we construct a cover $\{U_{(i,j)}\}_{i=1,j=0}^{n,m}$ from neighborhoods centered at the data points $\boldsymbol{u}_j^i$. Assuming the trajectories sample the asymptotic dynamics on the attractor, the neighboring points contained in $U_{(i,j)}$ are samples from the physical measure restricted to that neighborhood, $\{\boldsymbol{u}_{j_k}^{i_k}\} \sim \mu|_{U_{(i,j)}}$, where (i_k, j_k) are the trajectory and time indices of the k-th neighbor, and their forward evolution provides samples of the evolved neighborhood, $\{\boldsymbol{u}_{j_k+s}^{i_k}\} \sim (\phi_{s\Delta t})_{\#}\ \mu|_{\phi_{s\Delta t}(U_{(i,j)})}$.

While a simple choice for the neighborhood around $\boldsymbol{u}_j^i$ would be an open ball of radius r_{max}, in real-world applications noise obscures the time evolution of neighbors that are too close to the center point: additive noise of strength ϵ perturbs a to-neighbor displacement by $\sim 2\epsilon$. To counter this, we modify the neighborhood to be an annulus with inner and outer radii r_{min} and r_{max} (Fig. 1b). We set $r_{min} = c\,\tilde{\epsilon}$, where $\tilde{\epsilon}$ is the noise standard deviation *estimated from the data* by detrending each noisy trajectory with a smoothing cubic spline whose smoothing strength is selected by generalized cross-validation (Virtanen et al., 2020), and $c = 8$. For r_{max}, we find that each neighborhood needs to contain some fraction of the total number of points; otherwise the learned local information overfits on the few neighbors watched during training. We therefore select r_{max} so that on average each annulus contains $p = 5\%$ of the points in the dataset. Ablations over c and p (Appendix F) show that performance is stable with respect to both radii, degrading only at high noise when c is small (displacements swamped by noise) or p is large (truncation error of the expansion below), and even then the Jacobian accuracy remains comparable to or better than DySLIM's.

Neighbors are identified once per dataset, prior to training, using KDTrees (Maneewongvatana & Mount, 1999), with negligible impact on training and evaluation.

Evolving phase space neighborhoods with Taylor expansion of the Neural ODE Instead of directly evolving the neighboring points $\phi_{\Delta t;\boldsymbol{\theta}}(\boldsymbol{u}_{j_k}^{i_k})$, we evolve perturbations relative to the center point $\boldsymbol{u}_j^i$ and reconstruct the evolved neighborhood from them (Fig. 1a), to capture how the Neural

ODE propagates small perturbations; constraining the perturbations relative to a common center trajectory also enforces coherent local deformation of nearby points, preventing spurious local distortions of the vector field. Concretely, the neighborhoods are evolved through the Taylor expansion

$$\frac{d}{dt}\begin{bmatrix}\tilde{\boldsymbol{u}}\\ \tilde{\boldsymbol{\omega}}^k\end{bmatrix} = \begin{bmatrix} f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{u}}) \\ \partial f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}}\,[\tilde{\boldsymbol{\omega}}^k] + \frac{1}{2}\partial^2 f_{\boldsymbol{\theta}}\big|_{\tilde{\boldsymbol{u}}}\,[\tilde{\boldsymbol{\omega}}^k, \tilde{\boldsymbol{\omega}}^k]\end{bmatrix}, \quad \begin{bmatrix}\tilde{\boldsymbol{u}}(0) = \boldsymbol{u}_j^i \\ \tilde{\boldsymbol{\omega}}^k(0) = \boldsymbol{u}_{j_k}^{i_k} - \boldsymbol{u}_j^i,\end{bmatrix} \tag{7}$$

where $\tilde{\boldsymbol{\omega}}^k$ is the time-evolving perturbation vector between the center $\boldsymbol{u}_j^i$ and its k-th neighbor $\boldsymbol{u}_{j_k}^{i_k}$ and the second order term accounts for the fact that these perturbations are not infinitesimal.

From this, samples from the Neural ODE-evolved cover $U_{(i,j)}$ can be obtained as: $\{\tilde{\boldsymbol{u}}_{j_k+s}^{i_k}\} \sim (\phi_{s\Delta t;\boldsymbol{\theta}})_{\#}\,\mu|_{\phi_{s\Delta t;\boldsymbol{\theta}}(U_{(i,j)})}$, where $\tilde{\boldsymbol{u}}_{j_k+s}^{i_k} = \tilde{\boldsymbol{u}}_j^i + \tilde{\boldsymbol{\omega}}^k$.

Why not rely on the trajectory loss alone? Since the model is trained with many neighboring trajectories, one may ask why the evolution of perturbations must explicitly be incorporated into the loss. Eq. (7) makes explicit that perturbations are propagated by the Jacobian (and Hessian) of the vector field, whereas Eq. (3) constrains *states*: if training achieves $\|\tilde{\boldsymbol{u}}_i - \boldsymbol{u}_i\| < \epsilon$ for all states, the separation $\boldsymbol{\omega}^k(0) = \boldsymbol{u}_{j_k}^{i_k} - \boldsymbol{u}_j^i$ between a center point and its k-th neighbor is only constrained to $\|\tilde{\boldsymbol{\omega}}^k - \boldsymbol{\omega}^k\| \leq 2\epsilon$. As $\|\boldsymbol{\omega}^k\| \ll \|\boldsymbol{u}_j^i\|$, an ϵ that gives satisfactory trajectories leaves the *relative* error of the separations, and hence the Jacobian, essentially unconstrained. Moreover, Eq. (3) sums independent per-trajectory terms, so incorrect relative geometry between nearby trajectories is never penalized directly. Finally, with finite sampling, a network can fit the values at the measurement points while taking arbitrary derivatives in between (Latrémolière et al., 2022). The same observation underlies Sobolev training (Czarnecki et al., 2017; Cho et al., 2025), where supervising derivatives alongside values improves both; Appendix G.1 shows an oracle supplying the true Jacobians indeed improves every aspect of a trained Neural ODE. Since neither the vector field nor its Jacobian is ever observed in our setting, the loss introduced next is our route to the same effect from the trajectory data at hand: it compares predicted and observed separations at their own scale and couples nearby trajectories within each neighborhood.

Computational cost Eq. (7) requires only the directional derivatives $\partial f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}}[\tilde{\boldsymbol{\omega}}^k],\ \partial^2 f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}}[\tilde{\boldsymbol{\omega}}^k, \tilde{\boldsymbol{\omega}}^k] \in \mathbb{R}^d$, never the full Hessian $\partial^2 f_{\boldsymbol{\theta}} \in \mathbb{R}^{d\times d\times d}$. We compute both with a single nested Jacobian-vector product (forward-over-forward differentiation; Nzoyem et al., 2025, Eq. (30)). Since forward-mode differentiation multiplies the time and memory of a primal evaluation by constants (≈ 3 and 2) independent of d (Griewank & Walther, 2008), a neighbor point costs at most $\approx 9\times$ / $4\times$ the time / memory of a center point, and the cost of our method relative to Vanilla or DySLIM at fixed batch size is bounded by a constant independent of the system dimension (Appendix H). Empirically, the relative FLOPs / peak memory are $3.2\times/1.6\times$ at $d = 2$ and decrease towards 1 with d (Table 9), and training wall times are indistinguishable from the baselines on all systems studied (Table 10).

Matching neighborhood predictions with the maximum mean discrepancy To measure the discrepancy between the Neural ODE and ground truth versions of the time-evolved local neighborhoods using point samples, we use the squared maximum mean discrepancy (MMD^2) (Gretton et al., 2012):

$$\begin{aligned}\ell\left((\phi_{s\Delta t;\boldsymbol{\theta}})_{\#}\,\mu|_{\phi_{\Delta t;\boldsymbol{\theta}}(U_{(i,j)})}, (\phi_{s\Delta t})_{\#}\,\mu|_{\phi_{\Delta t}(U_{(i,j)})}\right) &= \widehat{\text{MMD}^2}(\{\boldsymbol{u}_{j_k+s}^{i_k}\}, \{\tilde{\boldsymbol{u}}_{j_k+s}^{i_k}\}) \\ = \frac{1}{K(K-1)}\sum_{k=1}^{K}\sum_{k'\neq k}^{K}\left[\kappa\left(\tilde{\boldsymbol{u}}_{j_k}^{i_k}, \tilde{\boldsymbol{u}}_{j_{k'}}^{i_{k'}}\right) + \kappa\left(\boldsymbol{u}_{j_k}^{i_k}, \boldsymbol{u}_{j_{k'}}^{i_{k'}}\right)\right] &- \frac{2}{K^2}\sum_{k=1}^{K}\sum_{k'=1}^{K}\kappa\left(\tilde{\boldsymbol{u}}_{j_k}^{i_k}, \boldsymbol{u}_{j_{k'}}^{i_{k'}}\right)\end{aligned} \tag{8}$$

where κ is a suitable **characteristic** kernel function (see Appendix D for further details). Because κ is characteristic, $\widehat{\text{MMD}^2} \approx 0$ implies that the pushforward measures, and hence the neighborhood images on which the samples are supported, approximately agree - recovering the setting of Proposition 1. Comparing evolved neighborhoods as distributions, rather than point-by-point, is what drives the local deformation of each neighborhood, such as expansion, contraction and shearing, to agree with that of the underlying flow.

With Eq. (8), we can formulate our loss function, corresponding to a relaxed version of Eq. (6):

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{traj}(\boldsymbol{\theta}) + \lambda \cdot \frac{1}{S \cdot |\mathcal{D}_{train}|} \sum_{(i,j)\in\mathcal{D}_{train}} \sum_{s=1}^{S} \ell\left((\phi_{s\Delta t;\boldsymbol{\theta}})_{\#}\, \mu|_{U_{(i,j)}}, (\phi_{s\Delta t})_{\#}\, \mu|_{U_{(i,j)}} \right), \quad (9)$$

where λ is a hyperparameter that balances the relative influence of the two losses.

Analogously to Eq. (3), our loss function is also minimized using mini-batch gradient descent. For this, we set a fixed batch size $|\mathcal{B}|$, and sample B' center points and K neighbor points per cover set such that $|\mathcal{B}| = B'(K+1)$. We set the number of neighbors K as a hyperparameter of our method, along with the neighborhood loss weight λ.

5 Experiments

5.1 Datasets

We performed experiments on three different chaotic systems with varying dimensionalities: the 3D Lorenz 63 system (Lorenz, 1963), the 4D Hyperchaotic Chen system (Li et al., 2005), and the 6D Lorenz 96 system (Lorenz, 1996). The sampling interval was set to 0.01τ, where the characteristic timescale τ of each system was estimated from the averaged Fourier spectrum of long trajectories. For each system, the training, validation, and test data consisted of 50 trajectories of 1000 time points, standardized to zero mean and unit variance along each dimension. The training and validation data were then corrupted with Gaussian noise of standard deviation 0.01, 0.05 or 0.1, i.e., 1%, 5%, or 10% of the data standard deviation. See Appendix B for details.

5.2 Baselines

To test the efficacy of our proposed **Neighborhood** method, we perform experiments against the following two baselines: (i) **Vanilla**: which corresponds to the standard trajectory-based Neural ODE training (Eq. (3)), and (ii) **DySLIM** (Schiff et al., 2024): which adds two loss terms estimating the global discrepancy between the invariant measures of data and model. Hyperparameters of DySLIM and our Neighborhood method[2] were determined via a small grid search prior to the main experiments, which we detail in Appendix C. In Appendix G, we additionally compare against a Jacobian-oracle regularizer (Park et al., 2025) and a classical local Jacobian estimator (Eckmann et al., 1986; Abarbanel et al., 1992).

5.3 Model training and evaluation

The Neural ODE vector field $f_{\boldsymbol{\theta}}$ is a multi-layer perceptron with layer widths [d, 64, 64, d]. Time-Δt maps were computed with the adaptive 5th-order Runge–Kutta solver `Tsit5` (Tsitouras, 2011) of `diffrax` (Kidger, 2022) (rel./abs. tolerences: 10^{-4}/10^{-6}). We use ten-step rollouts and the AdaBelief optimizer (Zhuang et al., 2020) with batch size 2048 and learning rate 2e-3, for 10000 training steps on Lorenz96 and 5000 otherwise, keeping the model with the lowest validation loss. All computations were performed on NVIDIA V100 GPUs (32GB). Each experiment was repeated with three random seeds for weight initialization; we report means and standard deviations across seeds.

6 Results

6.1 Short-term trajectory predictions

As only long-term trajectories are impossible to predict exactly, properly trained chaotic models should still be able to generate accurate short-term forecasts. Therefore, we first check the short-term prediction accuracy of the trained models using the valid prediction time (VPT), which is the time (scaled by the maximum Lyapunov exponent λ_1) elapsed until the normalized root mean squared error (NRMSE) between the data and the model trajectory exceeds a prespecified threshold value (Fig. 2a, Appendix D).

[2]Both methods have two hyperparameters each.

