Parallel-in-Time Training of Recurrent Neural Networks for Dynamical Systems Reconstruction

Florian Hess[1,2†] **Florian Götz**[1,3] **Daniel Durstewitz**[1,2,4,5†]

[1]Dept. of Theoretical Neuroscience, Central Institute of Mental Health, Mannheim, Germany
[2]Faculty of Physics and Astronomy, Heidelberg University, Germany
[3]Faculty of Mathematics and Computer Science, Heidelberg University, Germany
[4]Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Germany
[5]ELLIS Unit Heidelberg, Germany

Abstract

Reconstructing nonlinear dynamical systems (DS) from data (DSR) is a fundamental challenge in science and engineering, but it inherently relies on *sequential* models. Recent breakthroughs for sequential models have produced algorithms that parallelize computation along sequence length T, achieving logarithmic time complexity, $\mathcal{O}(\log T)$. Since sequence lengths have been practically limited due to the linear runtime complexity $\mathcal{O}(T)$ of classical backpropagation through time, this opens new avenues for DSR. This paper studies two prominent classes of parallel-in-time algorithms for this task, both of which leverage *parallel associative scans* as their core computational primitive. The first class comprises models with linear yet non-autonomous dynamics and a nonlinear readout, such as modern State Space Models (SSMs), while the second consists of general nonlinear models which can be parallelized using the DEER framework. We find that the linear training-time recurrence of the first class of models imposes limitations that often hinder learning of accurate nonlinear dynamics. To address this, we augment DEER with Generalized Teacher Forcing (GTF), a novel variant within the more general nonlinear framework that ensures stable and effective learning of nonlinear dynamics across arbitrary sequence lengths. Using GTF-DEER, we investigate the benefits of training on extremely long sequences (T approaching 10^5) for DSR. Our results show that access to such long trajectories significantly improves DSR if the data features long time scales. This work establishes GTF-DEER as a robust tool for data-driven discovery and underscores the largely untapped potential of long-sequence learning in modeling complex DS.[1]

1 Introduction

Understanding and predicting the behavior of complex nonlinear systems from neural circuits and climate dynamics to fluid flows and ecological networks is a central aim across the natural and engineering sciences [6, 42, 43, 85]. A particularly ambitious goal is to *reconstruct* the underlying dynamical system (DS) directly from observed time series, a problem known as dynamical systems reconstruction (DSR). Beyond short-term forecasting, DSR requires that the learned model faithfully reproduces the long-term statistical and geometric properties of the true system, such as attractor geometry, power spectra, and Lyapunov exponents.

[†]Correspondence to florian.hess@zi-mannheim.de and daniel.durstewitz@zi-mannheim.de.

[1]Code to reproduce the results of this work is available at https://github.com/DurstewitzLab/Parallel-In-Time-Training-DSR.

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

Central to the definition of DS is the flow operator, which provides a recursive rule of how the DS evolves in time [45, 82]. DSR methods that approximate it are inherently recursive [35, 59], such as RNNs, commonly trained by backpropagation through time (BPTT; [92]). While several well-documented pathologies of BPTT – most notably exploding gradients under chaotic dynamics [59] – can be successfully mitigated by control-theoretic training algorithms such as sparse and generalized teacher forcing (STF/GTF) [40, 59], the computational cost of BPTT remains fundamentally linear in the sequence length T, rendering training on problems with long intrinsic timescales prohibitively expensive. Indeed, DSR applications have historically been confined to modest sequence lengths to keep training tractable [11, 40, 88], leaving open the question of whether learning from substantially longer sequences offers any benefit for reconstruction quality.

Recent advances in parallel-in-time sequence modeling offer a path forward, and we examine two common paradigms with respect to their performance in DSR. Linear training-time recurrences with a nonlinear readout, as instantiated by modern SSMs, admit trivial parallelization via linear scans and avoid chaos-induced exploding gradients by construction [59, 63, 93]. By exposing a *duality* between these linear SSMs and nonlinear RNNs, we show, however, that this convenience comes at a cost: the linear recurrence used during training imposes structural limitations that often prevent the model from learning accurate nonlinear dynamics. Moreover, its training suffers from exposure bias, degrading autoregressive roll-outs at test-time [4, 71]. The second paradigm – general nonlinear RNNs parallelized via DEER [50] – is in principle better suited to DSR, but naive application fails on chaotic data because the Jacobian products driving DEER's Newton updates diverge whenever the underlying dynamics exhibits positive Lyapunov exponents [29, 59].

Our main methodological contribution is to resolve this tension by combining DEER with Generalized Teacher Forcing (GTF) [40]. The resulting algorithm, **GTF-DEER**, inherits the long-sequence scalability of DEER while retaining GTF's chaos-taming properties: GTF turns the model into a stable DS during training such that GTF-DEER enjoys the average-case $\mathcal{O}((\log T)^2)$ scaling established in [29], even when the underlying system is chaotic. Empirically, GTF-DEER delivers speedups of up to $870\times$ over sequential training while matching or improving reconstruction quality. Equipped with GTF-DEER, we then revisit the central empirical question: *does training on much longer sequences actually improve DSR?* Leveraging the ability to train stably on trajectories of length T approaching 10^5, we find that access to long trajectories allows the DSR model to efficiently capture long time scales if present in the data, yielding substantial gains in long-term statistics, which an equivalent linear SSM based model cannot match. Taken together, our results establish GTF-DEER as a direct replacement of sequential training while highlighting long-sequence training as a largely untapped lever for DSR.

2 Related work

Dynamical systems reconstruction. Data-driven methods for DSR fall into two broad categories: The first approximates the *vector field* that underlies the observed dynamics, assuming the process is governed by differential equations. Sparse Identification of Nonlinear Dynamics (SINDy) and its variants [13, 14, 15, 17, 58] are particularly popular in the physical sciences and enable quick training through least-squares regression, but rely on pre-defined function libraries and struggle with noisy and partially observed or empirical data more generally. Neural ODE/PDE methods [1, 2, 16, 44, 46] offer universal approximation capabilities and can be augmented with physical priors [49, 69], but are difficult to train in practice. The second and larger class of models directly approximates the *flow operator* through universal approximators based on neural networks, including neural operators [48, 54], Koopman operators [3, 12, 55, 61, 64, 91] and hybrids thereof, reservoir computers [27, 65, 66, 67, 68, 86], and RNNs trained through BPTT [9, 11, 40, 72, 87, 88], often accompanied by specialized control techniques that ease optimization and address exploding gradients under chaos [11, 40, 59, 73]. The latter approach in particular achieves SOTA performance on a wide range of benchmark systems and performs well even on challenging empirical systems [10, 40, 90], which has led to its adoption as a backbone in foundation models for DSR [38].

Efficient sequence modeling. A central challenge in sequence modeling is to stably capture and retrieve long-term dependencies [5, 33, 34, 41, 63, 93]. One practical bottleneck when using autoregressive models such as SSMs or RNNs is the $\mathcal{O}(T)$ runtime of BPTT [92]. Recent developments in the field of sequence models address this issue by parallelizing the inherently sequential operation on

parallel accelerators, such as GPUs or TPUs, using *parallel associative scans* [7, 56, 81]. Parallel scans evaluate *linear* recurrences, often used as the core block of modern SSMs [33, 63, 81], in $\mathcal{O}(\log T)$ time. Furthermore, [50] introduced the DEER framework which parallelizes general *nonlinear* sequence models by reformulating the forward pass as a fixed-point iteration problem, where each iteration can be solved using a parallel scan. While the worst-case runtime complexity is $\mathcal{O}(T \log T)$ [30], the average-case observed in practice for models exhibiting contracting dynamics scales as $\mathcal{O}((\log T)^2)$ [28, 29], sparking a renaissance for nonlinear RNNs in long-term sequence modeling. Although DEER demonstrates promising performance in various ML problems, its applicability to the field of DSR remains impractical due to unavoidable worst-case scaling when evaluating chaotic RNNs.

3 Theoretical background

3.1 Dynamical systems reconstruction with autoregressive models

Given observed time series data $\boldsymbol{X} \in \mathbb{R}^{T_{\text{obs}} \times N}$ originating from some underlying physical process, DSR seeks to learn a generative model that is able to both perform accurate short-term predictions and reproduce the long-term behavior of the observed system. To tackle this task, we consider parameterized state space models of the form

$$\begin{aligned} \boldsymbol{z}_t &= F_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t-1}, \boldsymbol{s}_t) \\ \hat{\boldsymbol{x}}_t &= G_{\boldsymbol{\psi}}(\boldsymbol{z}_t), \end{aligned} \tag{1}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ denote parameter vectors, $\boldsymbol{z}_t$ is an M-dimensional state vector, $\hat{\boldsymbol{x}}_t$ are N-dimensional predicted observations, $\boldsymbol{x}_{t-1}$ is an optional teacher signal used during training, and $\boldsymbol{s}_t$ are K-dimensional optional external inputs. $F_{\boldsymbol{\theta}}$ is a discrete-time universal DS modeling a latent process which is coupled to the observations (data) through $G_{\boldsymbol{\psi}}$. The aim of training is to learn $\{\boldsymbol{\theta}, \boldsymbol{\psi}\}$ such that the model approximates the flow operator of the underlying DS, and after training we have

$$\boldsymbol{x}_t \approx G_{\boldsymbol{\psi}}(F_{\boldsymbol{\theta}}(\ldots F_{\boldsymbol{\theta}}(F_{\boldsymbol{\theta}}(\boldsymbol{z}_0, \boldsymbol{s}_1), \boldsymbol{s}_2) \ldots, \boldsymbol{s}_t)) =: G_{\boldsymbol{\psi}}(F_{\boldsymbol{\theta}}^{\circ t}(\boldsymbol{z}_0, \boldsymbol{s}_{1:t})). \tag{2}$$

In this work, we will investigate two general parameterizations of Eq. (1).

3.1.1 Linear training-time recurrences

In one setting, $F_{\boldsymbol{\theta}}$ is strictly *linear*, which shifts the burden of capturing nonlinearities in the data to a nonlinear observation function $G_{\boldsymbol{\psi}}$ which feeds back into the system at test time,

$$\begin{aligned} \boldsymbol{z}_t &= \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{U}\boldsymbol{x}_{t-1} + \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{h} \\ \hat{\boldsymbol{x}}_t &= \boldsymbol{B}\phi(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}), \end{aligned} \tag{3}$$

where $\boldsymbol{A} \in \mathbb{R}^{M \times M}$, $\boldsymbol{U} \in \mathbb{R}^{M \times N}$, $\boldsymbol{C} \in \mathbb{R}^{M \times K}$, $\boldsymbol{h} \in \mathbb{R}^M$, $\boldsymbol{B} \in \mathbb{R}^{N \times L}$, $\boldsymbol{V} \in \mathbb{R}^{L \times M}$, $\boldsymbol{b} \in \mathbb{R}^L$ and ϕ is a nonlinear function such as the $\mathrm{ReLU}(\boldsymbol{z}) = \max(0, \boldsymbol{z})$. The recurrence matrix $\boldsymbol{A}$ is usually parameterized as a *diagonal* matrix to boost computational efficiency and facilitate optimization [33, 63, 93], at the cost of a decrease in dynamical repertoire [60, 70]. Eq. (3) is essentially the architectural setup of modern SSMs, where linear recurrences are followed by non-linear, point-wise sequence transformations [33, 62, 63, 81], often implemented by MLPs. In our case, the MLP is a simple one-hidden-layer neural network with hidden layer size L. In the following, we will refer to models defined by Eq. (3) as 'LSSM'. A crucial insight is that while the recurrence in Eq. (3) constitutes a linear, *non-autonomous* DS during training, the model can produce nonlinear dynamics during evaluation by replacing the teacher signals $\boldsymbol{x}_{t-1}$ with predictions $\hat{\boldsymbol{x}}_{t-1}$. Indeed, during trajectory generation *after* training, the recursion of Eq. (3) turns into

$$\boldsymbol{z}_t = \boldsymbol{A}\boldsymbol{z}_{t-1} + \overline{\boldsymbol{W}}\phi(\boldsymbol{V}\boldsymbol{z}_{t-1} + \boldsymbol{b}) + \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{h} \tag{4}$$

where $\boldsymbol{U}\boldsymbol{B} =: \overline{\boldsymbol{W}} \in \mathbb{R}^{M \times L}$ with $\mathrm{rank}(\overline{\boldsymbol{W}}) \leq \min(N, M, L)$. This allows (L)SSMs to exhibit inherently nonlinear traits such as chaotic dynamics and multistability during autoregressive generation. While the major feature of Eq. (3) is the fact that the forward pass can be calculated in logarithmic time using parallel scan, another insight is that linear recurrences do not suffer from chaos-induced exploding gradients [59], and hence remedy associated training instabilities by design [59, 63, 93]; see Appx. A for details. Training of LSSMs makes use of a variant of teacher forcing originating

from the sequence modeling field, where ground-truth data from the previous time step $\boldsymbol{x}_{t-1}$ is fed through a dedicated input layer ($\boldsymbol{U}$) [4, 31]; a generic training algorithm is provided in Alg. 1. This type of training suffers from *exposure bias*, where the absence of forcing signals during evaluation (Eq. (4)) significantly degrades autoregressive roll-outs [71]. While methods such as scheduled sampling address this issue [4, 88], they destroy linearity and therefore efficient parallelization of the recurrence in Eq. (3) during training; see Appx. C for details.

3.1.2 Non-linear training-time recurrences

To compare training algorithms, we will consider a second parameterization of Eq. (1), which uses a nonlinear RNN as the latent model, coupled to the observations through a simple linear mapping:

$$\begin{aligned} \boldsymbol{z}_t &= \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{W}\phi(\boldsymbol{V}\boldsymbol{z}_{t-1} + \boldsymbol{b}) + \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{h} \\ \hat{\boldsymbol{x}}_t &= \boldsymbol{B}\boldsymbol{z}_t. \end{aligned} \tag{5}$$

With $\phi(\cdot) = \mathrm{ReLU}(\cdot)$, the latent model is known as a shallow piecewise-linear RNN (shPLRNN; [40]), an established RNN architecture designed for DSR. Indeed, up to the constraint of a low-rank connectivity matrix, which can be explicitly enforced in the latter parameterization, the recurrences of Eqs. (4) and (5) are *mathematically equivalent*. This is important as it allows us to *directly compare training algorithms and parameterizations, unconfounded* by any architectural differences at test time, studying optimized models with parameters $\boldsymbol{\theta}_{\mathrm{RNN}} \cup \boldsymbol{\psi}_{\mathrm{RNN}} = \{\boldsymbol{A}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{b}, \boldsymbol{C}, \boldsymbol{h}, \boldsymbol{B}\}$ and $\boldsymbol{\theta}_{\mathrm{LSSM}} \cup \boldsymbol{\psi}_{\mathrm{LSSM}} = \{\boldsymbol{A}, \overline{\boldsymbol{W}}, \boldsymbol{V}, \boldsymbol{b}, \boldsymbol{C}, \boldsymbol{h}, \boldsymbol{B}\}$. All results established below hold in the presence of external inputs $\boldsymbol{s}$, but we will drop them from notation for brevity.

Due to the nonlinearity of the latent model in Eq. (5), the forward pass is evaluated sequentially and cannot be naively parallelized. Hence, a common method to train for DSR is to generate trajectories of length T and then compare them to the data $\boldsymbol{x}_{1:T}$. To avoid training instabilities caused by, for example, exploding gradients, training is accompanied by control-theoretic forcing methods such as Generalized Teacher Forcing (GTF; [21, 40]). GTF alters the forward pass of the model during training by linearly interpolating between the latent state and a teacher signal before application of the sequence model:

$$\boldsymbol{z}_t = F_{\boldsymbol{\theta}} \circ \delta_\alpha(\boldsymbol{z}_{t-1}, \overline{\boldsymbol{z}}_{t-1}) = F_{\boldsymbol{\theta}}((1-\alpha)\boldsymbol{z}_{t-1} + \alpha\overline{\boldsymbol{z}}_{t-1}) = F_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_{t-1}), \tag{6}$$

where $0 \leq \alpha \leq 1$ is the forcing strength and $\overline{\boldsymbol{z}}$ denotes the teacher signal which is generally computed by inversion of the observation model or output layer, i.e. $\overline{\boldsymbol{z}}_t = G_{\boldsymbol{\psi}}^{-1}(\boldsymbol{x}_t)$,[2] and $\tilde{\boldsymbol{z}}$ is the forced state. During training, this leads to decomposition of the Jacobian as

$$\boldsymbol{J}_{F\circ\delta}(\boldsymbol{z}_{t-1}) = (1-\alpha)\boldsymbol{J}_F(\tilde{\boldsymbol{z}}_{t-1}), \tag{7}$$

where $\boldsymbol{J}_F(\boldsymbol{z}) := \frac{\partial F(\boldsymbol{z})}{\partial \boldsymbol{z}}$. The forcing parameter $\alpha \in [0, 1]$ controls the norm of the Jacobians during BPTT and can be optimally and adaptively adjusted in training to mitigate exploding gradients ([40], see also Appx. F.3).