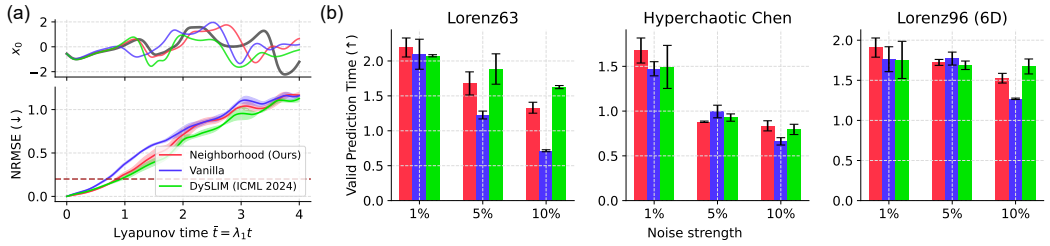

Figure 2: Short-term accuracy of the trained models. **(a)** Data (gray line) and model predictions (trained on 10% noise data) for the first state variable of the 6D Lorenz 96 system (upper panel) and the corresponding NRMSE as a function of Lyapunov time (lower panel). Dashed brown line marks the VPT threshold value (0.3) used in (b). **(b)** Valid prediction time (VPT) across different systems and training data noise strengths.

From the results in Fig. 2b, we see that for lower noise strengths, conventional trajectory-based training (Vanilla) performs competitively, indicating that regularizing the MSE objective is not necessary for short-term predictions if the data is not too degraded. As noise increases, all three methods lose short-term accuracy as the dynamics information in the training data becomes increasingly blurred, but the VPTs of Vanilla-trained models drop much faster than those of DySLIM and our Neighborhood method. This indicates that our neighborhood loss term (as well as the two additional loss terms of DySLIM) properly regularizes the learning problem, preventing overfitting to the noisy short-term trajectories.

6.2 Long-term statistical accuracy

Next, we evaluate the long-term statistical accuracy of the differently trained models by quantifying how well the learned dynamics preserve the chaotic attractor over long periods of time. To do so, we take a sample of 5000 points on the data attractor and evolve them in time with both the ground truth equations and the learned Neural ODEs. The resulting long-time evolution of the sampled attractors is then compared visually (Fig. 3a) and quantitatively, using the squared maximum mean discrepancy (MMD^2; Eq. (8)) (Fig. 3b, c).

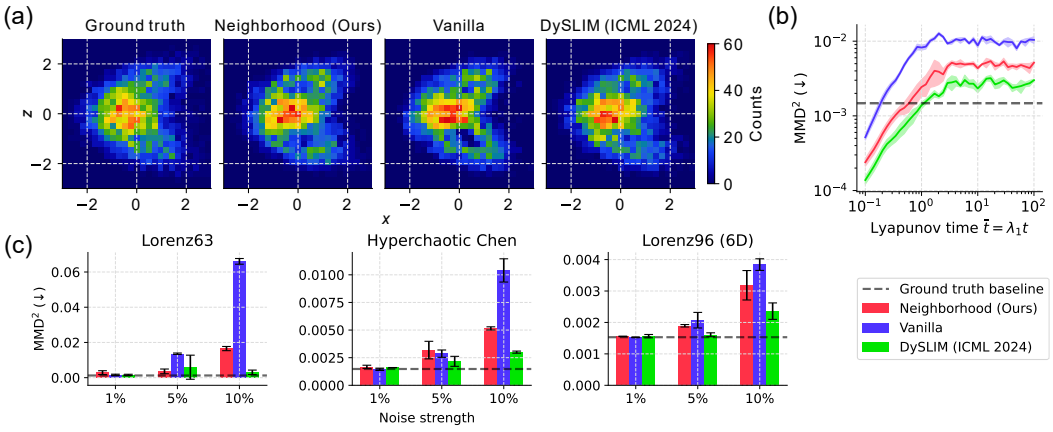

Figure 3: Long-term statistical accuracy of the trained models. **(a)** 2D histograms of the time-evolved samples of the hyperchaotic Chen system, at Lyapunov time $\bar{t} = 100$. The models depicted were trained on the 10% noise data. **(b)** The corresponding MMD^2 as a function of rollout time. The gray dashed line indicates the MMD^2 between two independent samples of the same data attractor (see Appendix D for details). **(c)** MMD^2 between the ground truth and model-evolved samples at $\bar{t} = 100$ for different systems and noise strengths.

As shown in Fig. 3b, the MMD^2 starts from zero, rises as the ground-truth- and model-evolved points decorrelate, and plateaus once the points settle onto the model's invariant measure, which

may differ from that of the data (Fig. 3a). The plateau values in Fig. 3c paint a similar picture as the short-term results: all methods learn the ground-truth attractor when only a small amount of noise is present, whereas for larger noise Vanilla's invariant measure becomes increasingly inaccurate while Neighborhood and DySLIM mitigate the effect of noise. The same ranking of the methods is obtained with the Sinkhorn divergence as an alternative long-term metric (Appendix I).

While our method performs competitively with DySLIM in short-term prediction, DySLIM achieves slightly higher long-term statistical accuracy, likely due to its direct optimization of physical measure agreement in the long term. In contrast, our method matches local covers and their pushforward measures, thereby indirectly capturing both statistical and local dynamical properties simultaneously. As shown next, this tradeoff yields substantially improved local dynamical behavior while maintaining competitive statistical performance.

6.3 Local behavior of the learned dynamics

Finally, we address the study's key motivation and assess how well the trained models capture the local behavior of the ground-truth dynamics. We evaluate the models on two fronts: (i) the relative vector field error $\|f_{\boldsymbol{\theta}} - f\|_2 / |f\|_2$, which gauges how well the trained models reproduce the ground truth velocities, and (ii) the relative Jacobian error $\|\partial f_{\boldsymbol{\theta}} - \partial f\|_F / |\partial f\|_F$ as it is the Jacobian of the dynamics that govern the local deformation of the chaotic attractor via the variational equation.

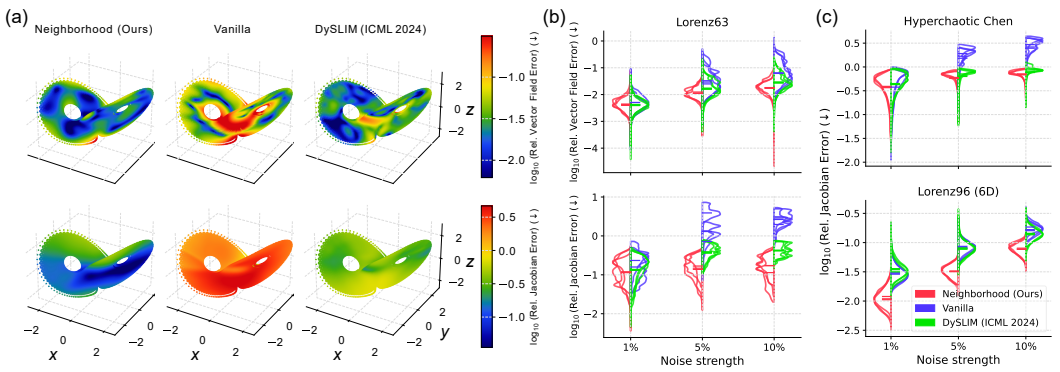

Figure 4: Local behavior of the trained models. **(a)** Relative error in the vector field (upper row) and the Jacobians (lower row) for models trained on the 3D Lorenz 63 system with 10% noise. **(b)** Violin plots of the relative vector field and Jacobian errors for the Lorenz 63 system, across all noise strengths. **(c)** Relative Jacobian error with varying noise strengths for different systems.

Fig. 4a visualizes these two metrics on the attractor of the Lorenz63 system, from which we immediately see the merits of "watching the neighbors" during training. In particular, we find that while the Vanilla method learns a moderately accurate vector field over most of the attractor, the corresponding Jacobians are completely erroneous, highlighting the fact that learning to reproduce the values of a target function does not guarantee accurate estimation of its derivatives (He et al., 2019; Latrémolière et al., 2022; Eisen et al., 2025). Although DySLIM produces more accurate vector fields and Jacobians than Vanilla, our Neighborhood method consistently outperforms DySLIM across all experimental settings (Fig. 4b, c). The vector field errors for the other two systems (Table 7) show the same pattern: comparable to the baselines at 1% noise and consistently better at 5% and 10%, with the margin widening with noise, as the Sobolev-training analogy of Section 4 predicts.

We particularly highlight the region near the origin of the Lorenz63 attractor, where both Vanilla and DySLIM fail to reconstruct the vector field (Fig. 4a, upper row; red patches). This region hosts the lobe-switching mechanism –the trajectory deciding between the two wings– and is therefore highly unpredictable (Dong et al., 2025; Ayers et al., 2023); remarkably, the Neighborhood method largely overcomes this difficulty, producing accurate and spatially uniform predictions across the entire attractor.

Lyapunov exponents As an application of accurate Jacobian learning, we assessed the model predictions of the Lyapunov exponents, which is a key descriptor of chaotic dynamics. Due to the

page constraint, we present the results in Appendix A, where we show that our model also yields greatly improved values for these exponents, including the most negative exponents, which are the hardest to recover from data.

7 Related Work

Estimating the Jacobian from time series data. Due to its role in Lyapunov exponent estimation and chaos detection, estimating the Jacobian from time-series data has been extensively studied, with approaches broadly classified as global or local (McCaffrey et al., 1992; Escot & Sandubete, 2023). Global methods fit a surrogate model –polynomials (Ataei et al., 2003), radial basis functions (Potts & Broomhead, 1991; McCaffrey et al., 1992; Holzfuss & Parlitz, 2006; Sun & Wu, 2012), or shallow neural networks (McCaffrey et al., 1992; Gencay & Dechert, 1992; Nychka et al., 1992; Shintani & Linton, 2004)– to the time series and differentiate it; Vanilla Neural ODE training is a global method in this sense, and inherits the problem that accurate values do not imply accurate derivatives (He et al., 2019; Latrémolière et al., 2022). Local methods (Eckmann et al., 1986; Brown et al., 1991; Abarbanel et al., 1992) instead fit an independent regression model –linear, quadratic (Brown et al., 1991; Abarbanel et al., 1992), or kernel-based (Escot & Sandubete, 2023)– to the evolution of neighbor displacements around each point and read off the Jacobian from its linear term. They condition directly on Jacobian information, but since nothing ties the models fitted at nearby points to one another, they are known to be highly sensitive to noise (Yang & Wu, 2011; Yao et al., 2013). In Appendix G.2 we compare against the local quadratic estimator of Eckmann et al. (1986); Abarbanel et al. (1992), the direct analogue of our second-order expansion: it is the most accurate method on noiseless data, yet falls behind even Vanilla at 1% noise, whereas our method is the most accurate at every noise level. Our Neighborhood method thus combines the strengths of both families: a single global model provides a smooth vector-field prior that keeps Jacobians consistent across phase space, while the neighborhood loss conditions it explicitly on local displacement information.

Regularizing chaotic dynamics learning with invariants. Beyond DySLIM (Schiff et al., 2024), training has been regularized with Lyapunov exponents (Platt et al., 2023) or summary statistics (Jiang et al., 2023), and chaotic dynamics have been learned with recurrent architectures (Mikhaeil et al., 2022; Hess et al., 2023). Closest to our work, Park et al. (2025) supervise a Neural ODE with ground-truth Jacobians, which Appendix G.1 reproduces as an idealized reference.

8 Limitations

Our method requires data dense enough that each state has neighbors inside its annulus; for sparsely sampled or high-dimensional systems the local information degrades and the (one-off) neighbor search becomes more demanding. While our ablations indicate robustness, the hyperparameters (K, λ, c, p) require selection. The computational overhead is a dimension-independent constant, but not zero. Finally, we consider deterministic dynamics with observation noise on synthetic benchmarks, where ground-truth Jacobians are available for evaluation; process noise, whose invariant measures obey a Fokker–Planck equation rather than the deterministic flow, and real-world data are left for future work.

9 Conclusion

We introduced the Neighborhood training method for Neural ODEs, a noise-robust approach that jointly achieves long-term statistical accuracy and accurate local dynamics by analyzing the evolution of local covering sets of the attractor. Experiments show substantially improved local dynamical accuracy while maintaining competitive statistical performance, at a cost comparable to existing methods. Future directions include extensions to higher-dimensional and partially observed systems, to discrete-time architectures such as state-space models and recurrent networks (Appendix J), and covering constructions that account for variations in local predictability across the attractor (Abarbanel et al., 1991; Bailey & Nychka, 1997; Ziehmann et al., 2000). We hope this work provides a complementary starting point for developing more sophisticated methods for learning chaotic dynamics from data.

AI USE STATEMENT

We did not use generative AI tools for any task requiring disclosure under the ICLR 2027 AI Policy for Authors: no AI tools were used to develop the theoretical framework, formulate or prove the mathematical claims, propose or refine hypotheses, design the methodology or experiments, implement methods, generate or process data, or interpret the results. Among the tasks with recommended disclosure, we used generative AI tools (large language models) for two purposes. First, to edit the manuscript for readability, specifically to polish wording and trim excess words in drafts that were first written by the authors; all AI-assisted edits were reviewed by the authors, and the technical content, claims, and results were not altered by them. Second, to augment the literature review: after an initial survey conducted by the authors, AI tools were used as a second pass to identify potentially relevant related work, and every suggested reference was read and verified by the authors before being cited.

REPRODUCIBILITY STATEMENT

We have taken the following steps to make our results reproducible. All datasets are synthetic and can be regenerated exactly: the governing equations, parameter values, initial-condition distributions, burn-in times, characteristic-timescale estimation, normalization, and the noise model are specified in Section 5 and Appendix B, including the procedure used to estimate the noise level from data. The model architecture, ODE solver and tolerances, optimizer, batch size, learning rate, training budget, and checkpoint selection are stated in Section 5; the hyperparameter grids and selection protocol for both DySLIM and our Neighborhood method are given in Appendix C. All evaluation metrics (VPT, MMD^2, relative vector field and Jacobian errors, Lyapunov exponents, and the Sinkhorn divergence) are defined precisely in Appendices D and I, and the complete proof of Proposition 1 is given in Appendix E. The additional baselines, ablations, and cost measurements are fully specified in Appendices F to H, and all reported numbers are means and standard deviations over three seeds. Finally, the full source code –dataset generation, training of all methods including the baselines, and evaluation– is provided at `https://anonymous.4open.science/r/neighborwatch`.

REFERENCES

H. D. I. Abarbanel, R. Brown, and M. B. Kennel. Variation of Lyapunov exponents on a strange attractor. *J Nonlinear Sci*, 1(2):175–199, 1991. ISSN 0938-8974, 1432-1467. doi: 10.1007/BF01209065.

H. D. I. Abarbanel, R. Brown, and M. B. Kennel. Local Lyapunov exponents computed from observed data. *J Nonlinear Sci*, 2(3):343–365, 1992. ISSN 0938-8974, 1432-1467. doi: 10.1007/BF01208929.

Kathleen T. Alligood, Tim Sauer, and James A. Yorke. *Chaos: An Introduction to Dynamical Systems*, chapter 2.1 Mathematical Models. Textbooks in Mathematical Sciences. Springer, New York, 1996. ISBN 978-0-387-94677-1.

M. Ataei, A. Khaki-Sedigh, B. Lohmann, and C. Lucas. Estimating the Lyapunov exponents of chaotic time series: A model based method. In *2003 European Control Conference (ECC)*, pp. 3106–3111, 2003. doi: 10.23919/ECC.2003.7086516.

Daniel Ayers, Jack Lau, Javier Amezcua, Alberto Carrassi, and Varun Ojha. Supervised machine learning to estimate instabilities in chaotic systems: Estimation of local Lyapunov exponents. *Quarterly Journal of the Royal Meteorological Society*, 149(753):1236–1262, 2023. ISSN 1477-870X. doi: 10.1002/qj.4450.

B. Bailey and Douglas W. Nychka. Local Lyapunov exponents: Predictability depends on where you are. *Nonlinear Dynamics and Economics, Kirman et al. Eds*, 1997.

Reggie Brown, Paul Bryant, and Henry D. I. Abarbanel. Computing the Lyapunov spectrum of a dynamical system from an observed time series. *Phys. Rev. A*, 43(6):2787–2806, 1991. doi: 10.1103/PhysRevA.43.2787.

Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

Guanrong Chen and Tetsushi Ueta. Yet another chaotic attractor. *Int. J. Bifurcation Chaos*, 09(07): 1465–1466, 1999. ISSN 0218-1274. doi: 10.1142/S0218127499001024.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Namkyeong Cho, Junseung Ryu, and Hyung Ju Hwang. Sobolev training for operator learning. *J. Comput. Phys.*, 543:114408, 2025. ISSN 0021-9991. doi: 10.1016/j.jcp.2025.114408.

F Christiansen and H H Rugh. Computing Lyapunov spectra with continuous Gram - Schmidt orthonormalization. *Nonlinearity*, 10(5):1063–1072, 1997. ISSN 0951-7715, 1361-6544. doi: 10.1088/0951-7715/10/5/004.

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

Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Marco Cuturi, Laetitia Meng-Papaxanthos, Yingtao Tian, Charlotte Bunne, Geoff Davis, and Olivier Teboul. Optimal Transport Tools (OTT): A JAX Toolbox for all things Wasserstein, 2022.

Wojciech M. Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev Training for Neural Networks. In *Adv. Neural Inf. Process. Syst.*, volume 30. Curran Associates, Inc., 2017.

Chenyu Dong, Davide Faranda, Adriano Gualandi, Valerio Lucarini, and Gianmarco Mengaldo. Time-lagged recurrence: A data-driven method to estimate the predictability of dynamical systems. *Proceedings of the National Academy of Sciences*, 122(20):e2420252122, 2025. doi: 10.1073/pnas.2420252122.

J. P. Eckmann, S. Oliffson Kamphorst, D. Ruelle, and S. Ciliberto. Liapunov exponents from time series. *Phys. Rev. A*, 34(6):4971–4979, 1986. ISSN 0556-2791. doi: 10.1103/PhysRevA.34.4971.

Adam Joseph Eisen, Mitchell Ostrow, Sarthak Chandra, Leo Kozachkov, Earl K. Miller, and Ila R. Fiete. Characterizing control between interacting subsystems with deep Jacobian estimation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Lorenzo Escot and Julio E. Sandubete. Estimating Lyapunov exponents on a noisy environment by global and local Jacobian indirect algorithms. *Applied Mathematics and Computation*, 436: 127498, 2023. ISSN 0096-3003. doi: 10.1016/j.amc.2022.127498.

Jan Frøyland and Knut H. Alfsen. Lyapunov-exponent spectra for the Lorenz model. *Phys. Rev. A*, 29(5):2928–2931, 1984. ISSN 0556-2791. doi: 10.1103/PhysRevA.29.2928.

Ramazan Gencay and W. Davis Dechert. An algorithm for the n Lyapunov exponents of an n-dimensional unknown dynamical system. *Physica D: Nonlinear Phenomena*, 59(1):142–157, 1992. ISSN 0167-2789. doi: 10.1016/0167-2789(92)90210-E.

Aude Genevay, Gabriel Peyre, and Marco Cuturi. Learning Generative Models with Sinkhorn Divergences. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pp. 1608–1617. PMLR, 2018.

Robert Gilmore and Marc Lefranc. *The Topology of Chaos: Alice in Stretch and Squeezeland*. Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany, second revised and enlarged edition edition, 2011. ISBN 978-3-527-41067-5.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press Cambridge, 2016.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. ISSN 1533-7928.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian Neural Networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, 2. ed edition, 2008. ISBN 978-0-89871-659-7.

John Guckenheimer and Philip Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, volume 42 of *Applied Mathematical Sciences*, chapter Statistical Properties: Dimension, Entropy and Liapunov Exponents. Springer New York, New York, NY, 1983. ISBN 978-1-4612-7020-1 978-1-4612-1140-2. doi: 10.1007/978-1-4612-1140-2.

Xing He, Lei Chu, Robert Qiu, Qian Ai, and Wentao Huang. Data-driven Estimation of the Power Flow Jacobian Matrix in High Dimensional Space, 2019.

Florian Hess, Zahra Monfared, Manuel Brenner, and Daniel Durstewitz. Generalized Teacher Forcing for Learning Chaotic Dynamics. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 13017–13049. PMLR, 2023.

Joachim Holzfuss and Ulrich Parlitz. Lyapunov exponents from time series. In *Lyapunov Exponents: Proceedings of a Conference Held in Oberwolfach, May 28–June 2, 1990*, pp. 263–270. Springer, 2006.

JAX developers. JAX documentation: Just-in-time compilation. `https://docs.jax.dev/en/latest/jit-compilation.html`, 2024a.

JAX developers. JAX documentation: Forward- and reverse-mode autodiff. `https://docs.jax.dev/en/latest/jacobian-vector-products.html`, 2024b.

Ruoxi Jiang, Peter Y. Lu, Elena Orlova, and Rebecca Willett. Training neural operators to preserve invariant measures of chaotic attractors. In *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023.

Patrick Kidger. On Neural Differential Equations, 2022.

Suyong Kim, Weiqi Ji, Sili Deng, Yingbo Ma, and Christopher Rackauckas. Stiff neural ordinary differential equations. *Chaos*, 31(9):093122, 2021. ISSN 1054-1500, 1089-7682. doi: 10.1063/5.0060697.

Frédéric Latrémolière, Sadananda Narayanappa, and Petr Vojtěchovský. Estimating the Jacobian matrix of an unknown multivariate function from sample values by means of a neural network, 2022.

Xuyang Li, John Harlim, and Romit Maulik. A Weak Penalty Neural ODE for Learning Chaotic Dynamics from Noisy Time Series, 2025.

Yuxia Li, Wallace K. S. Tang, and Guanrong Chen. Generating hyperchaos via state feedback control. *Int. J. Bifurcation Chaos*, 15(10):3367–3375, 2005. ISSN 0218-1274. doi: 10.1142/S0218127405013988.

Zongyi Li, Miguel Liu-Schiaffini, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Learning Chaotic Dynamics in Dissipative Systems. In *Advances in Neural Information Processing Systems*, 2022.

Edward N. Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sci.*, 20(2):130–141, 1963. ISSN 0022-4928, 1520-0469. doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on Predictability*, volume 1, pp. 1–18. Reading, 1996.

Songrit Maneewongvatana and David M. Mount. Analysis of approximate nearest neighbor searching with clustered point sets, 1999.

Daniel F. McCaffrey, Ellner , Stephen, Gallant , A. Ronald, and Douglas W. and Nychka. Estimating the Lyapunov Exponent of a Chaotic System with Nonparametric Regression. *Journal of the American Statistical Association*, 87(419):682–695, 1992. ISSN 0162-1459. doi: 10.1080/01621459.1992.10475270.

Jonas Mikhaeil, Zahra Monfared, and Daniel Durstewitz. On the difficulty of learning chaotic dynamics with RNNs. In *Advances in Neural Information Processing Systems*, volume 35, pp. 11297–11312, 2022.

Joshua S. North, Christopher K. Wikle, and Erin M. Schliep. A Review of Data-Driven Discovery for Dynamic Systems. *International Statistical Review*, 91(3):464–492, 2023. ISSN 1751-5823. doi: 10.1111/insr.12554.

Douglas Nychka, Stephen Ellner, A. Ronald Gallant, and Daniel McCaffrey. Finding Chaos in Noisy Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 54(2):399–426, 1992. ISSN 0035-9246.

Roussel Desmond Nzoyem, David A. W. Barton, and Tom Deakin. Neural Context Flows for Meta-Learning of Dynamical Systems. In *The Thirteenth International Conference on Learning Representations*, 2025.

Valery Iustinovich Oseledec. A multiplicative ergodic theorem, Lyapunov characteristic numbers for dynamical systems. *Transactions of the Moscow Mathematical Society*, 19:197–231, 1968.

Jeongjin Park, Nicole Yang, and Nisha Chandramoorthy. When are dynamical systems learned from time series data statistically accurate? In *Advances in Neural Information Processing Systems*, volume 37, pp. 43975–44008, 2025.

Jason A. Platt, Stephen G. Penny, Timothy A. Smith, Tse-Chun Chen, and Henry D. I. Abarbanel. Constraining chaos: Enforcing dynamical invariants in the training of reservoir computers. *Chaos*, 33(10), 2023. ISSN 1054-1500. doi: 10.1063/5.0156999.

Michael A. S. Potts and David S. Broomhead. Time series prediction with a radial basis function neural network. In *Adaptive Signal Processing*, volume 1565, pp. 255–266. SPIE, 1991. doi: 10.1117/12.49782.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal Differential Equations for Scientific Machine Learning, 2021.

Yair Schiff, Zhong Yi Wan, Jeffrey B. Parker, Stephan Hoyer, Volodymyr Kuleshov, Fei Sha, and Leonardo Zepeda-Núñez. DySLIM: Dynamics stable learning by invariant measure for chaotic systems. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pp. 43649–43684, Vienna, Austria, 2024. JMLR.org.

Ippei Shimada and Tomomasa Nagashima. A Numerical Approach to Ergodic Problem of Dissipative Dynamical Systems. *Progress of Theoretical Physics*, 61(6):1605–1616, 1979. ISSN 0033-068X. doi: 10.1143/PTP.61.1605.

Mototsugu Shintani and Oliver Linton. Nonparametric neural network estimation of Lyapunov exponents and a direct test for chaos. *Journal of Econometrics*, 120(1):1–33, 2004. ISSN 0304-4076. doi: 10.1016/S0304-4076(03)00205-7.

Steven Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, chapter 1.2 The Importance of Being Nonlinear. CRC Press, Boca Raton, third edition edition, 2024.

Yuming Sun and Christine Qiong Wu. A radial-basis-function network-based method of estimating Lyapunov exponents from a scalar time series for analyzing nonlinear systems stability. *Nonlinear Dyn*, 70(2):1689–1708, 2012. ISSN 1573-269X. doi: 10.1007/s11071-012-0567-z.

Ch. Tsitouras. Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption. *Computers & Mathematics with Applications*, 62(2):770–775, 2011. ISSN 0898-1221. doi: 10.1016/j.camwa.2011.06.002.

Warwick Tucker. A Rigorous ODE Solver and Smale's 14th Problem. *Found. Comput. Math.*, 2(1): 53–117, 2002. ISSN 1615-3375. doi: 10.1007/s002080010018.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat Methods*, 17(3):261–272, 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2.

Caixia Yang and Christine Qiong Wu. A robust method on estimation of Lyapunov exponents from a noisy time series. *Nonlinear Dyn.*, 64(3):279–292, 2011. ISSN 1573-269X. doi: 10.1007/s11071-010-9860-x.

Tian-Liang Yao, Hai-Feng Liu, Jian-Liang Xu, and Wei-Feng Li. Lyapunov-exponent spectrum from noisy time series. *Int. J. Bifurc. Chaos*, 23(06):1350103, 2013. ISSN 0218-1274. doi: 10.1142/S0218127413501034.