For linear observation models (Eq. (5)) and the common case where the RNN has more units than there are observed dynamical variables ($M > N$), [73] introduced a correction to GTF (6) which includes the row-space projector of $\boldsymbol{B}$ in the forcing equation:

$$\delta_{\alpha,\boldsymbol{B}}(\boldsymbol{z}_t, \boldsymbol{x}_t) = (\boldsymbol{I} - \alpha\boldsymbol{B}^+\boldsymbol{B})\boldsymbol{z}_t + \alpha\boldsymbol{B}^+\boldsymbol{x}_t, \tag{8}$$

where $\boldsymbol{B}^+$ denotes the pseudo-inverse of $\boldsymbol{B}$, such that Eq. (7) becomes

$$\boldsymbol{J}_{F\circ\delta_{\alpha,\boldsymbol{B}}}(\boldsymbol{z}_{t-1}) = \boldsymbol{J}_F(\tilde{\boldsymbol{z}}_{t-1})\boldsymbol{P}_\alpha, \tag{9}$$

where $\boldsymbol{P}_\alpha := \boldsymbol{I} - \alpha\boldsymbol{B}^+\boldsymbol{B}$. While this correction does not preserve the mitigation of exploding gradients in general, it dampens Jacobian singular values along directions corresponding to expansion in observation space. Indeed, we find empirically that Jacobian damping still holds in practice in this case, which we attribute to mixing of directions through the Jacobian product. Note that for $M \leq N$ and assuming full column-rank, $\boldsymbol{B}^+\boldsymbol{B} = \boldsymbol{I}$ and vanilla GTF is recovered. For a detailed overview of the training protocol under GTF, see Alg. 2.

[2]The observation model does not need to be strictly invertible and often an approximate inverse, such as the pseudo-inverse in case of linear or affine $G_{\boldsymbol{\psi}}$, is used [40, 59]. Alternatively, the latent model may be embedded in an encoder-decoder architecture, where forcing signals are given through the encoder mapping [10].

3.1.3 Parallel-in-time algorithms for nonlinear sequence models

Training nonlinear flow operators on long sequences sequentially is prohibitive due to the linear $\mathcal{O}(T)$ scaling of BPTT in sequence length. To make training feasible, we will make use of the recently proposed DEER algorithm [30, 50]. Given an initial condition $\boldsymbol{z}_0$, let $\boldsymbol{z}_{1:T}$ be a series of candidate latent states and define the residual vector $\boldsymbol{r}(\boldsymbol{z}_{1:T}) := [\boldsymbol{z}_1 - F(\boldsymbol{z}_0), \ldots, \boldsymbol{z}_T - F(\boldsymbol{z}_{T-1})] \in \mathbb{R}^{MT}$. The fixed point $\boldsymbol{z}^*_{1:T} = F(\boldsymbol{z}^*_{0:T-1})$ is the only solution with zero residual, i.e. $\boldsymbol{r}(\boldsymbol{z}^*_{1:T}) = \boldsymbol{0}$. Conventionally, the forward pass is generated sequentially by iterating F starting from $\boldsymbol{z}_0$. DEER turns the roll-out of the map F into a root-finding problem for the residual, which is solved using Newton's method. Starting from an initial guess $\boldsymbol{z}^{(0)}_{1:T}$, where the superscript indicates the current Newton iteration, the true trace $\boldsymbol{z}^*_{1:T}$ is approximated by iteratively solving the update equation

$$\boldsymbol{z}^{(i+1)}_{1:T} = \boldsymbol{z}^{(i)}_{1:T} - \left(\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{z}_{1:T})\right)^{-1} \boldsymbol{r}\left(\boldsymbol{z}^{(i)}_{1:T}\right), \tag{10}$$

where $\boldsymbol{J}_{\boldsymbol{r}} := \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{z}_{1:T}}$. Inverting the Jacobian matrix explicitly is infeasible for large MT. Instead, it is more practical to multiply both sides of the equation by the Jacobian and exploit its block-bidiagonal structure (cf. Eq. (43) in Appx. E) to obtain a recursive formula for $\Delta\boldsymbol{z}^{(i+1)}_t := \boldsymbol{z}^{(i+1)}_t - \boldsymbol{z}^{(i)}_t$:

$$\Delta\boldsymbol{z}^{(i+1)}_t = \boldsymbol{J}_F\left(\boldsymbol{z}^{(i)}_{t-1}\right)\Delta\boldsymbol{z}^{(i+1)}_{t-1} - \left[\boldsymbol{r}(\boldsymbol{z}^{(i)}_{1:T})\right]_t \tag{11}$$

Equation (11) is linear in $\Delta\boldsymbol{z}$ and hence each Newton iteration can be solved in $\mathcal{O}(\log T)$ time by a parallel associative scan. Due to its usage of the full Jacobian and hence full matrix products, increasing the dimensionality of the state of the sequence model can become a bottleneck. To address this, [30] suggested the use of quasi-Newton methods which replace the full Jacobian in Eq. (11) by its diagonal $\mathrm{diag}(\boldsymbol{J}_F) \in \mathbb{R}^M$, reducing the *work* $\mathcal{W}$ of each Newton iteration from $\mathcal{W}_{\text{full}} = \mathcal{O}(M^3T)$ to $\mathcal{W}_{\text{diag}} = \mathcal{O}(MT)$.

While DEER and its variants can be directly used to train sequence models on data from nonlinear DS, the methods suffer from Jacobian divergence when tasked to generate orbits from systems with (on average) unstable dynamics, i.e. where $\|\boldsymbol{J}_F(\boldsymbol{z}_t)\|_2 > 1$ for most t [29]. This divergence is inevitable when training on chaotic systems [59]. Since most if not all natural complex DS are chaotic [23, 26, 32, 80, 84], this needs to be amended.

4 Methods

4.1 DSR from long sequences

To enable stable training on long sequences of chaotic dynamics, we propose to combine GTF and DEER, which we coin GTF-DEER. By simply replacing the RNN map $F_{\boldsymbol{\theta}}$ with the teacher-forced variant (6), the residual becomes the vector with entries

$$[\boldsymbol{r}_\alpha(\boldsymbol{z}_{1:T}, \boldsymbol{x}_{1:T})]_t = \boldsymbol{z}_t - F\left(\delta_{\alpha,\boldsymbol{B}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t-1})\right) \tag{12}$$

for $t = 2 \ldots T$ and $[\boldsymbol{r}_\alpha(\boldsymbol{z}_{1:T}, \boldsymbol{x}_{1:T})]_1 = \boldsymbol{z}_1 - F_{\boldsymbol{\theta}}(\boldsymbol{z}_0)$ since $\boldsymbol{z}_0$ is given. Hence the linear recursion uses the decomposed Jacobians (9)

$$\Delta\boldsymbol{z}^{(i+1)}_t = \left[\boldsymbol{J}_F(\tilde{\boldsymbol{z}}^{(i)}_{t-1})\boldsymbol{P}_\alpha\right]\Delta\boldsymbol{z}^{(i+1)}_{t-1} - \left[\boldsymbol{r}_\alpha\left(\boldsymbol{z}^{(i)}_{1:T}, \boldsymbol{x}_{1:T}\right)\right]_t \tag{13}$$

Similar to sequential GTF, Jacobian divergence caused by chaotic dynamics is tamed by $\boldsymbol{P}_\alpha$ which for $M \leq N$ strictly turns a DS with diverging state space directions into a contracting one, improving DEER convergence [29]. To demonstrate this formally, we need the following definitions.

Definition 1. The *largest Lyapunov exponent* (LLE) of a trajectory of the iterated map F starting at $\boldsymbol{z}_0$ is

$$\lambda(\boldsymbol{z}_0) = \lim_{T\to\infty} \frac{1}{T} \log \|\boldsymbol{J}_T \boldsymbol{J}_{T-1} \ldots \boldsymbol{J}_1\|_2, \tag{14}$$

where $\boldsymbol{J}_t := \boldsymbol{J}(\boldsymbol{z}_{t-1}) := \frac{\partial F(\boldsymbol{z}_{t-1})}{\partial \boldsymbol{z}_{t-1}}$.

We say the iterated map F is *divergent* in $\boldsymbol{z}_0$ if $\lambda(\boldsymbol{z}_0) > 0$. When $\lambda(\boldsymbol{z}_0) < 0$, F is *contracting*.

Proposition 1. *Let an SSM be given by a latent model $F : \mathbb{R}^M \to \mathbb{R}^M$ and a linear observation model $G : \mathbb{R}^M \to \mathbb{R}^N$, $\boldsymbol{z} \mapsto \boldsymbol{B}\boldsymbol{z}$ where $M \leq N$ and $\boldsymbol{B}$ is assumed to have full rank. Let the supremal Jacobian norm satisfy $\tilde{\sigma}_{\max} := \sup \left\{ \|\boldsymbol{J}(\boldsymbol{z})\|_2 = \sigma_{\max}(\boldsymbol{J}(\boldsymbol{z})) \mid \boldsymbol{z} \in \mathbb{R}^M \right\} > 1$. Choose the GTF forcing strength $\alpha \in (\alpha^*,\ 1]$, where $\alpha^* := 1 - \tilde{\sigma}_{\max}^{-1}$ [40], and define the effective contraction rate*

$$\rho := (1 - \alpha)\, \tilde{\sigma}_{\max} < 1. \tag{15}$$

Then the forced system $F \circ \delta_{\alpha,\boldsymbol{B}}$ is globally contracting with rate ρ. Its LLE satisfies

$$\lambda_{\mathrm{GTF}} \leq \log \rho < 0. \tag{16}$$

Proof. See Appx. B. □

Proposition 1 says that given a suitable α, training a nonlinear sequence model using GTF-DEER leads to guaranteed convergence of the forward pass regardless of the dynamics that underlie the data, enabling stable and efficient parallel-in-time training for DSR. We validate this theoretical finding in Sect. 5.1 and investigate the effect of $M > N$ on GTF-DEER convergence empirically. Moreover, since we directly want to match forced model trajectories to targets given by the data, we can initialize the Newton iterations of DEER using the forcing signals, i.e. $\boldsymbol{z}_{1:T}^{(0)} = \overline{\boldsymbol{z}}_{1:T}$, which speeds up convergence over the typical initialization by zeros [30, 50]. We provide the training routine using GTF-DEER in Alg. 3.

4.2 Capturing long-term dependencies

Regularization. To effectively capture long-term dependencies, sequence models must maintain stable flow of information over extended temporal horizons [5, 34, 41, 93]. Crucially, for maintaining connections among states across time we must not only avoid chaotic divergence, but also prevent error signals propagated backward from decaying too fast. In this work we therefore make use of Manifold Attractor Regularization (MAR; [76]) to ameliorate diminishing gradients by equipping the sequence model with a latent subspace that adaptively captures slow timescales in the data. In Appx. F, we review the MAR contribution to the loss function and show how this regularization leads to stable error propagation by imposing a block-diagonal identity structure in the model Jacobian $\boldsymbol{J}_F$.

Latent state warm-up. To provide the RNN with sufficient dynamical context during training, we employ latent state warm-up [75, 89]. During training we sample sequences of length T, where the first T_w steps are used for warm-up. The warm-up consists of forcing the sequence model with data $\boldsymbol{x}_{1:T_w}$, either by feeding them through $\boldsymbol{U}$ for the LSSM (Eq. (3)) or by GTF in the case of the nonlinear RNN (Eqs. (6) and (8)) with $\alpha = 1$. This yields a latent state $\boldsymbol{z}_{T_w}$ which holds a compressed history of the warm-up signal, which is then used to predict the remaining $T - T_w$ time steps that contribute to the loss function given by the mean-squared-error $\mathrm{MSE}(\boldsymbol{x}_{T_w+1:T}, \hat{\boldsymbol{x}}_{T_w+1:T})$. Crucially, the forward pass of the entire sequence of length T *including* the warm-up can be computed by a single call to GTF-DEER, enabling long contexts during training (cf. Alg. 3).

Performance measures. To evaluate DSR performance we use an established long-term measure which computes the KL divergence between multivariate state distributions in observation space, D_{stsp} [40, 47]. For partially observed systems, we first perform a delay embedding [74, 83] of ground-truth and generated trajectories and compute D_{stsp} on these embeddings. We will make delay embeddings explicit in notation by writing the measure as $D_{\mathrm{stsp}}^{\mathrm{DE}}$. We also report the 128-step root-mean-square-error (RMSE) as a measure of short- to medium-length prediction accuracy. For mathematical and experimental details on evaluation measures, see Appx. H.

5 Results

We will first evaluate the computational efficiency of GTF-DEER by performing runtime analyses. We will then investigate the benefit of long-sequence training for DSR through a set of ablation experiments and comparisons to state-of-the-art sequence models. Specifically, we 1) ask whether long sequences improve DSR and 2) reveal and explain the shortcomings of SSMs for DSR.

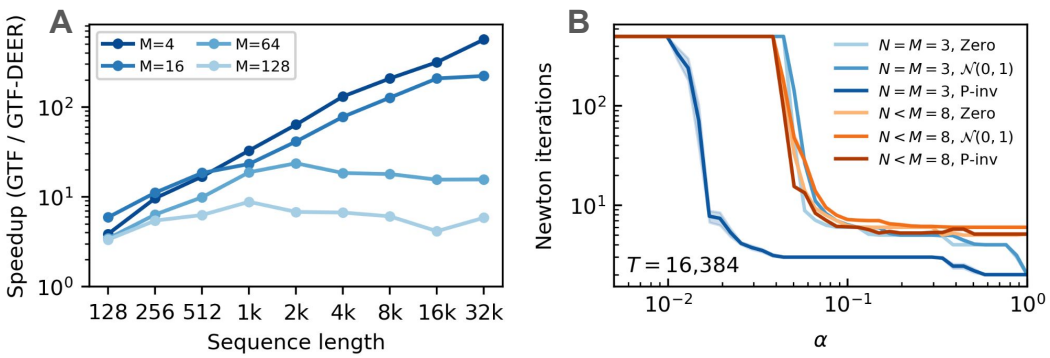


Figure 1: **A**: GTF-DEER scales favorably in sequence length, but the GPU quickly saturates for larger problems. Note the logarithmic scaling of the y-axis. **B**: The forcing parameter α controls Jacobian norms and hence reduces the number of Newton iterations needed for convergence of the GTF-DEER forward pass. Initializing the model trajectory using estimated latents through $\hat{\boldsymbol{z}}_t = \boldsymbol{B}^+ \boldsymbol{x}_t$ ('P-inv') improves GTF-DEER convergence over naive initializations for $N = M$; however, the effect vanishes for $M > N$. All analyses were performed on an NVIDIA RTX PRO 6000 Blackwell (96GB) GPU.