Lai-Sang Young. What are srb measures, and which dynamical systems have them? *Journal of Statistical Physics*, 108(5-6):733–754, 2002. ISSN 1572-9613. doi: 10.1023/a:1019762724717. URL `http://dx.doi.org/10.1023/A:1019762724717`.

Rose Yu and Rui Wang. Learning dynamical systems from data: An introduction to physics-guided deep learning. *Proc. Natl. Acad. Sci. U.S.A.*, 121(27):e2311808121, 2024. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2311808121.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18795–18806. Curran Associates, Inc., 2020.

Christine Ziehmann, Leonard A. Smith, and Jürgen Kurths. Localized Lyapunov exponents and the prediction of predictability. *Physics Letters A*, 271(4):237–251, 2000. ISSN 0375-9601. doi: 10.1016/S0375-9601(00)00336-4.

A ADDITIONAL RESULTS

Lyapunov exponents One key application of accurate Jacobian learning is the estimation of the Lyapunov exponents from noisy trajectory data. This is because Lyapunov exponents characterize the average contraction and expansion rates on the chaotic attractor, serving as a key descriptor of chaotic systems. However, accurately inferring these quantities from surrogate models is challenging, as their computation requires both accurate long-term physical measures and local Jacobian information.

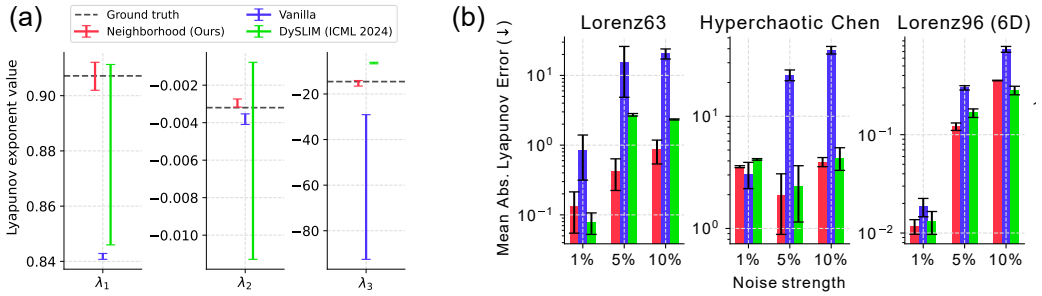

Figure 5: Lyapunov exponents of the trained models. **(a)** Predictions for the three Lyapunov exponents of the Lorenz63 system, from models trained on 10% noise data. **(b)** Mean absolute error of the computed Lyapunov exponents, across systems and noise levels.

As shown in Fig. 5b, we find that the vanilla method struggles to reproduce the Lyapunov exponents in the presence of larger noise. This is particularly pronounced for the negative exponent(s) (Fig. 5a; rightmost panel), which are known to be more difficult to recover from data, as chaotic attractors tend to be very "thin" in directions associated with these exponents (Brown et al., 1991). Compared to Vanilla, DySLIM returns more accurate exponents because it estimates the physical measure more accurately. However, our Neighborhood yields the most accurate predictions for the Lyapunov exponents, as it properly captures both ingredients –the global physical measure and the local Jacobians– that give rise to them.

The full Lyapunov spectra of the Hyperchaotic Chen and Lorenz96 systems are listed in Table 1. Except for the Lorenz96 system at 10% noise, our Neighborhood method also captures the smallest (most negative) exponent more accurately than the baselines. This exponent is associated with the "thin" direction of the attractor and is therefore particularly difficult to estimate from data, as the to-neighbor displacements contain little information along this direction (Brown et al., 1991).

B DATA GENERATION

B.1 OVERVIEW

For each ODE system considered in this paper, we generated train, validation and test datasets in the following manner:

1. Sample $m = 50$ random initial points on the attractor
2. Estimate the characteristic time scale τ of the system, via the average frequency spectrum.
3. Simulate m on-attractor trajectories in the time span $[0, 10\tau]$ with $\Delta t = 0.01\tau$
4. Shift and scale datasets using the train dataset statistics.

Sampling on-attractor initial points To sample points on the chaotic attractor, we first drew m initial conditions $\{\boldsymbol{u}_0^i\}_{i=1}^m$ from a normal distribution, with the mean and standard deviation chosen so that the sampled points would lie in the basin of attraction of the chaotic attractor. Afterwards, the dynamical equations were numerically integrated up to some prespecified burn-in time $t_{burn\text{-}in}$ for the trajectories to converge onto the chaotic attractor.

Table 1: Mean Lyapunov spectra (over three seeds) of the trained models for the Hyperchaotic Chen and Lorenz96 systems, compared with the ground truth.

System	Model (noise)	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
Chen ($d = 4$)	Ground truth	5.51e-1	1.27e-1	-3.69e-3	-2.61e1		
	Vanilla (1%)	5.38e-1	1.21e-1	-5.12e-3	-1.38e1		
	Vanilla (5%)	4.69e-1	1.27e-1	-3.12e-3	-1.19e2		
	Vanilla (10%)	1.84e-1	5.29e-2	-3.01e-3	-1.81e2		
	DySLIM (1%)	5.30e-1	1.47e-1	-3.94e-3	-9.64		
	DySLIM (5%)	4.83e-1	1.18e-1	-3.14e-3	-3.55e1		
	DySLIM (10%)	4.41e-1	1.37e-1	-2.74e-3	-4.30e1		
	Neighborhood (1%)	5.29e-1	1.50e-1	-3.82e-3	-1.19e1		
	Neighborhood (5%)	4.32e-1	1.19e-1	-2.91e-3	-3.38e1		
	Neighborhood (10%)	3.61e-1	1.80e-1	-2.78e-3	-1.07e1		
Lorenz96 ($d = 6$)	Ground truth	1.24	2.53e-3	-1.08e-1	-8.57e-1	-1.61	-4.67
	Vanilla (1%)	1.24	3.27e-3	-1.11e-1	-8.62e-1	-1.64	-4.74
	Vanilla (5%)	1.25	8.85e-4	-2.57e-1	-1.03	-1.92	-5.83
	Vanilla (10%)	1.20	2.42e-4	-5.87e-1	-1.34	-2.28	-7.42
	DySLIM (1%)	1.24	3.04e-3	-1.15e-1	-8.63e-1	-1.63	-4.71
	DySLIM (5%)	1.26	1.55e-3	-1.83e-1	-9.33e-1	-1.79	-5.31
	DySLIM (10%)	1.28	9.01e-4	-2.86e-1	-9.64e-1	-1.91	-5.74
	Neighborhood (1%)	1.24	3.42e-3	-1.07e-1	-8.52e-1	-1.62	-4.72
	Neighborhood (5%)	1.23	1.63e-3	-1.51e-1	-8.94e-1	-1.71	-5.20
	Neighborhood (10%)	1.23	8.23e-4	-2.76e-1	-9.86e-1	-1.83	-6.27

Characteristic time scale estimation Using the previously obtained initial conditions $\{\boldsymbol{u}_0^i\}_{i=1}^m$, we generated m long time trajectories, which were then Fourier transformed along each dimension to yield m sets of d frequency spectra. These were then averaged across trajectories to yield a single set of averaged frequency spectra. After a slight smoothing with a 1D Gaussian filter to remove spurious peaks, `scipy.signal.find_peaks` function (Virtanen et al., 2020) was used to identify the most prominent frequency peak for each dimension. Finally, the characteristic time scale τ was chosen as the median of the reciprocals of the discovered prominent frequencies per dimension.

Trajectory simulation We sample trajectories using the previously found m random initial conditions on the attractor $\{\boldsymbol{u}_0^i\}_{i=1}^m$. We normalize the simulation time and the sampling interval using the characteristic time scale τ to account for variations in the intrinsic timescales of different ODEs.

All numerical integration of the ODEs were performed using the `Tsit5` solver of the `diffrax` (Kidger, 2022) library, which is an implementation of Tsitouras's 5/4 method (Tsitouras, 2011). Adaptive time stepping was used to march the solutions forward in time, with relative and absolute tolerances of 10^{-8} and 10^{-10}, respectively.

Dataset normalization It is known that the differing size scales of the ODE dimensions can adversely affect neural ODE training (Kim et al., 2021). Therefore, we shifted and scaled the train dataset to have zero mean and unit standard deviation and used the same statistics to normalize the validation and test datasets.

Note that transforming the data in such a way changes the corresponding ground-truth ODE of a dataset. For an ODE and a bijection

$$\frac{d\boldsymbol{u}}{dt} = f(\boldsymbol{u}), \quad \boldsymbol{v} = g(\boldsymbol{u}) \tag{10}$$

the transformed state $\boldsymbol{v}$ evolves according to

$$\frac{d\boldsymbol{v}}{dt} = \partial g|_{g^{-1}(\boldsymbol{v})} f(g^{-1}(\boldsymbol{v})). \tag{11}$$

This effect was accounted for when computing the relative vector field and Jacobian errors in Section 6.3.

Noise level estimation The inner annulus radius $r_{min} = c\tilde{\epsilon}$ uses an estimate $\tilde{\epsilon}$ of the observational noise standard deviation computed from the noisy training data alone. Each noisy trajectory is detrended, dimension by dimension, with a smoothing cubic spline (`scipy.interpolate.make_smoothing_spline`, Virtanen et al., 2020) whose smoothing parameter is selected by the generalized cross-validation criterion, which requires no additional hyperparameters. The residuals between the noisy data and the smoothed trajectory are pooled and their standard deviation is taken as $\tilde{\epsilon}$.

B.2 Per-system settings

Here, we describe the dynamical equations of the systems used, as well as any system-specific parameters for data generation.

Lorenz63 system The Lorenz63 system (Lorenz, 1963) is given by the following equations:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sigma(y-x) \\ x(\rho - z) - y \\ xy - \beta z \end{bmatrix}. \tag{12}$$

We use the canonical parameter values $(\rho, \sigma, \beta) = (10, \frac{8}{3}, 28)$. The random initial conditions were sampled from the standard normal distribution, and following Frøyland & Alfsen (1984), we used a burn-in time of $t_{burn\text{-}in} = 50$.

Hyperchaotic Chen system The hyperchaotic Chen system was introduced in Li et al. (2005), as a modified version of the chaotic Chen system (Chen & Ueta, 1999). Hyperchaos means that the system has two or more positive Lyapunov exponents, in contrast to chaotic systems, which only have a single positive exponent. The governing equations are given by

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} a(y-x) + w \\ x(d-z) + cy \\ xy - bz \\ yz + rw \end{bmatrix}. \tag{13}$$

We use the parameters $(a, b, c, d, r) = (35, 3, 12, 7, 0.58)$, which are the parameters reported in the original paper, to give rise to hyperchaos. For the random initial conditions, we use samples from $\mathcal{N}([0, 0, 20, 0]^\mathsf{T}, I_{4\times 4})$, which we found allows the trajectories to converge faster onto the attractor. The burn-in time was set to $t_{burn\text{-}in} = 100$.

Lorenz96 system The Lorenz96 system (Lorenz, 1996) is system of ODEs defined on a periodic lattice of d nodes, given by

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \quad i = 0, \ldots, d-1. \tag{14}$$

For large values of the external force F, this system displays chaotic behavior. In our experiments, we used $F = 10$ and $d = 6$. We drew random initial conditions from the standard normal distribution and used a burn-in time of $t_{burn\text{-}in} = 1000$.

C Hyperparameter selection

To choose the hyperparameters for our main experiments, we performed a small grid search using a single random seed. Hyperparameter sweep was performed for each dataset and noise level, and we detail our selection procedure below and present the selection results in the following subsection.

C.1 Selection criteria overview

We designed our hyperparameter selection criteria to **prefer models with long-term statistical accuracy**. This was because hyperparameter combinations with the strongest short-term performances tended to have overfit on the short-term dynamics, especially for experiments with larger noise levels. Alternatively, choosing hyperparameters based on the accuracy of the learned Jacobians also had the

problem of unnecessarily skewing the results in our favor, as the baseline DySLIM method was not designed for this task. As both our Neighborhood method and DySLIM were both developed with the statistical accuracy of the trained models in mind, selecting hyperparameters based on long-term statistical accuracy ensured that both training methods had a chance of showcasing their efficacy.

To attribute a score to each hyperparameter combination in the grid search, we computed the Sinkhorn divergence (Cuturi, 2013; Genevay et al., 2018) –which is an optimal transport-based metric that also gauges the discrepancy between two measures using point samples– between the model predictions and the ground truth trajectories in the validation dataset. Here, we used the Sinkhorn divergence instead of the MMD^2 to avoid optimizing the hyperparameter selection by the exact metric used for subsequent model evaluations.

Sinkhorn divergences were computed using the `ott-jax` package (Cuturi et al., 2022) using trajectories from the validation dataset. As the 50 trajectory samples of length 10τ was too small to produce a reliable estimate of the Sinkhorn divergence, we first cut each trajectory into 10 pieces to produce 500 segments of length τ. Subsequently, the state values at the start of each trajectory segments were used as initial conditions to the trained Neural ODEs to produce 500 segment predictions. Finally, the Sinkhorn divergence between the data values and model predictions at the end of the segments (corresponding to the 500 point samples of the attractor evolved for time τ) was calculated using the default parameter settings of the `ott-jax` library, and used as the score for a given hyperparameter combination.

We comment that for this Sinkhorn divergence computation, the noiseless version of the validation dataset was used, contrary to model checkpointing where the noisy validation dataset was used. This choice was to decouple the effects of the hyperparameter selection process from the actual algorithm performances. Otherwise, the experimental results would measure the adequacy of the hyperparameter selection criteria in identifying the best performing models for each training method, as opposed to gauging the actual algorithm performances. Indeed, the original DySLIM paper (Schiff et al., 2024) also selects hyperparameters based on model performances on clean datasets, as the work does not consider noise effects. Nonetheless, a complete deployment of both our Neighborhood method and DySLIM to noisy, real world datasets would require a more effective hyperparameter selection strategy, which we leave for future work.

C.2 Hyperparameter selection results

DySLIM DySLIM has two hyperparameters λ_1 and λ_2, corresponding to the weights of its two additional loss functions. We used the same grid used in the original paper (Schiff et al., 2024) and performed a grid search over $\lambda_1 \in [0, 1, 10]$ and $\lambda_2 \in [1, 10, 100, 1000]$.

Neighborhood Our Neighborhood method also has two parameters: K, which is the number of neighbors sampled from each local neighborhood per training step, and λ, which is the weight factor for our neighborhood-based regularization term (Eq. (9)). We performed a grid search over $K \in [8, 16, 32, 64]$ and $\lambda \in [1, 10, 100, 1000]$.

D Evaluation metrics

As our goal is to assess how closely the models recover the ground truth dynamics, even when the training data is corrupted by noise, we perform all evaluations on the noiseless version of the test dataset (e.g., trajectory errors are computed with respect to the clean trajectories and not the noisy ones). All metrics, except for MMD^2 is computed using the 50 trajectories in the test dataset.

D.1 Valid prediction time

To evaluate the short-term accuracy of the model predictions, the first metric we introduce is the normalized root mean squared error between the data and model predictions, which for a trajectory prediction starting at the initial condition $\boldsymbol{u}_0^i$ is given as,

$$\text{NRMSE}^i(t_j) = \frac{1}{\sqrt{d}} ||\phi_{j\Delta t;\boldsymbol{\theta}}(\boldsymbol{u}_0^i) - \boldsymbol{u}_j^i||_2^2. \tag{15}$$

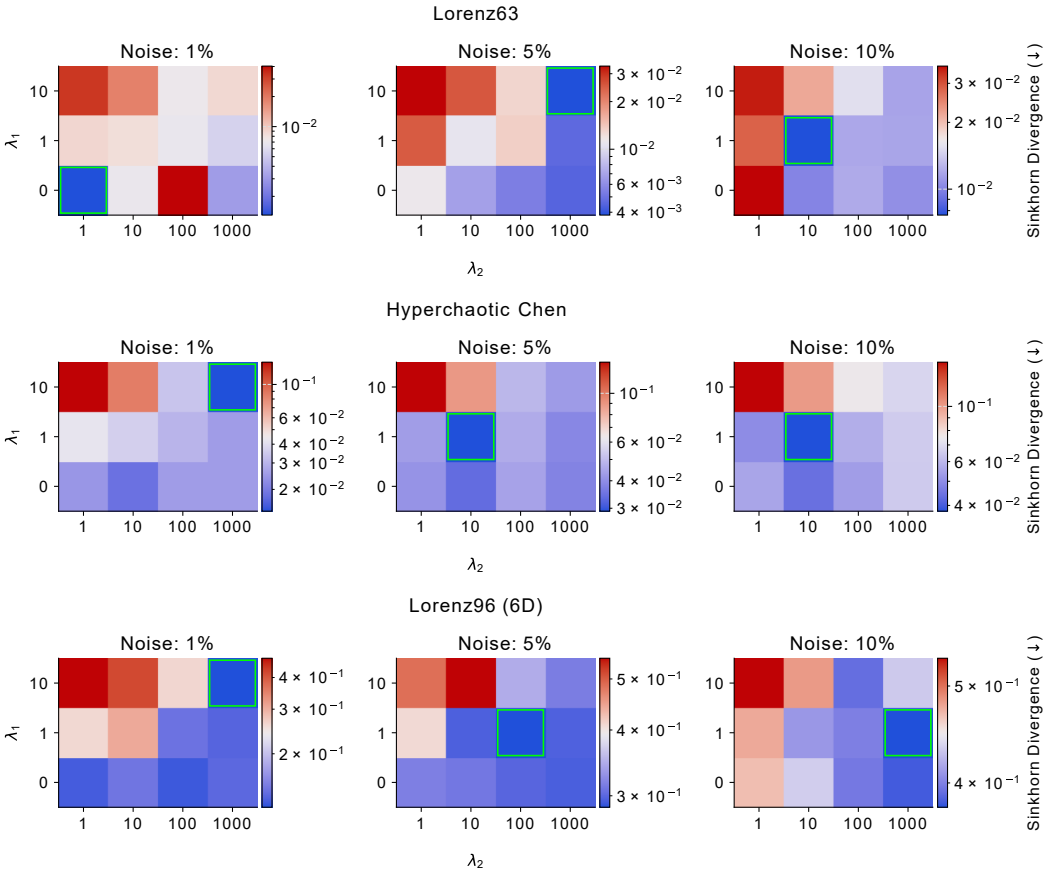

Figure 6: Hyperparameter sweep results for DySLIM. The green box indicates the selected hyperparameter combination.

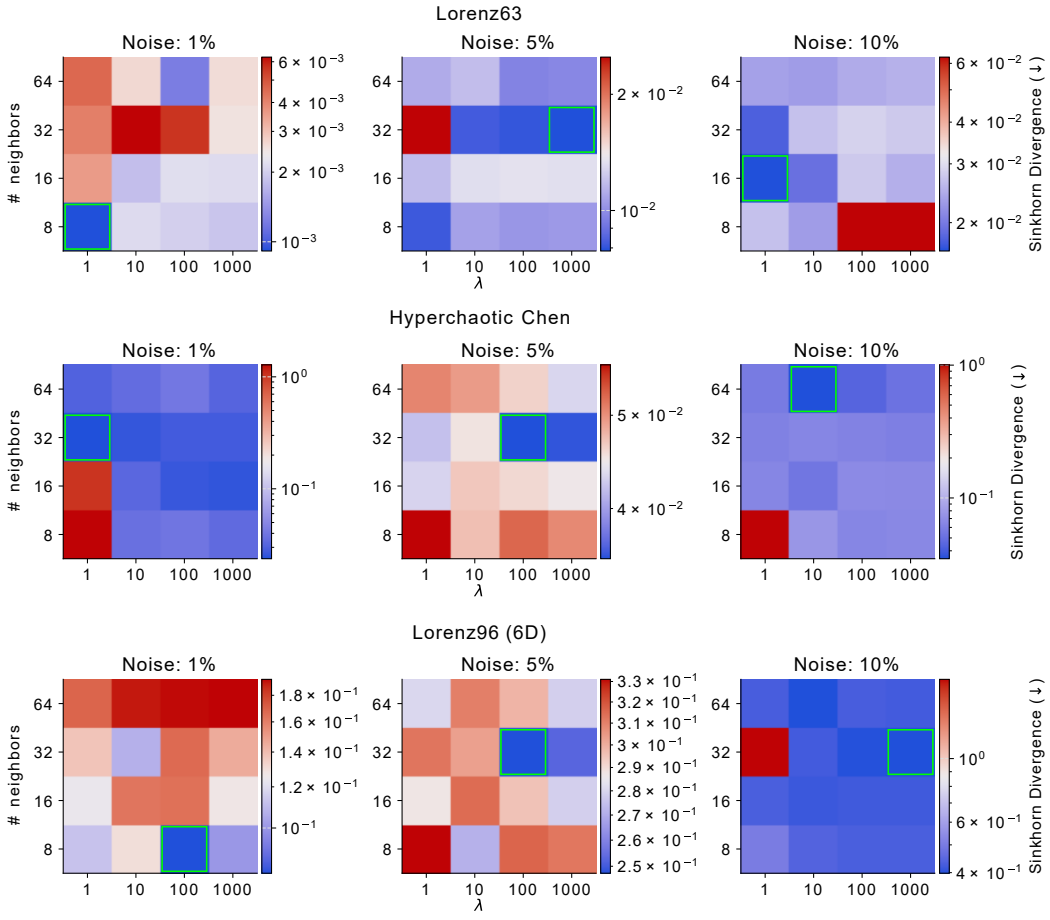

Figure 7: Hyperparameter sweep results for our Neighborhood method. The green box indicates the selected hyperparameter combination.

Note that while the original NRMSE requires scaling the error $\phi_{j\Delta t;\boldsymbol{\theta}}(\boldsymbol{u}_0^i) - \boldsymbol{u}_j^i$ with the standard deviation of the data trajectory before taking the norm, it is not necessary in our case as the data is affine transformed to have unit standard deviation along each dimension beforehand. For plots like Fig. 2a, these per trajectory NRMSEs can be averaged to yield the mean NRMSE:

$$\text{NRMSE}(t_j) = \frac{1}{m}\sum_{i=1}^{m} \text{NRMSE}^i(t_j). \tag{16}$$

The valid prediction time is defined as the maximum time until which the NRMSE value stays below a user-determined threshold ϵ, normalized by the maximum Lyapunov exponent λ_1:

$$\text{VPT}^i = \lambda_1 \cdot \max\{T \in \mathbb{R} \mid \text{NRMSE}^i(t) \leq \epsilon,\ \forall t \leq T\}. \tag{17}$$

To yield a single representative value for each experiment, these per-trajectory VPT values were also averaged as,

$$\text{VPT} = \frac{1}{m}\sum_{i=1}^{m} \text{VPT}^i. \tag{18}$$

For our analyses, the VPT threshold was set to $\epsilon = 0.3$.

D.2 Maximum mean discrepancy

The maximum mean discrepancy (MMD) is a metric that quantifies the difference between two measures μ and ν under mild assumption (Gretton et al., 2012). A big advantage of MMD is that it can be efficiently estimated using samples from the measures of interest. For samples $\{\boldsymbol{u}^k\}_{k=1}^K \sim \mu$ and $\{\boldsymbol{v}^k\}_{k=1}^K \sim \nu$, the squared MMD can be estimated using the sample as (Gretton et al., 2012),

$$\widehat{\text{MMD}^2}(\mu,\nu) = \frac{1}{K(K-1)}\sum_{k=1}^{K}\sum_{k'\neq k}^{K}\left[\kappa\left(\boldsymbol{u}^k,\boldsymbol{u}^{k'}\right) + \kappa\left(\boldsymbol{v}^k,\boldsymbol{v}^{k'}\right)\right] - \frac{2}{K^2}\sum_{k=1}^{K}\sum_{k'=1}^{K}\kappa\left(\boldsymbol{u}^k,\boldsymbol{v}^{k'}\right) \tag{19}$$

where $\kappa(\cdot,\cdot)$ is a suitable kernel function. For our analyses, we used the mixture of rational quadratic kernels (Schiff et al., 2024)

$$\kappa(\boldsymbol{u},\boldsymbol{v}) = \sum_{\sigma_q\in\boldsymbol{\sigma}} \frac{\sigma_q^2}{\sigma_q^2 + ||\boldsymbol{u}-\boldsymbol{v}||_2^2}; \quad \boldsymbol{\sigma} = \{0.2, 0.5, 0.9, 1.3\}. \tag{20}$$

In particular, since κ is a characteristic kernel, the condition $\text{MMD}^2 = 0$, implies $\mu = \nu$.

For the MMD^2 results, we made a separate test dataset of 5000 points on the attractor that are evolved for a longer period in time, as more samples and a longer time integration were needed in order to properly assess the quality of the learned measures. The on-attractor points were generated analogously to the data generation process, and the integration time was taken to be $[0, 100\lambda_1^{-1}]$ where the maximum Lyapunov exponent of each dataset was determined beforehand as according to Appendix D.4.

The baseline values for MMD^2 characterizes the value between two independent samples of the chaotic attractor that are both evolved according to the ground truth equations. This value corresponds to the best possible attainable value, when the model is able to exactly reproduce the long-term physical measure of the ground truth system. This baseline MMD^2 value was computed by taking two 5000 point samples on the ground truth attractor, evolving the both with the ground truth dynamics, then computing the MMD^2 for each time instant. Subsequently, the MMD^2 values for all time points were averaged to yield a single baseline value Fig. 8.

D.3 Relative vector field / Jacobian error

To evaluate the accuracy of the learned vector fields, we computed the relative vector field error for each point in the test dataset, as defined below:

$$\text{Rel. Vector Field Error}(\boldsymbol{u}) = \frac{||f_{\boldsymbol{\theta}}(\boldsymbol{u}) - f(\boldsymbol{u})||_2}{||f(\boldsymbol{u})||_2}, \tag{21}$$

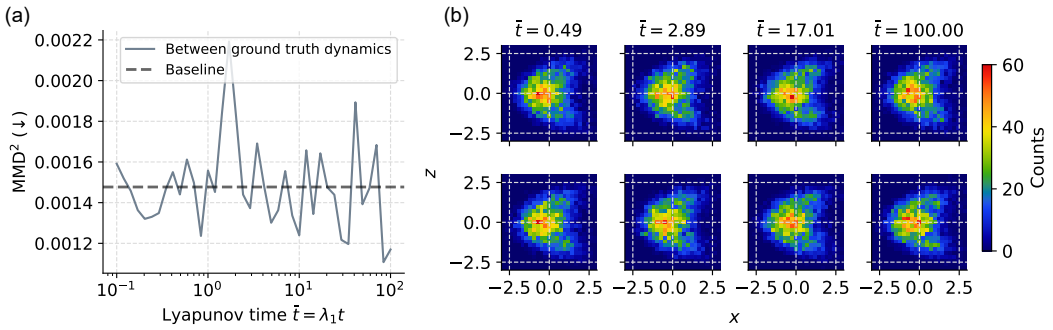

Figure 8: Calculation of baseline MMD^2 value for the hyperchaotic Chen system. **(a)** Time evolution of the MMD^2 computed between two samples of the chaotic attractor, both evolved using the ground truth dynamics. Note that unlike Fig. 3b, the values do not start from zero as the initial conditions for the two attractor samples are different. **(b)** Histograms of the ground truth attractor samples at different rollout times. The two rows correspond to the two attractor samples that are evolved in time.

where we used the shifted and scaled version of the ground truth equations (Eq. (11)) to supply the values of $f(\boldsymbol{u})$.