5.1 Runtime convergence and performance of GTF-DEER

We evaluate the efficacy of GTF-DEER by performing runtime comparisons to the sequential baseline as well as performing ablations on several parameters of the GTF-DEER algorithm. Figure 1**A** shows median runtimes for combined forward+backward passes for both sequential and parallel evaluation using GTF-DEER while training to reconstruct dynamics of the chaotic Lorenz-63 attractor ($N = 3$). While the sequential approach exhibits linear scaling throughout, GTF-DEER enables sublinear scaling with sequence length, leading to a speedup of up to $\mathbf{870}\times$ over sequential evaluation (see Fig. 1**A**, setting $M = 4$, $T = 32{,}768$). However, for large problems, i.e. large $B \times T \times M$, GPU memory bandwidth quickly saturates leading to near linear to constant scaling, a common bottleneck in parallel scan implementations on GPU devices [33, 57]. Nevertheless, for all settings GTF-DEER outperforms naive sequential training in terms of raw runtime. Moreover, single-device saturation does not pose a fundamental limitation: since the scan is associative, it can be chunked across D devices using hierarchical scans [7, 18, 78], reducing work and memory per device by a factor of roughly D. We show in Fig. A1 that this yields further speedups with (near-)linear scaling in the number of devices D on sequences of length $T = 10^6$, far exceeding what could be effectively parallelized on a single device; see Appx. D for details.

Figure 1**B** shows the convergence of the DEER forward pass under increasing values of $\alpha \in [0, 1]$. For GTF-DEER, α does not only control Jacobian norms during the backward pass, addressing the exploding gradient problem in the face of chaos [40, 59], but also controls convergence of the linear recurrence in Eq. (13). If α is too small, the Jacobians will diverge in chaotic settings, and hence the forward pass becomes numerically unstable such that the algorithm suffers its worst-case scaling where the number of Newton iterations needed to converge approaches the length of the sequence itself, $\mathcal{O}(T)$ [30]. This defeats DEER's purpose, as the algorithm then suffers from a runtime complexity of $\mathcal{O}(T \log T)$, which is worse than generating the forward pass sequentially in $\mathcal{O}(T)$ time. Increasing α pushes Jacobian norms below 1 such that the forced model constitutes a stable DS, leading to convergence of GTF-DEER in only $\mathcal{O}(\log T)$ Newton iterations, which guarantees the sublinear average-case runtime of $\mathcal{O}((\log T)^2)$ [29]. In fact, for sufficiently large α, the forward pass even converges optimally in just 2 Newton iterations for $M \leq N$![3] For $M > N$, only a subspace of the latent space of the RNN is forced such that GTF-DEER needs more iterations to converge to the true latent dynamics. For guidance on how to choose α in practice, see Appx. F.3.

Finally, the different curves in Fig. 1**B** show the benefit of readily available forcing targets: Initializing the latent trajectory with estimated forcing signals by pseudo-inversion of the observation model drastically reduces the number of Newton iterations for a given α and $M \leq N$. This specific benefit vanishes when moving to the overdetermined case of $M > N$. This is expected; the RNN dynamics

[3]The reason that two Newton iterations are required instead of just one is that an additional iteration is needed to verify that the previous iteration has indeed converged.

will veer off the estimated forcing signals as GTF is only applied to the zero-error manifold [73]. For more details on the experimental setup underlying Fig. 1 we refer to Appx. I.1.

5.2 GTF-DEER ablations

An important observation is that using diagonal approximation of the Jacobians in Eq. (13) for the forward pass (quasi-DEER, [30]) affects gradients during the backward pass, as the same diagonal Jacobians will be used (see Appx. E for more details). In Fig. 2, we test how Jacobian diagonalization affects GTF-DEER convergence and reconstruction quality. We show loss curves and Newton iterations of shPLRNNs ($M = 5$, $L = 50$) trained on the Lorenz-63 system under two settings: In the 'FO' setting, the shPLRNN is tasked to reconstruct the dynamics from the fully observed system ($N = 3$), while in the more challenging 'PO' setting only the x-component is observed ($N = 1$). We then consider training with diagonalized Jacobians ('quasi') as well as full Jacobians, i.e. standard GTF-DEER, while keeping all other training-related hyperparameters the same. We find that while full-Jacobian training handled both settings with minimal Newton iterations and provided good reconstruction quality ('PO': $D_{\text{stsp}}^{\text{DE}} = (7.5 \pm 6.2) \cdot 10^{-3}$, 'FO': $D_{\text{stsp}} = (8.7 \pm 3.8) \cdot 10^{-3}$), diagonal approximations needed $\approx 100\times$ more Newton iterations for the forward pass to converge, failed to yield convergent loss curves in the more challenging 'PO' setting, and fell behind in reconstruction performance across both settings ('PO + quasi': $D_{\text{stsp}}^{\text{DE}} = (7 \pm 7)$ with $14/20$ runs diverging, 'FO + quasi': $D_{\text{stsp}} = (4.4 \pm 3.5) \cdot 10^{-2}$). This result highlights the role of the Jacobians during the backward pass: Since the diagonal approximations of the Jacobians are used to compute the gradient, they lose information on temporal mixing effects. This prevents the DSR model from learning a faithful embedding of the partially observed dynamics, which is also reflected in the erratic loss curve in Fig 2**A**. Thus, while full Jacobians in theory increase the time demand of each Newton step, this is more than compensated for by the orders of magnitude less iterations and the more stable training.

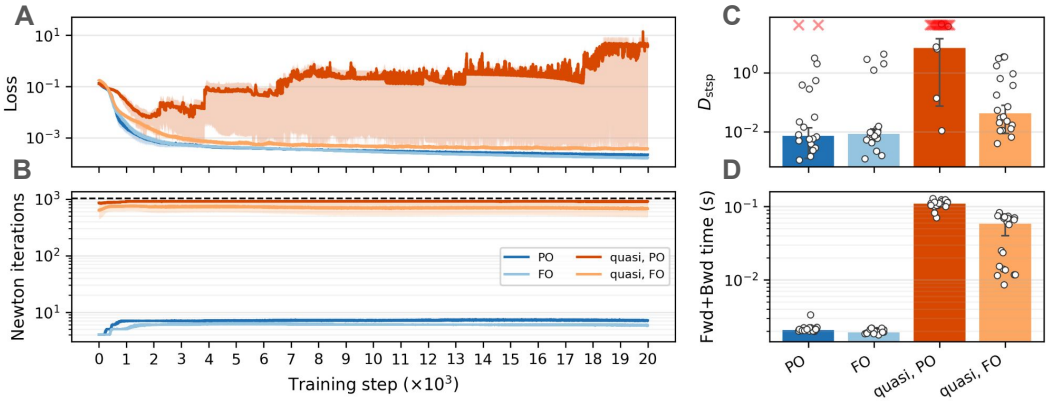


Figure 2: Training dynamics and reconstruction quality of a shPLRNN ($M = 5$, $L = 50$) on the Lorenz-63 system under fully observed (FO, $N = 3$) and partially observed (PO, $N = 1$) conditions. **A**: Training loss curves. **B**: Newton iterations per training step. **C**: State-space divergence D_{stsp} (lower is better); red crosses mark runs where trajectory divergence during generation produced NaNs. **D**: Wall-clock runtime per training step. All reported values are median $\pm$ MAD across 20 runs.

5.3 Reconstructing dynamical systems with long-term dependencies

To demonstrate the advantage of processing long sequences, we generated data from 1) a Lorenz-96 system ($N = 6$) [52] augmented with a sinusoidal forcing term of period 15,000 discrete time steps, thereby introducing an explicit long time scale into the dynamics, and 2) a bursting neuron biophysical model [23] which exhibits long inter-burst-intervals of $> 10^4$ time steps (see Appx. G for further details on data generation). We trained shPLRNNs with $M = 10$ for the Lorenz-96 system and $M = 6$ for the bursting neuron, where both settings share $L = 128$, to reconstruct the underlying dynamics while systematically varying the training sequence length. To control for the fact that longer sequences expose the model to more data per gradient update, we held the product $B \cdot T = 2^{15} = 32{,}768$ fixed. For full experimental details see Appx. I.3.1.

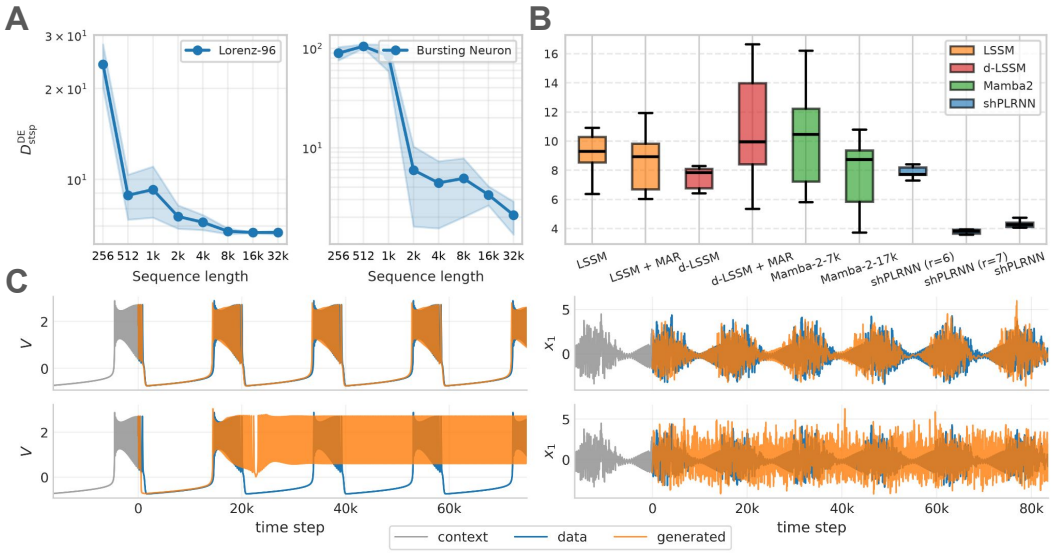


Figure 3: **A**: Long-term measure $D_{\text{stsp}}^{\text{DE}}$ as a function of sequence length for shPLRNNs trained on the forced Lorenz-96 and bursting neuron system. **B**: $D_{\text{stsp}}^{\text{DE}}$ evaluated on the forced Lorenz-96 system for different models (see Fig. A4 for qualitative comparison). Note that even Mamba-2 cannot match the performance of the $r = 7$ shPLRNN trained with GTF-DEER, even when allowed many more parameters. **C**: Example reconstructions for long-sequence GTF-DEER (top) vs. a model trained only on standard-length sequences (bottom) for the bursting neuron (left) and forced Lorenz-96 (right).

Figure 3**A** reports the long-term reconstruction measure $D_{\text{stsp}}^{\text{DE}}$ as a function of sequence length (see Fig. A5 for the RMSE(128)). For both systems, the measure improves with increasing sequence length, confirming that the model benefits substantially from training specifically on longer contexts, while keeping the *total* amount of training data constant. Training at such sequence lengths becomes tractable only through the favorable scaling of GTF-DEER: whereas training on sequences of length $32{,}768$ required on average ≈ 10 minutes for the Lorenz-96 setting on a single NVIDIA RTX PRO 6000 Blackwell GPU, sequential GTF [40] would require approximately 34 hours.

5.4 Linear vs. nonlinear training-time recurrences

To further evaluate the benefits of nonlinear training-time recurrences for DSR, we compare the DSR performance of nonlinear RNNs (Eq. (5)) trained by GTF-DEER with that of LSSMs (Eq. (3)) and Mamba-2-based models ([18], Appx. I.3.2) on two simulated benchmark systems (forced Lorenz-96 and the Deng–Hines model of chaotic and multi-timescale predator–prey dynamics [19, 37]), as well as on real-world data (membrane potential recordings from cortical cells in-vivo [36]). For comparability, we fix shared hyperparameters, such as shPLRNN or MLP hidden layer size L, sequence length T and batch size B. Results of all model comparisons are collected in Tab. A3.

Forced Lorenz-96 system. We trained all models on the forced Lorenz-96 with $T = 81{,}920$ and $B = 1$, and performed several ablations: First, to investigate the effect of the low-rank constraint in the LSSM (Eq. (4)), we also train a low-rank version of the shPLRNN, where $\boldsymbol{W} := \boldsymbol{W}_L \boldsymbol{W}_R$ with $\boldsymbol{W}_L \in \mathbb{R}^{M \times r}$ and $\boldsymbol{W}_R \in \mathbb{R}^{r \times L}$. Second, to test whether the LSSM's shortcomings stem from the restricted diagonal parameterization of $\boldsymbol{A}$, we train it with a dense matrix $\boldsymbol{A} \in \mathbb{R}^{M \times M}$ (d-LSSM).[4] Third, to assess the effect of the MAR used for the shPLRNN, we add the same regularization term to the training of LSSMs, see Appx. F for details. Figure 3**B** summarizes the results of these ablations quantitatively using $D_{\text{stsp}}^{\text{DE}}$, while qualitative example reconstructions are provided in Fig. A4.

When limiting the rank to the number of observed variables, $r = N = 6$, both the LSSMs and the shPLRNN fail to reconstruct the forced Lorenz-96 system and do not properly capture the influence

[4] Note that this does not affect the rank constraint on $\overline{\boldsymbol{W}} = \boldsymbol{U}\boldsymbol{B}$, which is inherent to feeding predictions back through the input layer $\boldsymbol{U}$ (see Sect. 3.1.1).

of the sinusoidal forcing. However, increasing the rank by just 1 to $r = 7$ was already sufficient to significantly improve reconstruction quality for the shPLRNN. In contrast, LSSMs with a dense connectivity matrix do not close the gap to the shPLRNN trained with GTF-DEER. While Mamba-2 is able to avoid the low-rank constraint through its selection mechanism, it performs much worse than the shPLRNN trained by GTF-DEER, a problem that could not be fixed by more than doubling its number of trainable parameters. Hence, the performance gap cannot be explained by model expressivity or parameterization alone, thus highlighting the importance of the training algorithm: Compared to conventional TF, GTF(-DEER) improves auto-regressive roll-outs at test time, since GTF reduces exposure bias by keeping the forcing strength α minimal. Applying similar strategies to address exposure bias in linear training-time recurrence models destroys their amenability to efficient parallelization (see Appx. C).

Deng–Hines model. Results were similar for the partially observed predator–prey system: LSSMs and the Mamba-2 models fail to reproduce the system's chaotic dynamics during autoregressive roll-outs (median $D_{\text{stsp}}^{\text{DE}} = 15$ and $D_{\text{stsp}}^{\text{DE}} = 34.5$, respectively), while shPLRNNs trained with GTF-DEER perform well ($D_{\text{stsp}}^{\text{DE}} = 0.6$), as long as the sequence length T contains enough information on the system's multiple time scales (see also Tab. A3).

Neurophysiological recordings. We also tested performance on long real-world time series, namely whole-cell patch clamp recordings of the membrane potential of single rodent entorhinal cortex neurons in-vivo [36]. These involve slow and irregular state transitions between hyperpolarized "DOWN" and depolarized "UP" states carrying fast spikes, thus multiple timescales from milliseconds to many seconds, as well as considerable noise levels, making them a challenging test case (see Fig. A2). Using GTF-DEER, we manage to train on sequences of length $T = 50,000$ (corresponding to 10 seconds) that include multiple transitions between UP and DOWN states, yielding shPLRNNs that remain stable under autonomous roll-outs. Trained models also capture the statistical long-term behavior (median $D_{\text{stsp}}^{\text{DE}} = 0.68$), which LSSMs ($D_{\text{stsp}}^{\text{DE}} = 6.2$) and Mamba-2 ($D_{\text{stsp}}^{\text{DE}} = 50.3$) fail to reproduce. Especially Mamba-2 struggles with this real-world dataset, where roll-outs collapse to trivial fixed point dynamics reflected in large $D_{\text{stsp}}^{\text{DE}}$ values (despite similar short-term forecast performance, see Tab. A3), confirming the exposure bias problem.

6 Discussion

We introduced GTF-DEER, a parallel-in-time training algorithm that combines the long-sequence scalability of DEER with the chaos-taming properties of Generalized Teacher Forcing (GTF). Through forcing, GTF-DEER turns the RNN during training into a stable DS, improving empirical runtime scaling from the worst-case of $\mathcal{O}(T \log T)$ under unstable dynamics to the average case of $\mathcal{O}((\log T)^2)$ for contractive RNNs [29]. Leveraging fast training on arbitrary sequence lengths, we provided what is to our knowledge the first systematic evidence that DSR benefits from training on such long sequences when the data carry slow timescales. Furthermore, we showed that feeding predictions of linear SSMs (LSSMs) with nonlinear read-out back into the recurrence during autoregressive generation imposes a low-rank constraint. Artificially enforcing this constraint in nonlinear RNNs prevents them from inferring unobserved dynamical variables, imposing the same limitation as in LSSMs. Even when this problem is ameliorated in (L)SSMs by introducing gating or selection mechanisms [18, 33], DSR quality suffers due to the inherent exposure bias of conventional teacher forcing [4, 71], which degrades auto-regressive roll-outs during testing. For modern SSMs, this problem cannot be resolved without trading the linear recurrence for a (locally) nonlinear one, such that parallelization by parallel scan and similar algorithms breaks down (cf. Appx. C, [7, 18, 33, 34, 81]).