Similarly, the accuracy of the learned Jacobians at each point was assessed using the relative Jacobian error, defined as:

$$\text{Rel. Jacobian Error}(\boldsymbol{u}) = \frac{||\partial f_{\boldsymbol{\theta}}(\boldsymbol{u}) - \partial f(\boldsymbol{u})||_F}{||\partial f(\boldsymbol{u})||_F}, \tag{22}$$

where we once again accounted for the dataset normalization for the computation of the ground truth Jacobians. In the tables of Appendices F and G, we summarize these pointwise errors by their base-10 logarithm averaged over all test points ("average log relative error"); the accompanying $\pm$ values denote the standard deviation across the three training seeds unless stated otherwise.

D.4 Lyapunov exponents

The Lyapunov exponents characterize the asymptotic contraction and expansion rates of the chaotic attractor, and are given as the spectrum of the matrix (Oseledec, 1968; Christiansen & Rugh, 1997; Guckenheimer & Holmes, 1983)

$$\Lambda = \lim_{t\to\infty} \frac{1}{2t} \log(M_{\boldsymbol{u}}^{\mathsf{T}}(t) M_{\boldsymbol{u}}(t)). \tag{23}$$

where $M_{\boldsymbol{u}}(t)$ is the fundamental matrix solution of the variational equation (Eq. (24), second line).

While one can attempt to directly solve Eq. (24), this procedure is numerically unstable as $M_{\boldsymbol{u}}(t)$ quickly becomes singular in time. Instead, we use the algorithm of Shimada & Nagashima (1979), where for each initial condition $\boldsymbol{u}_0^i;\ i = 1, \ldots, m$, the algorithm produces an estimate of the Lyapunov spectrum $\{\lambda_r(\boldsymbol{u}_0^i)\}_{r=1}^d$ by solving the $d(d+1)$ coupled ODEs corresponding to the original ODE and its variational equation

$$\begin{aligned} \frac{d\boldsymbol{u}(t)}{dt} &= f(\boldsymbol{u}(t)), & \boldsymbol{u}(0) &= \boldsymbol{u}_0 \\ \frac{d\boldsymbol{M}_{\boldsymbol{u}}(t)}{dt} &= Df|_{\boldsymbol{u}} \boldsymbol{M}_{\boldsymbol{u}}(t), & \boldsymbol{M}_{\boldsymbol{u}}(0) &= \mathbb{I}_d \end{aligned} \tag{24}$$

over the time interval $[0, T]$ and performing Gram-Schmidt orthonormalization at fixed time intervals Δt to keep the fundamental matrix $\boldsymbol{M}_{\boldsymbol{u}}(t)$ nonsingular. Eq. (24) was solved using the same solver settings as data generation: `Tsit5` solver from the `diffrax` (Kidger, 2022) library, with relative and absolute tolerances of 10^{-8} and 10^{-10}, respectively.

To compute the mean absolute error of the Lyapunov spectrum, we first produced a mean Lyapunov spectrum estimate by averaging over the results from different initial conditions:

$$\bar{\lambda}_r = \frac{1}{m} \sum_{i=1}^{m} \lambda_r(\boldsymbol{u}_0^i). \tag{25}$$

Denoting $\bar{\lambda}_{r;\boldsymbol{\theta}}$ as the mean Lyapunov exponents computed using the trained models, mean absolute Lyapunov error was then computed as,

$$\text{Mean Abs. Lyapunov Error} = \frac{1}{d}\sum_{r=1}^{d}\left|\bar{\lambda}_{r;\boldsymbol{\theta}} - \bar{\lambda}_r\right|. \tag{26}$$

E Proof of Proposition 1

Proposition. *Let $(X, \mathcal{C})$ be a measurable space, and let ν_1, ν_2 be finite measures on $(X, \mathcal{C})$ such that $\nu_1(X \setminus \mathcal{A}) = \nu_2(X \setminus \mathcal{A}) = 0$ for a set $\mathcal{A} \in \mathcal{C}$ (both measures have support on $\mathcal{A}$).*

Furthermore, let $\{B_j\}_{j\in\mathbb{N}} \subset \mathcal{C}$ be a countable cover of $\mathcal{A}$, and assume that $\nu_1|_{B_j} = \nu_2|_{B_j}$, $\forall j$, that is

$$\nu_1(C) = \nu_2(C) \quad \textit{for all measurable } C \subset B_j,\ \forall j.$$

Then $\nu_1 = \nu_2$ on $\mathcal{C}$.

Proof. Let $A \in \mathcal{C}$. Since both measures vanish outside $\mathcal{A}$,

$$\nu_k(A) = \nu_k(A \cap \mathcal{A}), \quad k = 1, 2.$$

Thus it suffices to compare ν_1 and ν_2 on $A \cap \mathcal{A}$. Define

$$C_1 := A \cap \mathcal{A} \cap B_1, \qquad C_n := A \cap \mathcal{A} \cap B_n \setminus \bigcup_{k=1}^{n-1} B_k, \quad n \geq 2.$$

Then $\{C_n\}$ are pairwise disjoint, $C_n \subset B_n$, and since the B_j are a cover we have that

$$A \cap \mathcal{A} = \bigcup_{n=1}^{\infty} C_n.$$

By the assumption, $\nu_1(C_n) = \nu_2(C_n), \forall n$. Using countable additivity,

$$\nu_1(A) = \sum_{n=1}^{\infty} \nu_1(C_n) = \sum_{n=1}^{\infty} \nu_2(C_n) = \nu_2(A).$$

Since A was arbitrary, $\nu_1 = \nu_2$. □

F Ablation of the neighborhood radii

We ablate the two radii that define the annular neighborhoods on the Lorenz63 dataset, using the same protocol as the main experiments (three seeds; mean ± standard deviation over seeds) and reporting the four metrics of the main text: valid prediction time (VPT), MMD^2 at $\bar{t} = 100$, and the average log relative Jacobian and vector field errors over the attractor. Entries in *italics* mark configurations with clearly degraded performance. DySLIM is listed for reference.

Inner radius r_{min} The cutoff r_{min} removes neighbors that are so close to the center point that their relative displacement is obscured by noise: additive noise of strength ϵ perturbs a displacement by $\sim 2\epsilon$, so neighbors within this distance carry no usable information about the relative dynamics. We set $r_{min} = c\,\tilde{\epsilon}$ with $\tilde{\epsilon}$ estimated as described in Appendix B and vary $c \in \{0, 4, 8, 12\}$ ($c = 8$ in the main experiments), with r_{max} chosen as in the main text so that the annulus contains 5% of the points on average. Table 2 shows that performance is largely insensitive to c, except at high noise with small c, where the neighborhood information is corrupted. Intriguingly, the short-term accuracy is unaffected even in this regime, presumably because $\mathcal{L}_{traj}$ directly optimizes the short-term trajectory error. In all cases, the Jacobian accuracy remains better than DySLIM's.

Table 2: Ablation of the inner radius $r_{min} = c\,\tilde{\epsilon}$ on the Lorenz63 dataset.

Metric	Noise	$c = 0$	$c = 4$	$c = 8$ (default)	$c = 12$	DySLIM
VPT (↑)	1%	1.98±2.80e-1	2.30±2.51e-1	2.19±1.36e-1	2.22±3.96e-2	2.07±1.56e-2
	5%	1.77±2.02e-2	1.77±7.40e-2	1.68±1.65e-1	1.89±1.69e-1	1.88±2.18e-1
	10%	1.54±1.78e-1	1.60±9.84e-2	1.33±7.83e-2	1.38±8.88e-2	1.63±2.24e-2
MMD^2 (↓)	1%	1.89e-3±7.06e-4	2.93e-3±5.62e-4	2.86e-3±1.09e-3	1.35e-3±1.44e-4	1.57e-3±3.65e-4
	5%	*6.06e-1±5.03e-1*	2.64e-2±2.66e-2	3.57e-3±1.34e-3	3.79e-3±1.53e-3	5.93e-3±6.84e-3
	10%	*2.24±6.44e-1*	*2.11e-2±6.35e-3*	1.65e-2±1.19e-3	1.64e-2±4.56e-3	3.04e-3±1.27e-3
Log rel. Jac. err. (↓)	1%	-9.48e-1±3.29e-1	-9.25e-1±3.26e-1	-9.33e-1±3.21e-1	-9.54e-1±3.48e-1	-8.73e-1±3.29e-1
	5%	-8.23e-1±2.98e-1	-8.37e-1±3.03e-1	-8.22e-1±2.78e-1	-9.24e-1±2.74e-1	-4.26e-1±1.82e-1
	10%	*-5.98e-1±1.31e-1*	-7.64e-1±2.97e-1	-8.23e-1±2.90e-1	-8.37e-1±3.51e-1	-3.76e-1±1.43e-1
Log rel. VF err. (↓)	1%	-2.43±2.55e-1	-2.38±2.62e-1	-2.38±2.65e-1	-2.41±2.52e-1	-2.39±2.50e-1
	5%	-1.95±2.42e-1	-1.96±2.32e-1	-1.94±2.32e-1	-2.04±2.57e-1	-1.78±3.14e-1
	10%	*-1.68±2.54e-1*	-1.74±2.42e-1	-1.75±2.58e-1	-1.80±2.52e-1	-1.55±3.22e-1

Outer radius r_{max} We choose r_{max} so that the annulus $[r_{min}, r_{max}]$ contains on average $p\%$ of the points in the dataset ($p = 5$ in the main experiments), which qualitatively corresponds to the relative size of a neighborhood compared to the entire attractor. As long as the second-order Taylor expansion of Eq. (7) is accurate on $[r_{min}, r_{max}]$, we expect the performance to be stable; for larger p (and thus r_{max}), the truncation error of the expansion should begin to affect the learned physical measure and Jacobians. We vary $p \in \{2.5, 5, 7.5, 10\}\%$ with $c = 8$ fixed. Table 3 confirms this picture: performance is stable for the most part, with degradation for large noise and large r_{max}, and the Jacobian accuracy is affected most strongly. Even in the degraded configurations, however, the Jacobian error of our method is comparable to that of DySLIM.

Table 3: Ablation of the outer radius r_{max} (annulus containing $p\%$ of the data on average) on the Lorenz63 dataset.

Metric	Noise	$p = 2.5$	$p = 5$ (default)	$p = 7.5$	$p = 10$	DySLIM
VPT (↑)	1%	1.98±7.19e-2	2.19±1.36e-1	2.12±2.42e-2	2.09±1.17e-1	2.07±1.56e-2
	5%	1.71±1.37e-1	1.68±1.65e-1	1.84±1.67e-1	1.69±1.50e-1	1.88±2.18e-1
	10%	1.22±1.13e-1	1.33±7.83e-2	1.13±9.93e-2	1.12±9.48e-2	1.63±2.24e-2
MMD^2 (↓)	1%	3.52e-3±2.41e-3	2.86e-3±1.09e-3	1.42e-3±9.22e-5	1.97e-3±7.55e-4	1.57e-3±3.65e-4
	5%	2.96e-3±1.95e-3	3.57e-3±1.34e-3	2.68e-3±5.33e-4	*5.24e-3±2.94e-3*	5.93e-3±6.84e-3
	10%	2.21e-2±1.57e-3	1.65e-2±1.19e-3	2.52e-2±1.31e-3	*3.02e-2±4.72e-3*	3.04e-3±1.27e-3
Log rel. Jac. err. (↓)	1%	-8.86e-1±2.83e-1	-9.33e-1±3.21e-1	-9.60e-1±3.05e-1	-9.98e-1±2.84e-1	-8.73e-1±3.29e-1
	5%	-6.83e-1±2.02e-1	-8.22e-1±2.78e-1	-5.12e-1±1.86e-1	-7.76e-1±2.93e-1	-4.26e-1±1.82e-1
	10%	-8.68e-1±2.67e-1	-8.23e-1±2.90e-1	*-3.95e-1±2.04e-1*	*-3.64e-1±1.81e-1*	-3.76e-1±1.43e-1
Log rel. VF err. (↓)	1%	-2.36±2.41e-1	-2.38±2.65e-1	-2.39±2.41e-1	-2.42±2.78e-1	-2.39±2.50e-1
	5%	-1.90±2.22e-1	-1.94±2.32e-1	-1.93±2.78e-1	-1.93±2.58e-1	-1.78±3.14e-1
	10%	-1.73±2.30e-1	-1.75±2.58e-1	-1.68±2.43e-1	-1.66±2.57e-1	-1.55±3.22e-1

G Additional baselines

G.1 Jacobian-oracle regularization

To test whether regularizing Neural ODE training with Jacobian information is beneficial in principle, independently of how that information is obtained, we train a **Jacobian oracle** baseline with the loss

$$\mathcal{L}_{jac\text{-}oracle}(\boldsymbol{\theta}) = \mathcal{L}_{traj}(\boldsymbol{\theta}) + \lambda \cdot \frac{1}{S \cdot |\mathcal{D}_{train}|} \sum_{(i,j)\in\mathcal{D}_{train}} \sum_{s=1}^{S} \left\| \partial\phi_{s\Delta t;\boldsymbol{\theta}}|_{\boldsymbol{u}_j^i} - \partial\phi_{s\Delta t}|_{\boldsymbol{u}_j^i} \right\|_F^2, \quad (27)$$

where the Jacobian of the time-$s\Delta t$ map of the ground-truth dynamics is assumed to be supplied by an oracle. This is the setting studied by Park et al. (2025), and the Neural ODE analogue of Sobolev training (Czarnecki et al., 2017; Cho et al., 2025), which supervises a network on both function values and derivatives. The weight λ was selected from $\{0.1, 1, 10, 100\}$ with the same selection criterion as for the other methods (Appendix C); $\lambda = 10$ was selected for the 1% noise Hyperchaotic Chen dataset and $\lambda = 100$ for all other experiments.