Limitations. GTF-DEER inherits DEER's cubic work in the latent dimension M per Newton-iteration, $\mathcal{W}_{\text{full}} = \mathcal{O}(M^3 T)$, and the diagonal (quasi-DEER) approximation that would alleviate this degrades gradients in partially observed settings substantially (Sect. 5.2). This imposes practical limits on the latent dimension M (see Fig. 1**A**). Finally, Proposition 1 strictly holds for $M \leq N$, while for $M > N$ we only find empirical evidence that GTF-DEER's convergence also holds for the settings and problems considered in this work.

Acknowledgments and Disclosure of Funding

This work was supported by individual grants Du 354/14-1 (project no. 437610067, within FOR-5159) and Du 354/15-1 (project no. 502196519) from the German Research Foundation (DFG). We thank Maxim Braunsburger for his contributions to the implementation and evaluation of an earlier version of the GTF-DEER algorithm as part of his B.Sc. thesis under our supervision.

References

[1] Julius Aka, Johannes Brunnemann, Jörg Eiden, Arne Speerforck, and Lars Mikelsons. Balanced neural ODEs: nonlinear model order reduction and koopman operator approximations. In *The Thirteenth International Conference on Learning Representations*, 2025.

[2] Victor M. Martinez Alvarez, Rareş Roşca, and Cristian G. Fălcuţescu. Dynode: Neural ordinary differential equations for dynamics modeling in continuous control. *arXiv preprint arXiv:2009.04278*, 2020.

[3] Omri Azencot, N. Benjamin Erichson, Vanessa Lin, and Michael W. Mahoney. Forecasting Sequential Data using Consistent Koopman Autoencoders. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[4] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[6] Upinder S. Bhalla and Ravi Iyengar. Emergent properties of networks of biological signaling pathways. *Science*, 283(5400):381–387, 1999.

[7] Guy E. Blelloch. Prefix sums and their applications. 1990.

[8] Jonah Botvinick-Greenhouse. Invariant measures for data-driven dynamical system identification: Analysis and application. *arXiv preprint arXiv:2502.05204*, 2025.

[9] Manuel Brenner, Christoph Jürgen Hemmer, Zahra Monfared, and Daniel Durstewitz. Almost-linear rnns yield highly interpretable symbolic codes in dynamical systems reconstruction. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 36829–36868. Curran Associates, Inc., 2024.

[10] Manuel Brenner, Florian Hess, Georgia Koppe, and Daniel Durstewitz. Integrating Multimodal Data for Joint Generative Modeling of Complex Dynamics. In *Proceedings of the 41st International Conference on Machine Learning*, pages 4482–4516. PMLR, July 2024. ISSN: 2640-3498.

[11] Manuel Brenner, Florian Hess, Jonas M. Mikhaeil, Leonard F. Bereska, Zahra Monfared, Po-Chen Kuo, and Daniel Durstewitz. Tractable Dendritic RNNs for Reconstructing Nonlinear Dynamical Systems. In *Proceedings of the 39th International Conference on Machine Learning*, pages 2292–2320. PMLR, June 2022. ISSN: 2640-3498.

[12] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern koopman theory for dynamical systems. *SIAM Review*, 64(2):229–340, 2022.

[13] Steven L. Brunton and J. Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[14] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences USA*, 113(15):3932–3937, 2016.

[15] Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences USA*, 116(45):22445–22451, 2019.

[16] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems 31*, 2018.

[17] Alexandre Cortiella, Kwang-Chun Park, and Alireza Doostan. Sparse identification of nonlinear dynamical systems via reweighted l1-regularized least squares. *Computer Methods in Applied Mechanics and Engineering*, 376:113620, April 2021.

[18] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *Proceedings of Machine Learning Research*, 235:10041–10071, 2024.

[19] Bo Deng. Food chain chaos due to junction-fold point. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 11(3):514–525, 2001.

[20] Bo Deng and Gwendolen Hines. Food chain chaos due to shilnikov's orbit. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 12(3):533–538, 2002.

[21] Kenji Doya. Bifurcations in the learning of recurrent neural networks. In *Proceedings of the 1992 IEEE International Symposium on Circuits and Systems*, 1992.

[22] Daniel Durstewitz. Implications of synaptic biophysics for recurrent network dynamics and active memory. *Neural Networks*, 22(8):1189–1200, 2009.

[23] Daniel Durstewitz and Thomas Gabriel. Dynamical Basis of Irregular Spiking in NMDA-Driven Prefrontal Cortex Neurons. *Cerebral Cortex*, 17(4):894–908, April 2007.

[24] Adam J Eisen, André M Bastos, Jacob A Donoghue, Scott L Brincat, Emery N Brown, Ila R Fiete, and Earl K Miller. Disruption of interareal control during propofol anesthesia. *bioRxiv*, pages 2026–07, 2026.

[25] Niklas Emonds, Evelyn Herberg, Martin Fungisai Gerchen, Marc Pritsch, Joshua Rocha, Vera Zamoscik, Peter Kirsch, Roland Herzog, and Georgia Koppe. A data-driven closed-loop control approach to drive neural state transitions for mechanistic insight. *Human Brain Mapping*, 47(11):e70600, 2026.

[26] Richard J. Field, Endre Koros, and Richard M. Noyes. Oscillations in chemical systems. ii. thorough analysis of temporal oscillation in the bromate-cerium-malonic acid system. *Journal of the American Chemical Society*, 94(25):8649–8664, 1972.

[27] Daniel J. Gauthier, Erik Bollt, Aaron Griffith, and Wendson A. S. Barbosa. Next generation reservoir computing. *Nature Communications*, 12(1):5564, September 2021. Number: 1 Publisher: Nature Publishing Group.

[28] Xavier Gonzalez, E Kelly Buchanan, Hyun Dong Lee, Jerry Weihong Liu, Ke Alexander Wang, David M Zoltowski, Christopher Ré, and Scott W Linderman. A unifying framework for parallelizing sequential models with linear dynamical systems. *arXiv preprint arXiv:2509.21716*, 2025.

[29] Xavier Gonzalez, Leo Kozachkov, David Zoltowski, Kenneth Clarkson, and Scott Linderman. Predictability enables parallelization of nonlinear state space models. In *Annual Conference on Neural Information Processing Systems*, 2025.

[30] Xavier Gonzalez, Andrew Warrington, Jimmy T. Smith, and Scott W. Linderman. Towards scalable and stable parallelization of nonlinear rnns. *Advances in Neural Information Processing Systems*, 37:5817–5849, 2024.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[32] R. B. Govindan, K. Narayanan, and M. S. Gopinathan. On the evidence of deterministic chaos in ecg: Surrogate and predictability analysis. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 8(2):495–502, 1998.

[33] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

[34] Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces, August 2022. arXiv:2111.00396 [cs].

[35] Niclas Alexander Göring, Florian Hess, Manuel Brenner, Zahra Monfared, and Daniel Durstewitz. Out-of-Domain Generalization in Dynamical Systems Reconstruction. In *Proceedings of the 41st International Conference on Machine Learning*, pages 16071–16114. PMLR, July 2024. ISSN: 2640-3498.

[36] Thomas TG Hahn, James M McFarland, Sven Berberich, Bert Sakmann, and Mayank R Mehta. Spontaneous persistent activity in entorhinal cortex modulates cortico-hippocampal interaction in vivo. *Nature neuroscience*, 15(11):1531–1538, 2012.

[37] Alan Hastings and Thomas Powell. Chaos in a three-species food chain. *Ecology*, 72(3):896–903, 1991.

[38] Christoph Jürgen Hemmer and Daniel Durstewitz. True zero-shot inference of dynamical systems preserving long-term statistics. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[39] John R. Hershey and Peder A. Olsen. Approximating the kullback leibler divergence between gaussian mixture models. *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, 4:IV–317–IV–320, 2007.

[40] 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*, pages 13017–13049. PMLR, July 2023. ISSN: 2640-3498.

[41] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[42] Eugene M. Izhikevich. *Dynamical systems in neuroscience: the geometry of excitability and bursting*. Computational neuroscience. MIT Press, Cambridge, Mass, 2007. OCLC: ocm65400606.

[43] Eugenia Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press, 2003.

[44] Daniel Karlsson and Olle Svanström. Modelling Dynamical Systems Using Neural Ordinary Differential Equations, 2019.

[45] Anatole Katok, A. B. Katok, and Boris Hasselblatt. *Introduction to the Modern Theory of Dynamical Systems*. Cambridge University Press, 1995. Google-Books-ID: 9nL7ZX8Djp4C.

[46] Joon-Hyuk Ko, Hankyul Koh, Nojun Park, and Wonho Jhe. Homotopy-based training of neuralodes for accurate dynamics discovery. *Advances in Neural Information Processing Systems*, 36:64725–64752, 2023.

[47] Georgia Koppe, Hazem Toutounji, Peter Kirsch, Stefanie Lis, and Daniel Durstewitz. Identifying nonlinear dynamical systems via generative recurrent neural networks with applications to fMRI. *PLOS Computational Biology*, 15(8):e1007263, 2019.

[48] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020.

[49] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations.

[50] Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing non-linear sequential models over the sequence length. In *International Conference on Learning Representations*, 2024.

[51] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[52] Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.

[53] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[54] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. Number: 3 Publisher: Nature Publishing Group.

[55] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nat Commun*, 9(1):4950, December 2018. arXiv: 1712.09707.

[56] Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018.

[57] Duane Merrill and Michael Garland. Single-pass parallel prefix scan with decoupled look-back. *NVIDIA, Tech. Rep. NVR-2016-002*, 2016.

[58] Daniel A. Messenger and David M. Bortz. Weak SINDy: Galerkin-Based Data-Driven Model Selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497, January 2021. Publisher: Society for Industrial and Applied Mathematics.

[59] Jonas Mikhaeil, Zahra Monfared, and Daniel Durstewitz. On the difficulty of learning chaotic dynamics with RNNs. *Advances in Neural Information Processing Systems*, 35:11297–11312, December 2022.

[60] Nicola Muca Cirone, Antonio Orvieto, Benjamin Walker, Cristopher Salvi, and Terry Lyons. Theoretical foundations of deep selective state-space models. *Advances in Neural Information Processing Systems*, 37:127226–127272, 2024.

[61] Ilan Naiman and Omri Azencot. A Koopman Approach to Understanding Sequence Neural Models. *arXiv:2102.07824 [cs, math]*, October 2021. arXiv: 2102.07824.

[62] Antonio Orvieto, Soham De, Caglar Gulcehre, Razvan Pascanu, and Samuel L Smith. Universality of linear recurrences followed by non-linear projections: Finite-width guarantees and benefits of complex eigenvalues. In *International Conference on Machine Learning*, pages 38837–38863. PMLR, 2024.

[63] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pages 26670–26698. PMLR, 2023.

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

[65] Dhruvit Patel and Edward Ott. Using machine learning to anticipate tipping points and extrapolate to post-tipping dynamics of non-stationary dynamical systems. *Chaos (Woodbury, N.Y.)*, 33(2):023143, February 2023.

[66] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. *Phys. Rev. Lett.*, 120(2):024102, 2018.

[67] Jaideep Pathak, Zhixin Lu, Brian R. Hunt, Michelle Girvan, and Edward Ott. Using Machine Learning to Replicate Chaotic Attractors and Calculate Lyapunov Exponents from Data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, December 2017. arXiv: 1710.07313.

[68] 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: An Interdisciplinary Journal of Nonlinear Science*, 33(10), 2023.

[69] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019.

[70] Yuval Ran-Milo, Eden Lumbroso, Edo Cohen-Karlik, Raja Giryes, Amir Globerson, and Nadav Cohen. Provable benefits of complex parameterizations for structured state space models. *Advances in Neural Information Processing Systems*, 37:115906–115939, 2024.

[71] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[72] T Konstantin Rusch, Siddhartha Mishra, N Benjamin Erichson, and Michael W Mahoney. Long expressive memory for sequence modeling. In *International Conference on Learning Representations*, 2022.

[73] A Erdem Sağtekin, Colin Bredenberg, and Cristina Savin. Error forcing in recurrent neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[74] Tim Sauer, James A. Yorke, and Martin Casdagli. Embedology. *Journal of statistical Physics*, 65(3):579–616, 1991.

[75] Julian D. Schiller, Malte Heinrich, Victor G. Lopez, and Matthias A. Müller. Tuning the burn-in phase in training recurrent neural networks improves their performance. In *The Fourteenth International Conference on Learning Representations*, 2026.

[76] Dominik Schmidt, Georgia Koppe, Zahra Monfared, Max Beutelspacher, and Daniel Durstewitz. Identifying nonlinear dynamical systems with multiple time scales and long-range dependencies. In *International Conference on Learning Representations*, 2021.

[77] Dominik Schmidt, Georgia Koppe, Zahra Monfared, Max Beutelspacher, and Daniel Durstewitz. Identifying nonlinear dynamical systems with multiple time scales and long-range dependencies. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

[78] Shubhabrata Sengupta, Mark Harris, and Michael Garland. Efficient parallel scan algorithms for gpus. Technical Report NVR-2008-003, NVIDIA Corporation, December 2008.

[79] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[80] B. Sivakumar. Chaos theory in geophysics: past, present and future. *Chaos, Solitons & Fractals*, 19(2):441–462, 2004.

[81] Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. International Conference on Learning Representations (ICLR), 2023.

[82] Steven H. Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. Chapman and Hall/CRC, 2024.

[83] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980*, volume 898, pages 366–381. Springer, 1981.

[84] Peter Turchin and Andrew D. Taylor. Complex dynamics in ecological time series. *Ecology*, 73(1):289–305, 1992.

[85] Eli Tziperman, Harvey Scher, Stephen E. Zebiak, and Mark A. Cane. Controlling spatiotemporal chaos in a realistic el niño prediction model. *Phys. Rev. Lett.*, 79:1034–1037, Aug 1997.

[86] Pietro Verzelli, Cesare Alippi, and Lorenzo Livi. Learn to synchronize, synchronize to learn. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(8):083119, August 2021.

[87] Pantelis R. Vlachas, Wonmin Byeon, Zhong Y. Wan, Themistoklis P. Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proc. R. Soc. A.*, 474(2213):20170844, 2018.

[88] Pantelis R Vlachas and Petros Koumoutsakos. Learning on predictions: Fusing training and autoregressive inference for long-term spatiotemporal forecasts. *Physica D: Nonlinear Phenomena*, 470:134371, 2024.

[89] Pantelis R. Vlachas, Jaideep Pathak, Brian R. Hunt, Themistoklis P. Sapsis, Michelle Girvan, Edward Ott, and Petros Koumoutsakos. Backpropagation Algorithms and Reservoir Computing in Recurrent Neural Networks for the Forecasting of Complex Spatiotemporal Dynamics. *arXiv:1910.05266 [physics]*, February 2020. arXiv: 1910.05266.

[90] Eric Volkmann, Alena Brändle, Daniel Durstewitz, and Georgia Koppe. A scalable generative model for dynamical system reconstruction from neuroimaging data. *Advances in Neural Information Processing Systems*, 37:80328–80362, 2024.

[91] Rui Wang, Yihe Dong, Sercan Ö Arik, and Rose Yu. Koopman Neural Forecaster for Time Series with Temporal Distribution Shifts, October 2022. arXiv:2210.03675 [cs, stat].

[92] Paul J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[93] Nicolas Zucchet and Antonio Orvieto. Recurrent neural networks: vanishing and exploding gradients are not the end of the story. *Advances in Neural Information Processing Systems*, 37:139402–139443, 2024.