The oracle is not a fair baseline We emphasize that this baseline is not attainable in practice, for three reasons. (1) The oracle is supplied with the *noiseless* Jacobians, whereas every other method sees only noisy trajectories; reliably estimating Jacobians from noisy data is itself extremely nontrivial (Appendix G.2). (2) The oracle supplies the full $d \times d$ Jacobian, while the information available from data consists of directional derivatives along the to-neighbor directions only. (3) Since the chaotic attractor is a lower-dimensional object that is "thin" along certain directions, the to-neighbor directions around a center point do not span $\mathbb{R}^d$, making full Jacobian recovery from data fundamentally impossible; this is also why the negative Lyapunov exponents associated with these thin directions are hard to recover (Appendix A). The oracle results should therefore be read not as a ceiling reachable with better algorithms, but as evidence for what Jacobian information does when it is available.

Results Tables 4 to 7 list the four metrics of the main text for all systems and noise levels, with Vanilla, DySLIM and Neighborhood repeated for reference. The oracle improves not only the Jacobian accuracy it directly optimizes, but also the short-term and long-term statistical accuracy, with margins that grow with the noise level. This is expected, since the clean Jacobians largely undo the effect of noise. (We did not corrupt the oracle Jacobians, for lack of a principled way to do so and to stay consistent with Park et al., 2025.) This reaffirms the finding of Park et al. (2025) and shows that the benefit of Sobolev-style training carries over to Neural ODEs: Jacobian-based regularization improves every aspect of the trained model. Our Neighborhood method is the natural next step, obtaining such regularization from noisy trajectory data alone while remaining cost-competitive with the baselines.

Table 4: Valid prediction time (↑) of the Jacobian-oracle baseline compared to the other methods (mean ± std over three seeds).

System	Method	1% noise	5% noise	10% noise
Lorenz63 ($d = 3$)	Vanilla	2.10±2.15e-1	1.23±5.65e-2	7.14e-1±1.41e-2
	DySLIM	2.07±1.56e-2	1.88±2.18e-1	1.63±2.24e-2
	Neighborhood	2.19±1.36e-1	1.68±1.65e-1	1.33±7.83e-2
	Jacobian oracle	1.92±3.78e-1	1.83±1.92e-1	2.14±1.46e-1
Chen ($d = 4$)	Vanilla	1.47±7.90e-2	9.95e-1±7.04e-2	6.62e-1±4.10e-2
	DySLIM	1.49±2.40e-1	9.28e-1±4.09e-2	7.95e-1±5.74e-2
	Neighborhood	1.68±1.40e-1	8.81e-1±7.71e-3	8.35e-1±5.67e-2
	Jacobian oracle	1.77±6.27e-2	1.80±1.51e-1	1.74±9.16e-2
Lorenz96 ($d = 6$)	Vanilla	1.76±1.55e-1	1.77±8.07e-2	1.27±8.54e-3
	DySLIM	1.75±2.32e-1	1.69±5.30e-2	1.67±9.32e-2
	Neighborhood	1.91±1.20e-1	1.72±3.72e-2	1.53±6.05e-2
	Jacobian oracle	1.28±1.51e-1	1.39±1.11e-1	1.83±7.48e-2

Table 5: MMD^2 at $\bar{t} = 100$ (↓) of the Jacobian-oracle baseline compared to the other methods.

System	Method	1% noise	5% noise	10% noise
Lorenz63 ($d = 3$)	Vanilla	1.62e-3±3.59e-4	1.36e-2±3.95e-4	6.60e-2±1.62e-3
	DySLIM	1.57e-3±3.65e-4	5.93e-3±6.84e-3	3.04e-3±1.27e-3
	Neighborhood	2.86e-3±1.09e-3	3.57e-3±1.34e-3	1.65e-2±1.19e-3
	Jacobian oracle	1.17e-3±1.17e-4	1.58e-3±6.36e-4	1.20e-3±4.95e-4
Chen ($d = 4$)	Vanilla	1.29e-3±2.90e-5	3.12e-3±1.84e-4	1.05e-2±9.13e-4
	DySLIM	1.47e-3±1.69e-4	2.35e-3±3.77e-4	2.93e-3±4.75e-4
	Neighborhood	1.51e-3±4.42e-4	3.15e-3±5.72e-4	5.13e-3±2.37e-4
	Jacobian oracle	1.66e-3±2.33e-4	1.45e-3±1.34e-4	1.34e-3±2.30e-4
Lorenz96 ($d = 6$)	Vanilla	1.52e-3±4.74e-6	2.07e-3±2.48e-4	3.84e-3±1.85e-4
	DySLIM	1.56e-3±5.27e-5	1.61e-3±6.23e-5	2.36e-3±2.62e-4
	Neighborhood	1.56e-3±8.17e-6	1.89e-3±3.90e-5	3.18e-3±4.68e-4
	Jacobian oracle	1.49e-3±7.72e-5	1.50e-3±2.75e-5	1.55e-3±5.92e-5

Table 6: Average log relative Jacobian error over the attractor (↓) of the Jacobian-oracle baseline compared to the other methods.

System	Method	1% noise	5% noise	10% noise
Lorenz63 ($d = 3$)	Vanilla	-7.69e-1±3.08e-1	1.87e-1±3.58e-1	3.92e-1±1.76e-1
	DySLIM	-8.73e-1±3.29e-1	-4.26e-1±1.82e-1	-3.76e-1±1.43e-1
	Neighborhood	-9.33e-1±3.21e-1	-8.22e-1±2.78e-1	-8.23e-1±2.90e-1
	Jacobian oracle	-2.88±1.99e-1	-2.85±1.88e-1	-2.77±1.62e-1
Chen ($d = 4$)	Vanilla	-3.84e-1±3.16e-1	2.30e-1±1.73e-1	4.27e-1±1.79e-1
	DySLIM	-3.57e-1±2.48e-1	-1.73e-1±1.51e-1	-9.39e-2±9.94e-2
	Neighborhood	-4.21e-1±2.74e-1	-2.30e-1±1.59e-1	-1.56e-1±9.47e-2
	Jacobian oracle	-2.68±2.55e-1	-2.72±2.53e-1	-2.73±2.36e-1
Lorenz96 ($d = 6$)	Vanilla	-1.53±1.47e-1	-1.08±1.09e-1	-7.69e-1±9.14e-2
	DySLIM	-1.47±1.47e-1	-1.10±1.27e-1	-8.53e-1±1.08e-1
	Neighborhood	-1.95±1.59e-1	-1.49±1.12e-1	-1.11±9.38e-2
	Jacobian oracle	-2.32±1.59e-1	-2.32±1.55e-1	-2.30±1.48e-1

Table 7: Average log relative vector field error over the attractor (↓) of the Jacobian-oracle baseline compared to the other methods. For Vanilla, DySLIM and Neighborhood, these values also answer how the vector field errors of Fig. 4 behave on the other two systems.

System	Method	1% noise	5% noise	10% noise
Lorenz63 ($d = 3$)	Vanilla	-2.36±2.82e-1	-1.54±3.93e-1	-1.20±4.08e-1
	DySLIM	-2.39±2.50e-1	-1.78±3.14e-1	-1.55±3.22e-1
	Neighborhood	-2.38±2.65e-1	-1.94±2.32e-1	-1.75±2.58e-1
	Jacobian oracle	-3.01±2.15e-1	-2.58±2.75e-1	-2.27±2.56e-1
Chen ($d = 4$)	Vanilla	-2.06±2.93e-1	-1.46±3.52e-1	-1.14±3.59e-1
	DySLIM	-2.00±3.07e-1	-1.62±3.00e-1	-1.42±3.05e-1
	Neighborhood	-2.06±3.26e-1	-1.71±3.38e-1	-1.46±3.31e-1
	Jacobian oracle	-2.62±2.21e-1	-2.61±2.77e-1	-2.14±2.48e-1
Lorenz96 ($d = 6$)	Vanilla	-2.06±1.95e-1	-1.64±2.00e-1	-1.30±2.16e-1
	DySLIM	-2.00±1.91e-1	-1.63±1.95e-1	-1.38±1.94e-1
	Neighborhood	-2.24±1.91e-1	-1.82±1.88e-1	-1.57±2.02e-1
	Jacobian oracle	-2.62±1.84e-1	-2.54±1.92e-1	-2.41±1.92e-1

G.2 Comparison to a local Jacobian estimation method

The oracle above uses clean Jacobians that are unattainable in realistic settings. To assess how accurately local Jacobians can actually be estimated from noisy trajectory measurements, we compare our trained models with a classical local Jacobian estimation method: the local second-order polynomial fit of Eckmann et al. (1986); Abarbanel et al. (1992). Let $\boldsymbol{\omega}_0^k = \boldsymbol{u}_{j_k}^{i_k} - \boldsymbol{u}_j^i$ be the displacement between a center point and its k-th neighbor, and $\boldsymbol{\omega}_1^k = \boldsymbol{u}_{j_k+1}^{i_k} - \boldsymbol{u}_{j+1}^i$ its single-timestep-evolved version, which satisfies

$$\boldsymbol{\omega}_1^k = \partial\phi_{\Delta t}(\boldsymbol{u}_j^i)[\boldsymbol{\omega}_0^k] + \tfrac{1}{2}\partial^2\phi_{\Delta t}(\boldsymbol{u}_j^i)[\boldsymbol{\omega}_0^k, \boldsymbol{\omega}_0^k] + \dots$$

The local method estimates the Jacobian of the time-Δt map by fitting, in each neighborhood, the model

$$\boldsymbol{\omega}_1^k = A_j^i[\boldsymbol{\omega}_0^k] + B_j^i[\boldsymbol{\omega}_0^k, \boldsymbol{\omega}_0^k], \qquad A_j^i \in \mathbb{R}^{d\times d},\ B_j^i \in \mathbb{R}^{d\times d\times d}, \tag{28}$$

by least squares, and takes A_j^i as the estimate of $\partial\phi_{\Delta t}(\boldsymbol{u}_j^i)$. We choose this method because it is the direct analogue, in the local Jacobian literature, of the second-order expansion used by our method, and its quadratic form makes it compatible with our annular neighborhoods. Unlike our method, which samples K neighbors per training step, the local fit uses all neighbors in the annulus (~ 2000 on average); being deterministic, it is reported as a single number.

Since the local method estimates $\partial\phi_{\Delta t}$ rather than ∂f, Table 8 reports the average log relative error of the *time-Δt map Jacobian* for all methods (for the Neural ODEs, $\partial\phi_{\Delta t;\boldsymbol{\theta}}$ is obtained by differentiating through the solver). We also report the local method on noiseless data. The local method is the most accurate estimator when the data are clean, but it is extremely susceptible to noise and falls below all Neural ODE-based methods, including Vanilla, already at 1% noise—a known problem in the literature (Yang & Wu, 2011; Yao et al., 2013) that has prompted more elaborate strategies such as grouping neighbors and averaging displacements within each group.

Table 8: Average log relative error (↓) of the time-Δt map Jacobian $\partial\phi_{\Delta t}$ for the local quadratic estimator and the Neural ODE-based methods (mean ± std over three seeds for the latter).

System	Method	0% noise	1% noise	5% noise	10% noise
Lorenz63 ($d = 3$)	Local quadratic	-2.82	-4.04e-1	-3.32e-1	-3.33e-1
	Vanilla	–	-1.81±9.55e-2	-9.62e-1±2.48e-1	-7.96e-1±5.23e-2
	DySLIM	–	-1.91±8.52e-3	-1.47±1.42e-2	-1.42±6.01e-3
	Neighborhood	–	-1.97±7.73e-3	-1.85±2.87e-2	-1.84±6.75e-2
Chen ($d = 4$)	Local quadratic	-1.60	-4.69e-1	-3.42e-1	-3.48e-1
	Vanilla	–	-8.41e-1±2.05e-2	-4.57e-1±1.33e-2	-4.03e-1±4.97e-3
	DySLIM	–	-8.39e-1±1.02e-2	-6.13e-1±1.26e-2	-5.52e-1±1.05e-2
	Neighborhood	–	-9.06e-1±9.72e-3	-6.62e-1±6.03e-3	-6.04e-1±6.50e-3
Lorenz96 ($d = 6$)	Local quadratic	-2.69	-1.69	-8.51e-1	-5.79e-1
	Vanilla	–	-2.41±9.77e-3	-1.97±1.20e-2	-1.66±2.38e-2
	DySLIM	–	-2.35±2.31e-2	-1.99±9.16e-3	-1.73±9.41e-3
	Neighborhood	–	-2.84±2.30e-2	-2.38±3.69e-3	-2.00±7.85e-3

These results expose a trade-off between global and local Jacobian estimation. Global methods assume a smooth vector field over phase space, which keeps nearby Jacobian estimates consistent, but the model is never explicitly trained on Jacobian-related information, which limits its accuracy. Local methods condition directly on neighbor displacements, but since an independent model is fitted in each neighborhood, nothing guides the Jacobians across phase space to be consistent with one another; with noise, each local model can overfit the noise in its neighborhood, leading to the observed degradation. Our Neighborhood method combines the strengths of both: the model is explicitly conditioned on local displacement information, while the global smooth prior of a single Neural ODE keeps its Jacobians consistent across phase space.