Appendix

A Absence of chaos-induced exploding gradients in linear SSMs

The gradient propagation properties of linear SSMs are well-established in prior work [59, 63, 93] and we briefly recap them here for completeness. Consider a discrete-time linear SSM with time-invariant dynamics:

$$\boldsymbol{z}_t = \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{v}_t, \tag{17}$$

where $\boldsymbol{z}_t \in \mathbb{R}^M$ is the latent state, $\boldsymbol{A} \in \mathbb{R}^{M\times M}$ is the state transition matrix, and $\boldsymbol{v}_t \in \mathbb{R}^M$ is an extrinsic input. Differentiating the recurrence yields a constant Jacobian $\boldsymbol{J}_i = \partial \boldsymbol{z}_i / \partial \boldsymbol{z}_{i-1} = \boldsymbol{A}$, so the BPTT chain-rule product collapses to

$$\frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{z}_r} = \prod_{i=r+1}^{t} \boldsymbol{J}_i = \boldsymbol{A}^{t-r}, \qquad \left\| \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{z}_r} \right\| \leq \|\boldsymbol{A}\|^{t-r}. \tag{18}$$

Gradient propagation is therefore independent of the state trajectory $\{\boldsymbol{z}_i\}$ and inputs $\{\boldsymbol{v}_i\}$, and is fully determined by the spectrum of $\boldsymbol{A}$. As a consequence, linear SSMs cannot exhibit chaos-induced gradient explosion: constraining the eigenvalues of $\boldsymbol{A}$ (typically near the unit disk) is sufficient to control gradient magnitudes.

B Proof of Proposition 1: GTF-DEER convergence

Proof. Since $M \leq N$ and the observation matrix $\boldsymbol{B}$ has full column rank, $\boldsymbol{B}^{+}\boldsymbol{B} = I_M$, and by Eq. (8), $\boldsymbol{P}_\alpha = (1-\alpha)\, I_M$. The Jacobian decomposition simplifies to Eq. (7),

$$\tilde{\boldsymbol{J}}_t := \tilde{\boldsymbol{J}}(\boldsymbol{z}_{t-1}) := \boldsymbol{J}_{F\circ\delta_\alpha}(\boldsymbol{z}_{t-1}) = (1-\alpha)\, \boldsymbol{J}_F(\tilde{\boldsymbol{z}}_{t-1}). \tag{19}$$

Taking spectral norms leads to:

$$(1-\alpha)\, \|\boldsymbol{J}_F(\tilde{\boldsymbol{z}}_{t-1})\|_2 \leq (1-\alpha)\, \tilde{\sigma}_{\max} = \rho. \tag{20}$$

Since $\alpha > \alpha^* = 1 - 1/\tilde{\sigma}_{\max}$, we have $(1-\alpha) < 1/\tilde{\sigma}_{\max}$ and hence $\rho < 1$.

For any product of k consecutive forced Jacobians:

$$\left\| \tilde{\boldsymbol{J}}_t \tilde{\boldsymbol{J}}_{t-1} \cdots \tilde{\boldsymbol{J}}_{t-k+1} \right\|_2 \leq \prod_{i=0}^{k-1} \left\| \tilde{\boldsymbol{J}}_{t-i} \right\|_2 \leq \rho^k \tag{21}$$

Then by Def. 1, we have

$$\lambda(\boldsymbol{z}) = \lim_{T\to\infty} \frac{1}{T} \log \left\| \prod_{i=0}^{T-1} \tilde{\boldsymbol{J}}_{T-i} \right\|_2 \leq \lim_{T\to\infty} \frac{1}{T} \log \rho^T = \log \rho < 0. \tag{22}$$

Thus the forced system is *contracting*. □

C Scheduled sampling breaks the linearity of LSSMs

Conventional teacher forcing [31] feeds the ground-truth observation $\boldsymbol{x}_{t-1}$ into the latent recurrence

$$\boldsymbol{z}_t = \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{U}\boldsymbol{x}_{t-1} + \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{h} \tag{23}$$

at every step. While the following holds for any forcing strategy that re-introduces a feedback from model-predicted observations $\hat{\boldsymbol{x}}$ into the latent recurrence, we will exemplify the problem using the well established method of scheduled sampling [4]. Scheduled sampling switches between teacher forcing and free-running generation during training by replacing $\boldsymbol{x}_{t-1}$ with the model's own prediction $\hat{\boldsymbol{x}}_{t-1}$ with a scheduled probability $\epsilon_t \in [0,1]$,

$$\tilde{\boldsymbol{x}}_{t-1} = \begin{cases} \boldsymbol{x}_{t-1} & \text{with probability } 1-\epsilon_t, \\ \hat{\boldsymbol{x}}_{t-1} & \text{with probability } \epsilon_t, \end{cases} \tag{24}$$

so that the latent recurrence used during training reads

$$\boldsymbol{z}_t = \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{U}\tilde{\boldsymbol{x}}_{t-1} + \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{h}. \tag{25}$$

Whenever the prediction branch is taken, substituting $\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{B}\phi(\boldsymbol{V}\boldsymbol{z}_{t-1} + \boldsymbol{b})$ from Eq. (3) into Eq. (25) yields

$$\boldsymbol{z}_t = \boldsymbol{A}\boldsymbol{z}_{t-1} + \overline{\boldsymbol{W}}\,\phi(\boldsymbol{V}\boldsymbol{z}_{t-1} + \boldsymbol{b}) + \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{h}, \tag{26}$$

with $\overline{\boldsymbol{W}} = \boldsymbol{U}\boldsymbol{B}$, i.e. exactly the test-time recurrence of Eq. (4). The transition $\boldsymbol{z}_{t-1} \mapsto \boldsymbol{z}_t$ is therefore no longer linear (affine) in $\boldsymbol{z}_{t-1}$ but contains the nonlinearity ϕ.

Incompatibility with parallel scan Evaluation of Eq. (3) via parallel scan [7, 56, 81] hinges on each transition being representable as an affine map $\boldsymbol{z}_t = \boldsymbol{A}_t\boldsymbol{z}_{t-1} + \boldsymbol{b}_t$, identified with the pair $(\boldsymbol{A}_t, \boldsymbol{b}_t)$. Composition of two such maps,

$$(\boldsymbol{A}_2, \boldsymbol{b}_2) \circ (\boldsymbol{A}_1, \boldsymbol{b}_1) = \big(\boldsymbol{A}_2\boldsymbol{A}_1,\ \boldsymbol{A}_2\boldsymbol{b}_1 + \boldsymbol{b}_2\big), \tag{27}$$

is associative and again affine, such that partial compositions over disjoint chunks of the training sequence can be combined in any order, enabling an $\mathcal{O}(\log T)$-depth reduction. In the standard LSSM recurrence the input term $\boldsymbol{U}\boldsymbol{x}_{t-1}$ depends only on observed data and contributes solely to $\boldsymbol{b}_t$, leaving the map affine in the latent state.

Once $\boldsymbol{x}_{t-1}$ is replaced by $\hat{\boldsymbol{x}}_{t-1}$, the input becomes a nonlinear function of $\boldsymbol{z}_{t-1}$ and the transition is no longer of this affine form. The composition of two such nonlinear transitions cannot in general be represented by a fixed, finite-parameter operator that is *independent* of the latent state, such that the recurrence cannot be reduced to an associative scan. Training must instead unroll Eq. (26) sequentially, bringing it back to the $\mathcal{O}(T)$ bottleneck.

D Scaling to multiple GPUs

As discussed in Sect. 5.1, processing very long sequences can quickly saturate a single device because each Newton iteration employs a resource-intensive parallel scan. Assuming P effective processors on one GPU, its runtime can be bounded as

$$R_1(T) \in \mathcal{O}\left(\frac{T}{P} + \log T\right). \tag{28}$$

With sufficient parallel resources, $P \gtrsim T$, the work term becomes negligible and the theoretical logarithmic depth $\mathcal{O}(\log T)$ is recovered. In practice, however, $T > P$ requires the device to execute the scan operations in multiple waves (sequentially), such that the T/P term can dominate. To alleviate this performance ceiling, we can distribute the scan across $D > 1$ devices using a hierarchical or chunked-scan strategy [7, 18, 78]. The sequence is partitioned into D contiguous chunks of length $T_{\text{sub}} = \left\lceil \frac{T}{D} \right\rceil$. Each device first computes the prefix scan within its local chunk. The resulting D chunk-level transformations (see right-hand side of Eq. (27)) are then exchanged between devices and combined using an associative scan of depth $\mathcal{O}(\log D)$. Finally, the appropriate prefix transformation is applied to each local chunk. Again assuming P effective processors per device, the resulting runtime is

$$R_D(T) \in \mathcal{O}\left(\frac{T}{DP} + \log T_{\text{sub}} + \log D + C_{\text{comm}}(D)\right), \tag{29}$$

where $C_{\text{comm}}(D)$ accounts for communication overhead such as exchanging the chunk-level transformations and synchronizing the devices. Since $\log T_{\text{sub}} + \log D \simeq \log T$, this simplifies to

$$R_D(T) \in \mathcal{O}\left(\frac{T}{DP} + \log T + C_{\text{comm}}(D)\right), \tag{30}$$

Thus, distributing the scan reduces the work and memory per device by approximately a factor of D. This in turn leads to (near-)linear speedups when the work $T/(DP)$ dominates both the logarithmic depth and communication overhead.

To demonstrate this speedup, we trained a shPLRNN ($M = 20, L = 128$) on the Lorenz-63 system with a sequence length of $T = 1{,}000{,}000$ and distributed the work over 1, 2 or 4 NVIDIA RTX PRO 6000 Blackwell (96GB) GPUs while measuring summed forward and backward pass times. The forcing α was fixed at $\alpha = 0.15$ and all settings were compared to the sequential baseline. The results are summarized in Fig. A1.

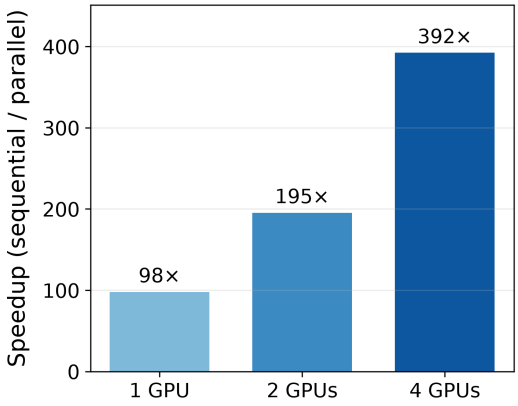


Figure A1: Speedup of distributing *very long* sequences across multiple devices. A shPLRNN ($M = 20, L = 128$) was trained on a sequence of length $T = 10^6 = 1{,}000{,}000$ using GTF-DEER. In this example, distributing the work to four devices leads to a $4\times$ speedup over using a single device, reducing combined average forward solve and gradient computation times from 209 to 52 milliseconds.

E Gradients

E.1 Derivation of GTF-DEER gradient formulas

We define the per-timestep MSE loss function (excluding regularization terms) $\ell : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$,

$$\ell(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{N} \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2 , \tag{31}$$

and the total loss, $\mathcal{L} : (\mathbb{R}^N)^T \times (\mathbb{R}^N)^T \to \mathbb{R}$, for the full time series,

$$\mathcal{L}(\boldsymbol{x}_{1:T}, \hat{\boldsymbol{x}}_{1:T}) = \frac{1}{T} \sum_{t=1}^{T} \ell(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t) = \frac{1}{NT} \sum_{t=1}^{T} \|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\|_2^2 , \tag{32}$$

where in the model (1), $\hat{\boldsymbol{x}}_t = \boldsymbol{G}_\psi(\boldsymbol{z}_t)$ are the observations associated with the latent states; we consider $\boldsymbol{x}_{1:T} \in \mathbb{R}^{NT} \cong (\mathbb{R}^N)^T$ as a stacked vector; and in practice truncate a warm-up period $0 < T_w < T$ and only supply vectors $\boldsymbol{x}, \hat{\boldsymbol{x}}$ of size $N\tilde{T}$, $\tilde{T} \coloneqq T - T_w$, cf. Sect. 4.2 and Eq. (95).

To optimize the parameters $\boldsymbol{\theta}$ of the RNN $\tilde{F}_{\boldsymbol{\theta},\alpha} = F_{\boldsymbol{\theta}} \circ \delta_{\alpha,\boldsymbol{B}}$, we need to compute the gradient of the loss function,

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \theta} \tag{33}$$

for all elements θ (for notational simplicity, we choose not to write any index) in the abstract parameter vector $\boldsymbol{\theta}$ which comprises all matrix weights.

In the DEER forward pass, we use the residual

$$\boldsymbol{r}_{\boldsymbol{\theta},\alpha}(\boldsymbol{z}_{t-1}, \boldsymbol{z}_t) = \boldsymbol{z}_t - \tilde{F}_{\boldsymbol{\theta},\alpha}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t-1}) \in \mathbb{R}^M, \tag{34}$$

$$\boldsymbol{R}_{\boldsymbol{\theta},\alpha}(\boldsymbol{z}_{1:T}) = (\boldsymbol{r}_\alpha(\boldsymbol{z}_0, \boldsymbol{z}_1), \dots, \boldsymbol{r}_\alpha(\boldsymbol{z}_{T-1}, \boldsymbol{z}_T)) \in \mathbb{R}^{MT}, \tag{35}$$

where $\boldsymbol{z}_0$ is a fixed initial state (which is not predicted), to obtain the predicted time series $\boldsymbol{z}_{1:T} \in \mathbb{R}^{MT} \cong (\mathbb{R}^M)^T$. At the end of each forward pass,

$$\boldsymbol{R}_\alpha(\boldsymbol{z}_{1:T}, \boldsymbol{\theta}) \coloneqq \boldsymbol{R}_{\boldsymbol{\theta},\alpha}(\boldsymbol{z}_{1:T}) = \boldsymbol{0} \in \mathbb{R}^{MT}, \tag{36}$$

so by the implicit function theorem,

$$\frac{\partial \boldsymbol{R}_\alpha}{\partial \boldsymbol{z}_{1:T}} \frac{\partial \boldsymbol{z}_{1:T}}{\partial \theta} + \frac{\partial \boldsymbol{R}_\alpha}{\partial \theta} = \boldsymbol{0} \in \mathbb{R}^{MT}. \tag{37}$$

Rearranging this equation gives us

$$\frac{\partial \boldsymbol{z}_{1:T}}{\partial \theta} = -\left(\frac{\partial \boldsymbol{R}_\alpha}{\partial \boldsymbol{z}_{1:T}}\right)^{-1} \frac{\partial \boldsymbol{R}_\alpha}{\partial \theta}. \tag{38}$$

We insert this into the gradient of the loss:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \underbrace{-\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_{1:T}}\left(\frac{\partial \boldsymbol{R}_\alpha}{\partial \boldsymbol{z}_{1:T}}\right)^{-1}}_{=:\boldsymbol{\mathcal{V}}^\top \in \mathbb{R}^{1\times MT}} \frac{\partial \boldsymbol{R}_\alpha}{\partial \theta}, \tag{39}$$

where we write $\boldsymbol{\mathcal{V}} = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_T) \in \mathbb{R}^{MT} \cong (\mathbb{R}^M)^T$. Rather than explicitly inverting the huge Jacobian matrix $\frac{\partial \boldsymbol{R}_\alpha}{\partial \boldsymbol{z}_{1:T}}$, we instead solve the adjoint problem

$$\boldsymbol{\mathcal{V}}^\top \frac{\partial \boldsymbol{R}_\alpha}{\partial \boldsymbol{z}_{1:T}} = -\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_{1:T}}. \tag{40}$$

To make this more explicit, we use

$$\frac{\partial \boldsymbol{r}_{\alpha,\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}, \boldsymbol{z}_t)}{\partial \boldsymbol{z}_{t-1}} = -\frac{\partial \tilde{F}_{\boldsymbol{\theta},\alpha}}{\partial \boldsymbol{z}}(\boldsymbol{z}_{t-1}) =: -\boldsymbol{J}_{\tilde{F}}(\boldsymbol{z}_{t-1}) =: -\tilde{\boldsymbol{J}}_t \in \mathbb{R}^{M\times M}, \tag{41}$$

$$\frac{\partial \boldsymbol{r}_{\alpha,\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}, \boldsymbol{z}_t)}{\partial \boldsymbol{z}_t} = \boldsymbol{I}_M, \tag{42}$$

and write

$$\frac{\partial \boldsymbol{R}_\alpha}{\partial \boldsymbol{z}_{1:T}} = \begin{pmatrix} \boldsymbol{I}_M & 0 & \cdots & \cdots & 0 \\ -\tilde{\boldsymbol{J}}_2 & \boldsymbol{I}_M & 0 & \cdots & 0 \\ 0 & -\tilde{\boldsymbol{J}}_3 & \boldsymbol{I}_M & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & -\tilde{\boldsymbol{J}}_T & \boldsymbol{I}_M \end{pmatrix} \tag{43}$$

and

$$\frac{\partial \boldsymbol{R}_\alpha}{\partial \theta} = \frac{\partial \boldsymbol{R}_\alpha}{\partial \tilde{F}_{\boldsymbol{\theta},\alpha}} \frac{\partial \tilde{F}_{\boldsymbol{\theta},\alpha}}{\partial \theta} = -\frac{\partial \tilde{F}_{\boldsymbol{\theta},\alpha}}{\partial \theta}. \tag{44}$$

The block-bidiagonal structure of the matrix turns (40) into a (backward) recursion in t:

$$\boldsymbol{v}_T^\top = -\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_T}, \qquad \boldsymbol{v}_{t-1}^\top = \boldsymbol{v}_t^\top \boldsymbol{J}_{\tilde{F}}(\boldsymbol{z}_{t-1}) - \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_{t-1}}, \qquad t = T, \dots, 2 \tag{45}$$

where, substituting in the observation model, $\hat{\boldsymbol{x}} = G_\psi(\boldsymbol{z})$,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_{t-1}} = \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_{t-1}} \frac{\partial G_\psi}{\partial \boldsymbol{z}_{t-1}} = \frac{2}{NT}(\hat{\boldsymbol{x}}_{t-1} - \boldsymbol{x}_{t-1})^\top \frac{\partial G_\psi}{\partial \boldsymbol{z}_{t-1}}. \tag{46}$$

In the same way as in the forward pass, this recursion can be solved in parallel via an associative scan. Unrolled in time, this is

$$\boldsymbol{v}_t^\top = -\sum_{\tau=t}^{T} \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_\tau} \tilde{\boldsymbol{J}}_\tau \tilde{\boldsymbol{J}}_{\tau-1} \dots \tilde{\boldsymbol{J}}_{t+1}, \tag{47}$$

where the $\tau = t$ case is to be read as the empty product of Jacobians and hence the identity matrix, and $\tau = t+1$ as a single factor of $\tilde{\boldsymbol{J}}_{t+1}$. Note that for quasi-DEER, the Jacobians $\boldsymbol{J}_{\tilde{F}}$ in (45) are replaced by a diagonal approximation and the equation becomes

$$\boldsymbol{v}_T^\top = -\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_T}, \qquad \boldsymbol{v}_{t-1}^\top = \left(\boldsymbol{v}_t \odot \operatorname{diag}\left(\boldsymbol{J}_{\tilde{F}}(\boldsymbol{z}_{t-1})\right)\right)^\top - \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_{t-1}} \tag{48}$$

with the element-wise product $\odot$.

In summary, the loss gradients are therefore calculated simply by solving the recursion (45) using the final latent trajectory and observations computed via the forward pass, and inserting the result into

$$\frac{\partial \mathcal{L}}{\partial \theta} = -\boldsymbol{\mathcal{V}}^\top \frac{\partial \tilde{F}_{\boldsymbol{\theta},\alpha}}{\partial \theta}. \tag{49}$$

E.2 shPLRNN gradients

We now apply the formulas derived in the previous subsection to the shPLRNN model, Eq. (5). The forced recurrence, Eq. (6), with forcing as in Eq. (8), takes the explicit form

$$\tilde{F}_{\boldsymbol{\theta},\alpha}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t-1}) = \boldsymbol{A}(\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}) + \\ + \boldsymbol{W}\,\mathrm{ReLU}(\boldsymbol{V}(\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}) + \boldsymbol{b}) + \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{h}. \quad (50)$$

The forced Jacobian is

$$\boldsymbol{J}_{\tilde{F}}(\boldsymbol{z}_{t-1}) = \left(\boldsymbol{A} + \boldsymbol{W}\,\mathrm{diag}\left(\Theta(\boldsymbol{V}(\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}) + \boldsymbol{b})\right)\boldsymbol{V}\right)\boldsymbol{P}_\alpha, \quad (51)$$

where the Heaviside step function Θ appears as the derivative of the ReLU function and is applied element-wise.

Inserting the observation model, $\hat{\boldsymbol{x}}_t = \boldsymbol{B}\boldsymbol{z}_t$, into Eq. (46) gives

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_t} = \frac{2}{NT}(\boldsymbol{B}\boldsymbol{z}_t - \boldsymbol{x}_t)^\top \boldsymbol{B}. \quad (52)$$

Combined, the previous two formulae, inserted into Eq. (45), give the explicit adjoint recurrence

$$\boldsymbol{v}_T^\top = -\frac{2}{NT}(\boldsymbol{B}\boldsymbol{z}_T - \boldsymbol{x}_T)^\top \boldsymbol{B} \quad (53)$$

$$\boldsymbol{v}_{t-1}^\top = \left[\boldsymbol{v}_t^\top \boldsymbol{A} + \left(\Theta(\boldsymbol{V}(\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}) + \boldsymbol{b})^\top \odot (\boldsymbol{v}_t^\top \boldsymbol{W})\right)\boldsymbol{V}\right]\boldsymbol{P}_\alpha \quad (54)$$

$$- \frac{2}{NT}(\boldsymbol{B}\boldsymbol{z}_{t-1} - \boldsymbol{x}_{t-1})^\top \boldsymbol{B}, \qquad t = T, \ldots, 2. \quad (55)$$

The unrolled expression, Eq. (47), becomes

$$\boldsymbol{v}_t^\top = -\frac{2}{NT}\sum_{\tau=t}^{T}(\boldsymbol{B}\boldsymbol{z}_\tau - \boldsymbol{x}_\tau)^\top \boldsymbol{B}\tilde{\boldsymbol{J}}_\tau \tilde{\boldsymbol{J}}_{\tau-1} \ldots \tilde{\boldsymbol{J}}_{t+1}. \quad (56)$$

Taking the parameter derivatives of Eq. (50) and writing out the inner product with $\boldsymbol{\mathcal{V}} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_T)$ (cf. Eq. (49)) as a sum, we obtain the expressions

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{A}} = -\sum_{t=1}^{T}\left[\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}\right]\boldsymbol{v}_t^\top \quad (57)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = -\sum_{t=1}^{T}\mathrm{ReLU}\Big(\boldsymbol{V}\left[\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}\right] + \boldsymbol{b}\Big)\boldsymbol{v}_t^\top \quad (58)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{V}} = -\sum_{t=1}^{T}\left[\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}\right]\Big(\Theta\big(\boldsymbol{V}\left[\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}\right] + \boldsymbol{b}\big) \odot (\boldsymbol{W}^\top \boldsymbol{v}_t)\Big)^\top \quad (59)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}} = -\sum_{t=1}^{T}\Big(\Theta\big(\boldsymbol{V}\left[\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}\right] + \boldsymbol{b}\big) \odot (\boldsymbol{W}^\top \boldsymbol{v}_t)\Big)^\top \quad (60)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{C}} = -\sum_{t=1}^{T}\boldsymbol{s}_t\,\boldsymbol{v}_t^\top \quad (61)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}} = -\sum_{t=1}^{T}\boldsymbol{v}_t^\top. \quad (62)$$

With diagonal parameterization, $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{a})$, $\boldsymbol{a} = \tanh(\bar{\boldsymbol{a}})$, $\bar{\boldsymbol{a}} \in \mathbb{R}^M$, Eq. (57) becomes

$$\frac{\partial \mathcal{L}}{\partial \bar{\boldsymbol{a}}} = -\sum_{t=1}^{T}\left(\boldsymbol{v}_t \odot \left(\boldsymbol{P}_\alpha \boldsymbol{z}_{t-1} + \alpha \boldsymbol{B}^+ \boldsymbol{x}_{t-1}\right) \odot \left(\boldsymbol{1} - \boldsymbol{a} \odot \boldsymbol{a}\right)\right)^\top. \quad (63)$$

Moreover,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{B}} &= \frac{2}{NT}\sum_{t=1}^{T} \boldsymbol{z}_t(\boldsymbol{B}\boldsymbol{z}_t - \boldsymbol{x}_t)^\top + \sum_{t=2}^{T} \alpha_{t-1}\, \boldsymbol{z}_{t-1}\boldsymbol{v}_t^\top \boldsymbol{J}_F(\boldsymbol{q}_{t-1})\, \boldsymbol{B}^+ \\ &+ \sum_{t=1}^{T} \Big[\boldsymbol{B}^+ \boldsymbol{y}_{t-1}\boldsymbol{v}_t^\top \boldsymbol{J}_F(\boldsymbol{q}_{t-1})\boldsymbol{B}^+ - \boldsymbol{B}^+(\boldsymbol{B}^+)^\top \boldsymbol{J}_F(\boldsymbol{q}_{t-1})^\top \boldsymbol{v}_t\, \boldsymbol{y}_{t-1}^\top (\boldsymbol{I}_N - \boldsymbol{B}\boldsymbol{B}^+) \\ &\qquad - (\boldsymbol{I}_M - \boldsymbol{B}^+\boldsymbol{B})\, \boldsymbol{J}_F(\boldsymbol{q}_{t-1})^\top \boldsymbol{v}_t\, \boldsymbol{y}_{t-1}^\top (\boldsymbol{B}^+)^\top \boldsymbol{B}^+ \Big],\end{aligned} \tag{64}$$

where we introduced $\boldsymbol{q}_{t-1} = \boldsymbol{P}_{\alpha_{t-1}}\boldsymbol{z}_{t-1} + \alpha_{t-1}\boldsymbol{B}^+\boldsymbol{x}_{t-1}$ for compactness. Note that all derivatives are in numerator layout, so that $\partial\mathcal{L}/\partial\boldsymbol{W}$ has the transposed shape of $\boldsymbol{W}$, and for parameter updates such as $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta\nabla_{\boldsymbol{W}}\mathcal{L}$, we need to take the transpose, $\nabla_{\boldsymbol{W}}\mathcal{L} = (\partial\mathcal{L}/\partial\boldsymbol{W})^\top$.

Taking warm-up into account, normalization factors would be replaced by $2/N\tilde{T}$ and loss terms would be zero for $t \leq T_w$. Moreover, the forcing α in recursion formulas would be t-dependent, $\boldsymbol{P}_{\alpha_{t-1}}\boldsymbol{z}_{t-1} + \alpha_{t-1}\boldsymbol{B}^+\boldsymbol{x}_{t-1}$, with $\alpha_t = 1$ during warmup.

In practical implementations, all of these gradients will be computed internally by automatic differentiation.

E.3 LSSM gradients

We now state the gradient expressions for the LSSM, Eq. (3). The same general formulae as for the shPLRNN apply for this model class, too. The training-time recurrence is linear in $\boldsymbol{z}$ and does not depend on the observation parameters $\boldsymbol{\psi} = \{\boldsymbol{B}, \boldsymbol{V}, \boldsymbol{b}\}$,

$$F_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t-1}, \boldsymbol{s}_t) = \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{U}\boldsymbol{x}_{t-1} + \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{h}, \qquad \boldsymbol{\theta} = \{\boldsymbol{A}, \boldsymbol{U}, \boldsymbol{C}, \boldsymbol{h}\}, \tag{65}$$

so the Jacobian is constant, $\tilde{\boldsymbol{J}} = \boldsymbol{J} = \boldsymbol{A}$ for all t. In the LSSM training protocol, no forcing operator or pseudo-inverse appears. Its forward pass is a single linear scan.

With the intermediate variables

$$\boldsymbol{u}_t \coloneqq \boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b} \in \mathbb{R}^L, \tag{66}$$

$$\boldsymbol{a}_t \coloneqq \phi(\boldsymbol{u}_t) = \mathrm{ReLU}(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) \in \mathbb{R}^L, \tag{67}$$

$$\hat{\boldsymbol{x}}_t = \boldsymbol{B}\boldsymbol{a}_t = \boldsymbol{B}\,\mathrm{ReLU}(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) \in \mathbb{R}^N, \tag{68}$$

$$\boldsymbol{e}_t \coloneqq \hat{\boldsymbol{x}}_t - \boldsymbol{x}_t = \boldsymbol{B}\,\mathrm{ReLU}(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) - \boldsymbol{x}_t \in \mathbb{R}^N, \tag{69}$$

$$\boldsymbol{k}_t \coloneqq \Theta(\boldsymbol{u}_t) \odot (\boldsymbol{B}^\top \boldsymbol{e}_t) = \Theta(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) \odot \Big(\boldsymbol{B}^\top \big(\boldsymbol{B}\,\mathrm{ReLU}(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) - \boldsymbol{x}_t\big)\Big) \in \mathbb{R}^L, \tag{70}$$

with the Heaviside step function Θ acting element-wise as in the previous section. The chain rule through the observation model gives

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_t} = \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_t}\frac{\partial \hat{\boldsymbol{x}}_t}{\partial \boldsymbol{a}_t}\frac{\partial \boldsymbol{a}_t}{\partial \boldsymbol{u}_t}\frac{\partial \boldsymbol{u}_t}{\partial \boldsymbol{z}_t} = \frac{2}{NT}\,\boldsymbol{e}_t^\top \boldsymbol{B}\,\mathrm{diag}\big(\Theta(\boldsymbol{u}_t)\big)\boldsymbol{V} = \frac{2}{NT}\,\boldsymbol{k}_t^\top \boldsymbol{V}. \tag{71}$$

Inserting $\boldsymbol{J}_t = \boldsymbol{A}$ and Eq. (71) into Eq. (45) gives

$$\boldsymbol{v}_T^\top = -\frac{2}{NT}\,\boldsymbol{k}_T^\top \boldsymbol{V}, \tag{72}$$

$$\boldsymbol{v}_{t-1}^\top = \boldsymbol{v}_t^\top \boldsymbol{A} - \frac{2}{NT}\,\boldsymbol{k}_{t-1}^\top \boldsymbol{V}, \qquad t = T, \ldots, 2. \tag{73}$$

This linear recursion can be solved in parallel by an associative scan, just like the forward pass. Unrolling the recursion yields

$$\boldsymbol{v}_t^\top = -\sum_{k=t}^{T} \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_k}\,\boldsymbol{A}^{k-t} = -\frac{2}{NT}\sum_{k=t}^{T} \boldsymbol{k}_k^\top \boldsymbol{V}\,\boldsymbol{A}^{k-t}, \tag{74}$$

with $\boldsymbol{A}^0 = \boldsymbol{I}_M$, so that the sensitivity of $\boldsymbol{v}_t$ to later losses is governed by the powers of $\boldsymbol{A}$ alone (for diagonal $\boldsymbol{A}$, $\boldsymbol{A}^{k-t} = \mathrm{diag}(\boldsymbol{a})^{k-t}$, with $\boldsymbol{a}$ the diagonal of $\boldsymbol{A}$).