H Computational cost

Directional derivatives as nested JVPs Computing the full Hessian of a neural network is costly, with a cost relative to a network evaluation that scales as $O(d^2)$ in the data dimension d. The second-order expansion of Eq. (7), however, only requires the first- and second-order *directional*

derivatives $\partial f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}}[\tilde{\boldsymbol{\omega}}^k], \partial^2 f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}}[\tilde{\boldsymbol{\omega}}^k, \tilde{\boldsymbol{\omega}}^k] \in \mathbb{R}^d$, which are far cheaper than the full Hessian $\partial^2 f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}} \in \mathbb{R}^{d\times d\times d}$. We compute them with the nested JVP construction of Nzoyem et al. (2025). Writing $\mathrm{JVP}(f, \boldsymbol{u}, \boldsymbol{\omega}) = \partial f|_{\boldsymbol{u}}[\boldsymbol{\omega}]$ and defining

$$g(\boldsymbol{v}) := \partial f_{\boldsymbol{\theta}}|_{\boldsymbol{v}}[\tilde{\boldsymbol{u}} + \tilde{\boldsymbol{\omega}}^k - \boldsymbol{v}] = \mathrm{JVP}(f_{\boldsymbol{\theta}}, \boldsymbol{v}, \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{\omega}}^k - \boldsymbol{v}), \tag{29}$$

the chain rule gives $\mathrm{JVP}(g, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k) = \partial^2 f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}}[\tilde{\boldsymbol{\omega}}^k, \tilde{\boldsymbol{\omega}}^k] - \partial f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}}[\tilde{\boldsymbol{\omega}}^k]$, while $g(\tilde{\boldsymbol{u}}) = \mathrm{JVP}(f_{\boldsymbol{\theta}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k)$. Hence

$$\partial f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}}[\tilde{\boldsymbol{\omega}}^k] + \tfrac{1}{2}\partial^2 f_{\boldsymbol{\theta}}|_{\tilde{\boldsymbol{u}}}[\tilde{\boldsymbol{\omega}}^k, \tilde{\boldsymbol{\omega}}^k] = \tfrac{3}{2}\ \mathrm{JVP}(f_{\boldsymbol{\theta}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k) + \tfrac{1}{2}\ \mathrm{JVP}(g, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k). \tag{30}$$

As the JVP primitive in JAX returns $g(\tilde{\boldsymbol{u}}) = \mathrm{JVP}(f_{\boldsymbol{\theta}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k)$ as a by-product of evaluating $\mathrm{JVP}(g, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k)$, the additional cost of evolving a neighbor point is that of a single double-JVP, i.e., a forward-over-forward differentiation of $f_{\boldsymbol{\theta}}$.

Theoretical complexity The nested-JVP formulation is crucial to scalability because forward-mode differentiation has bounded complexity: for suitable time and memory measures (Griewank & Walther, 2008, Chapter 3.2),

$$\begin{aligned} \mathrm{TIME}(\mathrm{JVP}(f_{\boldsymbol{\theta}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k)) &\le C_{time} \cdot \mathrm{TIME}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{u}})), \\ \mathrm{MEM}(\mathrm{JVP}(f_{\boldsymbol{\theta}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k)) &= 2 \cdot \mathrm{MEM}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{u}})), \end{aligned} \tag{31}$$

where C_{time} is a task-specific constant independent of d, typically $C_{time} \approx 3$ (JAX developers, 2024b). Applying the bounds twice,

$$\begin{aligned} \mathrm{TIME}(\mathrm{JVP}(g, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k)) &\le C_{time}\, \mathrm{TIME}(g(\tilde{\boldsymbol{u}})) \le C_{time}^2\, \mathrm{TIME}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{u}})), \\ \mathrm{MEM}(\mathrm{JVP}(g, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\omega}}^k)) &= 2 \cdot \mathrm{MEM}(g(\tilde{\boldsymbol{u}})) = 4 \cdot \mathrm{MEM}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{u}})). \end{aligned} \tag{32}$$

For a batch of size $B = B'(K+1)$ (Eq. (9)), the B' center points are advanced with the regular Neural ODE and the $B'K$ neighbor points with the Taylor-expanded system, so that

$$\begin{aligned} \mathrm{TIME}(\text{Neighborhood}) &= B'\, \mathrm{TIME}(\text{Neural ODE}) + B'K\, \mathrm{TIME}(\text{Taylor-Neural ODE}) \\ &\le (B' + B'KC_{time}^2)\, \mathrm{TIME}(\text{Neural ODE}) \approx BC_{time}^2\, \mathrm{TIME}(\text{Neural ODE}), \end{aligned} \tag{33}$$

and similarly $\mathrm{MEM}(\text{Neighborhood}) \approx 4B \cdot \mathrm{MEM}(\text{Neural ODE})$. Vanilla and DySLIM solve the Neural ODE for each of the B points in the batch, at cost $B \cdot \mathrm{COST}(\text{Neural ODE})$, so that

$$\frac{\mathrm{TIME}(\text{Neighborhood})}{\mathrm{TIME}(\text{DySLIM})} \lesssim C_{time}^2 \approx 9, \qquad \frac{\mathrm{MEM}(\text{Neighborhood})}{\mathrm{MEM}(\text{DySLIM})} \approx 4. \tag{34}$$

The relative time and memory cost of our method is therefore a constant independent of the data dimension d, so the method is scalable to higher dimensions. In practice the constants are smaller than these bounds, owing to the optimizations performed by JAX during just-in-time compilation (JAX developers, 2024a).

Empirical cost To confirm the analysis, we measured the cost of the Neural ODE evaluation of our method relative to that of Vanilla/DySLIM, using the total number of floating point operations (FLOPs) as the time measure and the peak device memory as the memory measure. Since the goal is to assess cost rather than to train, $\tilde{\boldsymbol{u}}, \boldsymbol{\omega}^k$ were sampled from d-dimensional normal distributions with d ranging from 2 to 1024, using a total batch size $B = 2048$ and $K = 63$ neighbors. Table 9 shows that the relative cost is well below the theoretical bounds and, moreover, decreases towards 1 with increasing d; we attribute this to compiler optimizations together with the fixed hidden width (64) of our networks, which makes the d-dependent part of the computation a diminishing fraction of the total. Either way, the results confirm the dimension-independent scalability of the method.

Table 9: FLOPs and peak memory of the Neighborhood method's Neural ODE evaluation relative to Vanilla/DySLIM, as a function of the data dimension d ($B = 2048$, $K = 63$).

d	2	4	8	16	32	64	128	256	512	1024
Rel. FLOPs	3.21	3.08	2.87	2.56	2.18	1.82	1.53	1.35	1.20	1.14
Rel. memory	1.62	1.52	1.43	1.25	1.18	1.11	1.11	1.05	1.04	1.04

Training wall time Finally, Table 10 compares the end-to-end training wall time of the three methods on the datasets of the main text. Two effects beyond the above analysis enter here. First, DySLIM computes its MMD over all B points of a batch, requiring $O(B^2)$ pairwise distances, whereas our loss computes MMD within each neighborhood at cost $O(B'K^2) = O(BK)$, i.e., cheaper by a factor K/B. Second, with an adaptive ODE solver, parameter updates that make the Neural ODE stiff lead to smaller time steps and longer training; this affects all methods and contributes to the variance across seeds. The training times of the three methods are statistically indistinguishable on all three systems, confirming that our method is cost-competitive in practice.

Table 10: Training wall time in minutes (mean $\pm$ std over three seeds) on a single NVIDIA V100 GPU.

System (steps)	Method	1% noise	5% noise	10% noise
Lorenz63 (5000)	Vanilla	10.41±0.07	10.61±0.69	10.87±0.15
	DySLIM	11.06±0.11	11.12±0.30	10.40±0.46
	Neighborhood	10.83±0.06	9.87±1.79	9.85±1.25
Chen (5000)	Vanilla	8.92±0.50	10.09±0.12	9.74±0.65
	DySLIM	10.15±0.18	9.27±0.84	9.77±0.81
	Neighborhood	11.29±1.49	11.01±1.19	10.39±1.22
Lorenz96 (10000)	Vanilla	23.97±0.50	24.97±0.61	24.06±0.66
	DySLIM	24.60±0.66	25.75±0.90	24.18±0.22
	Neighborhood	24.72±2.25	22.35±3.27	25.64±2.77

I ADDITIONAL LONG-TERM METRIC: SINKHORN DIVERGENCE

As a complementary measure of long-term statistical agreement, Table 11 reports the Sinkhorn divergence (Cuturi, 2013; Genevay et al., 2018) between the long-time-evolved attractor samples of the ground-truth system and of the trained Neural ODEs, computed with the same 5000-point samples and rollout as the MMD^2 of Fig. 3c. Like MMD^2, the Sinkhorn divergence characterizes the distance between two measures from point samples, but is based on optimal transport. The relative ranking of the methods is identical to that of Fig. 3c, and we expect the same to hold for other measure-based metrics such as the KL divergence.

Table 11: Sinkhorn divergence (↓) between the ground-truth and model-evolved attractor samples at $\bar{t} = 100$ (mean $\pm$ std over three seeds).

System	Method	1% noise	5% noise	10% noise
Lorenz63 ($d = 3$)	Vanilla	6.44e-3±2.02e-3	5.33e-2±6.81e-3	2.04e-1±2.52e-2
	DySLIM	7.41e-3±2.44e-3	2.14e-2±2.35e-2	1.14e-2±4.01e-3
	Neighborhood	1.55e-2±9.17e-3	1.96e-2±9.40e-3	5.09e-2±5.13e-3
Chen ($d = 4$)	Vanilla	2.68e-2±2.32e-3	4.87e-2±2.30e-3	1.62e-1±7.63e-3
	DySLIM	2.66e-2±2.48e-3	3.93e-2±4.81e-3	5.02e-2±4.03e-3
	Neighborhood	2.77e-2±4.10e-4	5.04e-2±3.87e-3	8.49e-2±5.63e-3
Lorenz96 ($d = 6$)	Vanilla	1.02e-1±1.93e-3	1.15e-1±1.13e-2	1.62e-1±9.05e-3
	DySLIM	1.04e-1±1.50e-3	1.05e-1±4.14e-3	1.26e-1±4.77e-3
	Neighborhood	1.03e-1±1.89e-3	1.13e-1±2.23e-3	1.43e-1±8.92e-3

J EXTENSION TO DISCRETE-TIME ARCHITECTURES

Although we have focused on Neural ODEs, the framework extends in principle to models that parametrize the time-Δt map directly in observation space, $\boldsymbol{u}_{j+1} = F_{\boldsymbol{\theta}}(\boldsymbol{u}_j)$, or that evolve a latent state $\boldsymbol{h}_{j+1} = F_{\boldsymbol{\theta}}(\boldsymbol{h}_j)$ with an invertible observation map $\boldsymbol{u}_j = G_{\boldsymbol{\theta}}(\boldsymbol{h}_j)$; state-space models and recurrent networks for dynamics discovery are often designed this way (Mikhaeil et al., 2022; Hess et al., 2023). The only model-dependent ingredient is the prediction of the perturbation $\boldsymbol{\omega}^k$ between a

state and its k-th neighbor. Analogously to Eq. (7), a second-order expansion of the model map gives the recursion

$$\begin{aligned} \boldsymbol{u}^i_{j+s+1} &= F_{\boldsymbol{\theta}}(\boldsymbol{u}^i_{j+s}), \\ \boldsymbol{\omega}^k_{s+1} &= \partial F_{\boldsymbol{\theta}}|_{\boldsymbol{u}^i_{j+s}}[\boldsymbol{\omega}^k_s] + \tfrac{1}{2}\partial^2 F_{\boldsymbol{\theta}}|_{\boldsymbol{u}^i_{j+s}}[\boldsymbol{\omega}^k_s, \boldsymbol{\omega}^k_s], \qquad \boldsymbol{\omega}^k_0 = \boldsymbol{u}^{i_k}_{j_k} - \boldsymbol{u}^i_j, \end{aligned} \tag{35}$$

whose directional derivatives can again be computed with nested JVPs (Appendix H); no ODE solve is required, as the perturbation is advanced by direct iteration of the map. The remaining ingredients carry over unchanged: neighbor construction and the ground-truth side of the loss depend only on the measured data, and Proposition 1 is stated for pushforward measures under a measurable map, with no assumption of continuous time. Confirming that the improvements reported here transfer to such architectures requires additional experiments, which we leave for future work.