For $\boldsymbol{\theta} = \{\boldsymbol{A}, \boldsymbol{U}, \boldsymbol{C}, \boldsymbol{h}\}$, Eq. (49) gives, with $\partial F_{\boldsymbol{\theta}}/\partial \boldsymbol{\theta}$ evaluated at $(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t-1}, \boldsymbol{s}_t)$,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{A}} = -\sum_{t=1}^{T} \boldsymbol{z}_{t-1} \boldsymbol{v}_t^\top, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}} = -\sum_{t=1}^{T} \boldsymbol{x}_{t-1} \boldsymbol{v}_t^\top, \tag{75}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{C}} = -\sum_{t=1}^{T} \boldsymbol{s}_t \boldsymbol{v}_t^\top, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}} = -\sum_{t=1}^{T} \boldsymbol{v}_t^\top. \tag{76}$$

If $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{a})$ is diagonal,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{a}} = -\sum_{t=1}^{T} \left(\boldsymbol{z}_{t-1} \odot \boldsymbol{v}_t\right)^\top. \tag{77}$$

The parameters $\boldsymbol{\psi} = \{\boldsymbol{B}, \boldsymbol{V}, \boldsymbol{b}\}$ do not enter the latent recurrence, so $\partial \boldsymbol{R}/\partial \boldsymbol{\psi} = \boldsymbol{0}$ and their gradients are given by the explicit dependence of $\mathcal{L}$ on them only,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{B}} = \frac{2}{NT} \sum_{t=1}^{T} \boldsymbol{a}_t \boldsymbol{e}_t^\top = \frac{2}{NT} \sum_{t=1}^{T} \mathrm{ReLU}(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) \left(\boldsymbol{B}\,\mathrm{ReLU}(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) - \boldsymbol{x}_t\right)^\top, \tag{78}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{V}} = \frac{2}{NT} \sum_{t=1}^{T} \boldsymbol{z}_t \boldsymbol{k}_t^\top = \frac{2}{NT} \sum_{t=1}^{T} \boldsymbol{z}_t \Big(\Theta(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) \odot \left(\boldsymbol{B}^\top (\boldsymbol{B}\,\mathrm{ReLU}(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) - \boldsymbol{x}_t)\right)\Big)^\top, \tag{79}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}} = \frac{2}{NT} \sum_{t=1}^{T} \boldsymbol{k}_t^\top = \frac{2}{NT} \sum_{t=1}^{T} \Big(\Theta(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) \odot \left(\boldsymbol{B}^\top (\boldsymbol{B}\,\mathrm{ReLU}(\boldsymbol{V}\boldsymbol{z}_t + \boldsymbol{b}) - \boldsymbol{x}_t)\right)\Big)^\top. \tag{80}$$

As in the nonlinear case, in practice these gradients are not implemented manually, but computed via automatic differentiation.

F Training methods and details

F.1 Model setup

Parameterization and Initialization For both the RNN and LSSM we parameterize $\boldsymbol{A} = \mathrm{diag}(\tanh(\bar{\boldsymbol{A}}))$ where $\bar{\boldsymbol{A}} \in \mathbb{R}^M$. The nonlinearity avoids instabilities in the $\boldsymbol{A}\boldsymbol{z}_{t-1}$ term when $\|\boldsymbol{A}\|_2 \geq 1$. To facilitate capturing long time scales in the data, we initialize the RNN near the identity to exhibit long time scales from the start:

$$\begin{aligned}
\bar{\boldsymbol{A}} &= \mathrm{artanh}(\kappa)\,\mathbf{I} \\
W_{ij} &\sim \mathcal{U}\left(-(1-\kappa)L^{-1/2}, (1-\kappa)L^{-1/2}\right) \\
V_{ij} &\sim \mathcal{U}\left(-(1-\kappa)M^{-1/2}, (1-\kappa)M^{-1/2}\right) \\
C_{ij} &\sim \mathcal{U}\left(-(1-\kappa)K^{-1/2}, (1-\kappa)K^{-1/2}\right) \\
\boldsymbol{b} &= \boldsymbol{0} \\
\boldsymbol{h} &= \boldsymbol{0},
\end{aligned} \tag{81}$$

where $\mathcal{U}$ is the uniform distribution. In practice, we use $(1-\kappa) \in [10^{-3}, 10^{-5}]$. For $M \geq N$, the observation matrix $\boldsymbol{B}$ is initialized to perform identity read-out on the first N units in the RNN, i.e. $\boldsymbol{B} = \begin{bmatrix}\boldsymbol{I}_N & \boldsymbol{0}_{N\times(M-N)}\end{bmatrix}$.

For the LSSM, we follow a similar strategy:

$$\begin{aligned}
\bar{\boldsymbol{A}} &= \mathrm{artanh}(\kappa)\,\mathbf{I} \\
U_{ij} &\sim \mathcal{U}\left(-(1-\kappa)N^{-1/2}, (1-\kappa)N^{-1/2}\right) \\
C_{ij} &\sim \mathcal{U}\left(-(1-\kappa)K^{-1/2}, (1-\kappa)K^{-1/2}\right) \\
\boldsymbol{h} &= \boldsymbol{0},
\end{aligned} \tag{82}$$

and the observation model is initialized as

$$\begin{aligned} B_{ij} &\sim \mathcal{U}\left(-(1-\kappa)L^{-1/2}, (1-\kappa)L^{-1/2}\right) \\ V_{ij} &\sim \mathcal{U}\left(-(1-\kappa)M^{-1/2}, (1-\kappa)M^{-1/2}\right) \\ \boldsymbol{b} &= \mathbf{0}. \end{aligned} \tag{83}$$

For all experiments, we use $\phi(\cdot) = \mathrm{ReLU}(\cdot) = \max(0, \cdot)$.

F.2 Loss and regularization

Manifold attractor regularization (MAR) For the nonlinear RNN introduced in Eq. (5), we can encourage a slow manifold in the last M_r units by adding the term [77]

$$\mathcal{L}_{\mathrm{MAR}}(\boldsymbol{\theta}_{\mathrm{RNN}}) = \frac{\lambda_{\mathrm{MAR}}}{M_r} \sum_{i=M-M_r+1}^{M} \left[|1 - A_{ii}|^p + \frac{1}{L} \sum_{j=1}^{L} \left(|W_{ij}|^p + |V_{ji}|^p\right) + |h_i|^p \right] \tag{84}$$

to the loss function, where λ_{MAR} is a regularization parameter and $p \in \{1, 2\}$ determines the type of penalty. In practice, we found it beneficial to scale regularization terms of connectivity matrices $\boldsymbol{W}$ and $\boldsymbol{V}$ based on their variance at initialization, i.e.

$$\mathcal{L}_{\mathrm{MAR}}(\boldsymbol{\theta}_{\mathrm{RNN}}) = \frac{\lambda_{\mathrm{MAR}}}{M_r} \sum_{i=M-M_r+1}^{M} \left[|1 - A_{ii}|^p + \frac{1}{L} \sum_{j=1}^{L} \left(\gamma_{\boldsymbol{W}} |W_{ij}|^p + \gamma_{\boldsymbol{V}} |V_{ji}|^p\right) + |h_i|^p \right], \tag{85}$$

where $\gamma_{\boldsymbol{W}} = \frac{1}{3L}$, $\gamma_{\boldsymbol{V}} = \frac{1}{3M}$. For $\lambda_{\mathrm{MAR}} \to \infty$, this regularization leads to stable error propagation by imposing a block-diagonal identity structure in the model Jacobian $\boldsymbol{J}_F$, as we demonstrate below.

Recall the nonlinear RNN from Eq. (5) with $\boldsymbol{A} = \mathrm{diag}(a_1, \ldots, a_M)$:

$$\boldsymbol{z}_t = \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{W}\phi(\boldsymbol{V}\boldsymbol{z}_{t-1} + \boldsymbol{b}) + \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{h}. \tag{86}$$

Its Jacobian is

$$\boldsymbol{J}_F(\boldsymbol{z}_{t-1}) = \boldsymbol{A} + \boldsymbol{W}\,\mathrm{diag}(\phi'(\boldsymbol{V}\boldsymbol{z}_{t-1} + \boldsymbol{b}))\,\boldsymbol{V}. \tag{87}$$

Let $M_s := M - M_r$ and partition $\boldsymbol{z} = (\boldsymbol{z}^{(s)}, \boldsymbol{z}^{(r)})^\top$ into "fast" (unregularized) and "slow" (regularized) units. The diagonal $\boldsymbol{A}$ and the non-linear coupling term decompose as

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_s & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_r \end{pmatrix}, \qquad \boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_s \\ \boldsymbol{W}_r \end{pmatrix}, \qquad \boldsymbol{V} = \begin{pmatrix} \boldsymbol{V}_s & \boldsymbol{V}_r \end{pmatrix}. \tag{88}$$

As $\lambda \to \infty$, the MAR penalty, Eq. (84), drives

$$\boldsymbol{A}_r \to \boldsymbol{I}_{M_r}, \qquad \boldsymbol{W}_r \to \mathbf{0}, \qquad \boldsymbol{V}_r \to \mathbf{0}. \tag{89}$$

Writing $\boldsymbol{D}_s(\boldsymbol{z}_{t-1}) := \mathrm{diag}\Big(\phi'(\boldsymbol{V}_s \boldsymbol{z}_{t-1}^{(s)} + \boldsymbol{b})\Big)$, the limiting Jacobian is block-diagonal:

$$\boldsymbol{J}_F(\boldsymbol{z}_{t-1}) \xrightarrow{\lambda\to\infty} \begin{pmatrix} \boldsymbol{A}_s + \boldsymbol{W}_s \boldsymbol{D}_s(\boldsymbol{z}_{t-1}) \boldsymbol{V}_s & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{M_r} \end{pmatrix}. \tag{90}$$

The latent space decouples into a nonlinear subsystem acting on $\boldsymbol{z}^{(s)}$ and a manifold-attractor subsystem on $\boldsymbol{z}^{(r)}$ with identity dynamics. Consequently, the Jacobian product for a length t sequence and $\lambda \to \infty$ satisfies

$$\prod_{k=0}^{t-1} \boldsymbol{J}_F(\boldsymbol{z}_{t-1-k}) = \begin{pmatrix} \prod_{k=0}^{t-1} \left[\boldsymbol{A}_s + \boldsymbol{W}_s \boldsymbol{D}_s(\boldsymbol{z}_{t-1-k}) \boldsymbol{V}_s\right] & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{M_r} \end{pmatrix}, \tag{91}$$

where error signals propagate without decay or amplification along the M_r directions, enabling stable capture of long-term dependencies.

Observation model regularization In addition to MAR, we reduce the direct contribution of the M_r MAR units to the read-out and hence discourage them from being forced through Eq. (8) by

$$\mathcal{L}_1(\boldsymbol{B}) = \frac{\lambda_1}{N \cdot M_r} \sum_{i=1}^{N} \sum_{j=M-M_r}^{M} |B_{ij}|^p, \tag{92}$$

where $\boldsymbol{B}$ is the readout matrix (cf. Eq. (5)). We also follow [40] and regularize $\boldsymbol{B}$ to stay well-conditioned by pulling its singular values towards 1:

$$\mathcal{L}_2(\boldsymbol{B}) = \frac{\lambda_2}{r} \sum_{i=1}^{r} \left(\sigma_i(\boldsymbol{B}) - 1\right)^2, \tag{93}$$

where $r = \operatorname{rank}(\boldsymbol{B})$ and $\sigma_i(\boldsymbol{B})$ denotes the i-th singular value of $\boldsymbol{B}$.

Total loss The overall loss function used for training of the nonlinear RNN defined in Eq. (5) is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}_{T_w+1:T}, \hat{\boldsymbol{x}}_{T_w+1:T};\, \boldsymbol{\theta}_{\mathrm{RNN}}, \boldsymbol{B}) = \mathcal{L}_{\mathrm{MSE}}(\boldsymbol{x}_{T_w+1:T}, \hat{\boldsymbol{x}}_{T_w+1:T}) \\ + \mathcal{L}_{\mathrm{MAR}}(\boldsymbol{\theta}_{\mathrm{RNN}}) + \mathcal{L}_1(\boldsymbol{B}) + \mathcal{L}_2(\boldsymbol{B}),\end{aligned} \tag{94}$$

with

$$\mathcal{L}_{\mathrm{MSE}}(\boldsymbol{x}_{T_w+1:T}, \hat{\boldsymbol{x}}_{T_w+1:T}) = \frac{1}{N(T-T_w)} \sum_{t=T_w+1}^{T} \|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\|_2^2. \tag{95}$$

MAR for the LSSM Application of MAR to LSSMs (Eq. (3)) is straightforward with

$$\mathcal{L}_{\mathrm{MAR}}(\boldsymbol{\theta}_{\mathrm{LSSM}}) = \frac{\lambda_{\mathrm{MAR}}}{M_r} \sum_{i=M-M_r+1}^{M} \left[|1 - A_{ii}|^p + \frac{1}{N} \sum_{j=1}^{N} |U_{ij}|^p + |h_i|^p \right]. \tag{96}$$

F.3 GTF forcing strength

There are several methods for choosing α in practice [40]. One option is to use *adaptive GTF (aGTF)*, a curriculum learning strategy described in [40], which starts at high forcing subsequently annealed using an exponential moving average. The target α of this annealing schedule is the minimum α required to mitigate exploding gradients and is estimated on the fly from proxies of model Jacobian products. Alternatively, the simpler option is to perform a grid or line search over α values, where usually $\alpha \in (0, 0.5)$ [25, 40, 90]. This search may be initialized using (an estimate of) the maximum Lyapunov exponent $\lambda_{\max}$ of the data [59], where the minimum forcing strength is given by $\alpha = 1 - e^{-\lambda_{\max}}$ [24].

However, [40] provides no guidelines on how to use aGTF in the partially forced setting in this work. Hence, we performed grid search to choose α and report chosen α values in the respective experimental details section (Appx. I). Nonetheless, we also tested aGTF on the forced Lorenz-96 dataset, where most of the dynamics ($N = 6$) are observed, and only the sinusoidal forcing is hidden. Using the `product_upper_bound` strategy from the codebase of [40], we found that reconstruction performance was similar (or even slightly improved) to the one obtained by determining α through grid search, with $D_{\mathrm{stsp}}^{\mathrm{DE}} = 3.7$ compared to 3.8 ± 0.2 in Tab. A3, where aGTF converged to a slightly lower α value of $\alpha \approx 0.055$ compared to $\alpha = 0.08$ as found by grid search.

G Datasets

Lorenz-63 Our first benchmark system is the three-dimensional Lorenz-63 system [51]. Its dynamics are described by the differential equations

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= x(\rho - z) - y, \\ \frac{dz}{dt} &= xy - \beta z.\end{aligned} \tag{97}$$

We chose the classical parameters $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, which put the system into a chaotic regime. We used a Runge-Kutta 4/5 scheme with $\Delta t = 0.01$ and integrated a trajectory of length $100{,}000$ time steps from an initial point $\boldsymbol{x}_0$. The trajectory was standardized per dynamical variable after generation.

Forced Lorenz-96 As a benchmark for long time scales, we equipped the vanilla Lorenz-96 equation [52] with a sinusoidal forcing term:

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})\, x_{i-1} - x_i + F_0 + A \sin(\omega t), \qquad i = 1, \ldots, N, \tag{98}$$

where F_0 is a constant forcing offset, A is the amplitude and ω the frequency of the sinusoidal forcing. We set $F_0 = 14$, $A = 12$, $\omega = 2\pi \,/\, 75$ and $N = 6$. The autonomous Lorenz-96 system (i.e. $A = 0$) is chaotic for $F \geq 8$, such that the sinusoidal forcing leads to a periodic switching between cyclic and chaotic dynamics (see Fig. A3). For integration we used a Runge-Kutta 4/5 scheme with $\Delta t = 5 \cdot 10^{-3}$ and integrated two trajectories from different initial t and $\boldsymbol{x}_0$ of length 500 internal time units, leading to two trajectories of length $100{,}000$ time steps. One trajectory is contaminated with 5% Gaussian observation noise and used for training, while the other trajectory is kept clean and used for testing. Both trajectories are standardized per dynamical variable after generation.

Bursting neuron model We used a three-dimensional simplified Hodgkin-Huxley-type neuron model; its dynamical variables are the membrane potential V as well as two gating variables n and h which control the opening of fast and slow potassium channels, respectively [22, 77]:

$$\begin{aligned}
\dot{V} &= \frac{1}{C}\Big[I - g_L(V - E_L) - g_{\mathrm{Na}}\, m_\infty(V)\,(V - E_{\mathrm{Na}}) - g_K\, n\,(V - E_K) \\
&\qquad - g_M\, h\,(V - E_K) - g_{\mathrm{NMDA}}\, s_\infty(V)\,(V - E_{NMDA})\Big], \\
\dot{n} &= \frac{n_\infty(V) - n}{\tau_n}, \\
\dot{h} &= \frac{h_\infty(V) - h}{\tau_h},
\end{aligned} \tag{99}$$

with

$$\begin{aligned}
m_\infty(V) &= \frac{1}{1 + \exp\big((V_{h,\mathrm{Na}} - V)/k_{\mathrm{Na}}\big)}, \quad & h_\infty(V) &= \frac{1}{1 + \exp\big((V_{h,M} - V)/k_M\big)}, \\
n_\infty(V) &= \frac{1}{1 + \exp\big((V_{h,K} - V)/k_K\big)}, & s_\infty(V) &= \frac{1}{1 + 0.33\, \exp\big(-0.0625\, V\big)}.
\end{aligned} \tag{100}$$

The model parameters we used for our experiments are reported in Tab. A1. Similar to the Lorenz-96 system, we used a Runge-Kutta 4/5 scheme with $\Delta t = 2.5 \cdot 10^{-2}$ and integrated two trajectories from different initial conditions of length 5000 internal time units from which the first 1000 were removed as transients, leading to two trajectories of length $160{,}000$ discrete time steps. One trajectory is contaminated with 5% Gaussian observation noise and used for training, while the other trajectory is kept clean and used for testing, and both trajectories are standardized per dynamical variable after generation. Moreover, to make reconstruction more challenging, we throw away the slow h variable after generation, such that the system is partially observed with $N = 2$. For an example reconstruction, see Fig. 3.

Table A1: Neuron model parameter settings

I	C	g_L	E_L	g_{Na}	E_{Na}	$V_{h,\mathrm{Na}}$	k_{Na}	g_K	E_K	$V_{h,K}$	k_K	τ_n	g_M	$V_{h,M}$	k_M	τ_h	g_{NMDA}	E_{NMDA}
0	6	8	-80	20	60	-20	15	10	-90	-25	7	1	25.2	-18	5	1000	10.2	0

Deng–Hines predator–prey system. The predator–prey system models a tritrophic Rosenzweig–MacArthur-type food chain with Holling type II responses [37] and a fast prey variable, following

[19, 20]. It is described by equations

$$\begin{aligned}\zeta\,\dot{x} &= x\Big(1 - x - \frac{y}{b_1 + x}\Big),\\ \dot{y} &= y\Big(\frac{x}{b_1 + x} - d_1 - \frac{z}{b_2 + y}\Big),\\ \dot{z} &= \varepsilon\, z\Big(\frac{y}{b_2 + y} - d_2\Big),\end{aligned} \tag{101}$$

with the chaotic parameter set $\zeta = 0.05$, $\varepsilon = 1$, $b_1 = 0.25$, $b_2 = 0.1$, $d_1 = 0.2$, $d_2 = 0.39$ and initial condition $(x, y, z) = (0.8, 0.3, 9)$. The prey x is fast ($\zeta \ll 1$); y and z evolve on a common $\mathcal{O}(1)$ timescale. The stiff system was integrated with the implicit Radau method ($\mathrm{rtol} = 10^{-8}$, $\mathrm{atol} = 10^{-10}$, maximum step 0.05) and sampled at $\Delta t = 0.01$. We generated a trajectory of length $540{,}000$ time steps, of which we discarded $40{,}000$ time steps of transient dynamics. We then retained $250{,}000$ steps for training and the following $250{,}000$ for testing. Both time series were standardized with the training mean and standard deviation, and contaminated with i.i.d. Gaussian observation noise of standard deviation 0.05. Only the prey coordinate (x) is observed.

Neurophysiological recordings. To test the algorithm on long real-world time series, whole-cell patch clamp recordings of the membrane potential of single rodent entorhinal cortex neurons in-vivo were used [36]. As described in the main text, these involve slow and irregular state transitions between hyperpolarized "DOWN" and depolarized "UP" states carrying fast spikes, thus multiple timescales from milliseconds to many seconds, as well as considerable noise levels (see Fig. A2). To deal with the noise and stiff spikes in the "UP" states, we applied a Gaussian smoothing filter with standard deviation of $\sigma = 30$ with a kernel length of 241 time points. Additionally, we center and scale the trace by its median and median absolute deviation and then apply an inverse hyperbolic sine,

$$x^{(\mathrm{trf})}_{1:T_{tot}} = \sinh^{-1}\left(\frac{x_{1:T_{tot}} - \mathrm{median}(x_{1:T_{tot}})}{\mathrm{MAD}(x_{1:T_{\mathrm{tot}}})}\right) \tag{102}$$

which leaves the slow up–down fluctuations nearly unchanged while compressing large spikes. The total length of the dataset is $20\mathrm{s}$, which corresponds to $100{,}000$ samples at a sampling rate of $5\mathrm{kHz}$.

H Evaluation measures

State space divergence $D_{\mathbf{stsp}}$ The state space divergence D_{stsp} measures the geometrical disagreement between state distributions of the data $p(\boldsymbol{x})$ and that of model generated trajectories $q(\boldsymbol{x})$ by evaluating the Kullback-Leibler (KL) divergence:

$$D_{\mathrm{stsp}} := D_{\mathrm{KL}}\left(p(\boldsymbol{x}) \,||\, q(\boldsymbol{x})\right) = \int_{\boldsymbol{x}\in\mathbb{R}^N} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x}. \tag{103}$$

In practice, we estimate $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ by placing Gaussian Mixture Models (GMM) along orbits [11, 47]), i.e.

$$\begin{aligned}\hat{p}(\boldsymbol{x}) &= \frac{1}{T_1}\sum_{t=1}^{T} \mathcal{N}(\boldsymbol{x};\ \boldsymbol{x}_t, \boldsymbol{\Sigma})\\ \hat{q}(\boldsymbol{x}) &= \frac{1}{T_2}\sum_{t=1}^{T_2} \mathcal{N}(\boldsymbol{x};\ \hat{\boldsymbol{x}}_t, \boldsymbol{\Sigma}),\end{aligned} \tag{104}$$

where $\boldsymbol{x}_{1:T_1}$ is the ground-truth data of length T_1 and $\hat{\boldsymbol{x}}_{1:T_2}$ is a model-generated trajectory of length T_2, $\mathcal{N}(\boldsymbol{x};\ \boldsymbol{x}_t, \boldsymbol{\Sigma})$ is a multivariate Gaussian with mean vector $\boldsymbol{x}_t$ and covariance matrix $\boldsymbol{\Sigma} = \mathrm{diag}([\sigma_1^2, \ldots, \sigma_N^2])$. The KL-divergence between two GMMs can be approximated using a Monte-Carlo approach [39]

$$D_{\mathrm{stsp}} = D_{\mathrm{KL}}\left(\hat{p}(\boldsymbol{x}) \,||\, \hat{q}(\boldsymbol{x})\right) \approx \frac{1}{n}\sum_{i=1}^{n} \log \frac{\hat{p}(\boldsymbol{x}^{(i)})}{\hat{q}(\boldsymbol{x}^{(i)})}, \tag{105}$$

with n Monte Carlo samples $\boldsymbol{x}^{(i)}$ randomly drawn from the GMM based on observed orbits, $\hat{p}(\boldsymbol{x}^{(i)})$. When the observed system is only partially observed and the true dynamics unfold in $d > N$

dimensions, the attractor first has to be unfolded to apply D_{stsp} correctly [8]. To this end we first apply a delay embedding (DE) [74, 83] of the observed and model generated orbits

$$\boldsymbol{\xi}_t = \left[\boldsymbol{x}_t, \boldsymbol{x}_{t-\tau}, \boldsymbol{x}_{t-2\tau}, \ldots, \boldsymbol{x}_{t-(m-1)\tau}\right]^\top \in \mathbb{R}^{mN} \tag{106}$$

with delay τ and embedding dimension m, and the same construction performed for $\hat{\boldsymbol{\xi}}_t$. We then fit GMMs and calculate D_{stsp} in embedding space, i.e.

$$D_{\text{stsp}}^{\text{DE}} = D_{\text{KL}}\left(\hat{p}(\boldsymbol{\xi}) \,||\, \hat{q}(\boldsymbol{\xi})\right) \approx \frac{1}{n}\sum_{i=1}^{n} \log \frac{\hat{p}(\boldsymbol{\xi}^{(i)})}{\hat{q}(\boldsymbol{\xi}^{(i)})}. \tag{107}$$

For all experiments, we use $n = 10^6$ and set the bandwidth $\boldsymbol{\Sigma} = f_{bw} \cdot \text{diag}([\hat{\sigma}_1^2, \ldots, \hat{\sigma}_d^2])$, where $f_{bw} = \left(\frac{T_1(d+2)}{4}\right)^{-1/(d+4)}$ is a bandwith factor chosen according to Silverman's rule-of-thumb [79] and $\hat{\sigma}_i$ is the empirical standard deviation of $x_{i,1:T_1}$. Furthermore, to probe long-term consistency beyond the available data, we choose to generate model orbits for $T_2 = 3 \cdot T_1$. This avoids low D_{stsp} when the model is only *transiently* generating the correct long-term behavior.

Since most systems are only partially observed, we use $D_{\text{stsp}}^{\text{DE}}$ ((107)) with parameters τ and m given per dataset in Tab. A2. These were chosen to specifically resolve long time scales in the data, except for the partially observed Lorenz-63, which does not exhibit particularly long time scales.

Table A2: Delay embedding parameters used for each dataset.

Dataset	$\boldsymbol{\tau}$	$\boldsymbol{m}$
Lorenz-63 (1D)	10	3
Forced Lorenz-96 (6D)	4,096	3
Bursting Neuron (2D)	1,024	7
Deng–Hines (1D)	350	5
Neural Recordings (1D)	1,634	5

Short-term RMSE The n-step-ahead RMSE assesses short- to medium-length forecasts and is defined as

$$\text{RMSE}(n) = \sqrt{\frac{1}{n \cdot N}\sum_{k=1}^{n} \left\|\boldsymbol{x}_k - G_{\boldsymbol{\psi}}(F_{\boldsymbol{\theta}}^{\circ k}(\boldsymbol{z}_0))\right\|^2} \tag{108}$$

where $\boldsymbol{x}_{1:n} := \boldsymbol{x}_{t_i:t_i+n-1}$ is a window starting at index t_i of the total ground-truth data $\boldsymbol{x}_{1:T_{\text{tot}}}$. The initial condition $\boldsymbol{z}_0$ is obtained by performing a warm-up using a history of length T_w, $\boldsymbol{x}_{t_i-T_w:t_i-1}$ (see Sect. 4.2 for details on warm-up). In practice we averaged the RMSE over 100 windows from different initial conditions.

I Additional experimental details

This appendix section lists specific experimental details of experiments performed in the main paper. All experiments were performed on a compute server with 2x96-Core AMD EPYC 9655 CPUs, 768 GB RAM, and 6x NVIDIA RTX Pro 6000 Blackwell (96 GB) GPUs; however, individual trainings were always performed on a single GPU at a time.

I.1 Figure 1

Fig. 1A We trained shPLRNNs (Eq. (5), $L = 50$) of varying latent dimensions $M \in \{4, 16, 64, 128\}$ on the Lorenz-63 system using GTF-DEER. The batch size was fixed at $B = 1$, while the sequence length was varied in powers of 2, i.e. $T \in \{128, 256, \ldots, 32{,}768\}$ and we used no warm-up for this experiment ($T_w = 0$). We measured the time of combined forward+backward passes throughout training. Training was performed for $10,000$ updates (samples). We chose to explicitly train a model to account for the fact that GTF-DEER's convergence depends on the dynamics of the

model, giving us an estimate of the spread of runtimes throughout training. Indeed, for the chosen $\alpha = 0.15$, GTF-DEER's convergence stayed consistent throughout training (see also Fig. 2**B**). The sequential baseline runtime is independent of explicit dynamics, and hence we only measured the time of trained models using $N_s = 500$ samples. In code, the only difference between the sequential and parallel implementation is the call to the forward pass solver, which is given by a straight-forward `jax.lax.scan` for the sequential case, and the `seq1d` (DEER solver) function for the parallel one, respectively. Both methods parallelize over the batch size using `jax.vmap`. All data were gathered on an NVIDIA RTX PRO 6000 Blackwell (96GB) GPU.

Fig. 1B For this figure we used a shPLRNN (varying M, fixed $L = 50$) that was trained on the full Lorenz-63 system ($N = 3$). We performed forward passes for different values of forcing strength $\alpha \in [0, 1]$. The maximum number of Newton iterations was capped at $n_{\text{iter}} = 500$: In practice, GTF-DEER loses its competitive performance over sequential evaluation even for moderate Newton iterations such that 500 is already a generous upper bound of what can be considered a useful regime. The different curves correspond to different initial guess strategies, where 'Zero' corresponds to initializing $\boldsymbol{z}_{1:T}^{(0)} = \mathbf{0}_{T\times M}$. '$\mathcal{N}(0, 1)$' draws entries from a standard Normal distribution, $z_{i,t}^{(0)} \sim \mathcal{N}(0, 1)$ and 'P-inv' uses $\boldsymbol{z}_{1:T}^{(0)} = \boldsymbol{B}^{+}\boldsymbol{x}_{1:T}$ to initialize the Newton iterations.

I.2 Figure 2

For this experiment we trained shPLRNNs ($M = 5$, $L = 50$) on the Lorenz-63 under different conditions. We trained on the full Lorenz-63 system ($N = 3$) as well as on the partially observed one ($N = 1$, x-component). Furthermore, we switched between GTF-DEER with full Jacobians (Eq. (13)) and diagonalized ones (quasi-DEER, $\boldsymbol{J}_F \to \text{diag}(\boldsymbol{J}_F)$). We trained 20 models per setting for 20,000 parameter updates. The forcing strength was $\alpha = 0.15$.

I.3 Section 5.4

I.3.1 Figure 3A

We trained shPLRNNs with $M = 10$ and $L = 128$ on the forced Lorenz-96 system and $M = 6$ for the bursting neuron model (see Appx. G). To test the effect of increasing sequence length on this problem, we trained the models on sequence lengths $T \in \{256, 512, 1{,}024, 2{,}048, 4{,}096, 8{,}192, 16{,}384, 32{,}768\}$, where for each setting we used $T_w = T/2$. To keep data parsed by the model per parameter update the same, we adjusted batch size accordingly, such that the product $B \cdot T = 2^{15} = 32{,}768$ staid fixed. This resulted in settings $(B, T) = (256, 128)$ and $(B, T) = (1, 32{,}768)$ for the lowest and highest sequence length used, respectively. For each data point in the figure, we trained 10 independent models. For the Lorenz-96 we used a forcing strength of $\alpha = 0.08$, for the bursting neuron model $\alpha = 0.4$.

I.3.2 Mamba-2

As a state-of-the-art SSM baseline, we compare the LSSM (Eq. (3)) and the GTF-DEER-trained shPLRNN (Eq. (5)) to a model in which the linear recurrence of the LSSM is replaced by a single Mamba-2 block [18], while keeping the rest of the architecture, in particular the one-hidden-layer MLP read-out from Eq. (3), identical. We refer to this configuration as Mamba-2-n_p, where n_p is the number of parameters. For the non-recurrence machinery of the Mamba-2 block (input projections, depth-wise causal convolution, gating branch, multi-head setup, output projection) we refer the reader to the original work by [18]. In the following, we clarify the–in our minds important–distinction to the LSSM introduced in Eq. (3).

Selective SSM recurrence. The main *functional* difference of the Mamba recurrence is the introduction of a *selection mechanism* [18, 33], which renders the model non-autonomous even in the absence of external inputs $\boldsymbol{s}_t$. Dropping $\boldsymbol{s}_t$ for brevity and using the LSSM (Eq. (3)) as a template, the core recurrence inside the Mamba-2 block can be written as

$$\boldsymbol{z}_t = \bar{\boldsymbol{A}}_t(\boldsymbol{x}_{t-1})\, \boldsymbol{z}_{t-1} + \bar{\boldsymbol{U}}_t(\boldsymbol{x}_{t-1})\, \boldsymbol{x}_{t-1} \tag{109}$$

i.e., *structurally* identical to the LSSM but with $\bar{\boldsymbol{A}}_t$ and $\bar{\boldsymbol{U}}_t$ now depending on the input $\boldsymbol{x}_{t-1}$. Specifically, parameterizations of $\bar{\boldsymbol{A}}_t$ and $\bar{\boldsymbol{U}}_t$ are given by

$$\begin{aligned}
\Delta_t &= \mathrm{softplus}(\boldsymbol{W}_\Delta \boldsymbol{x}_{t-1} + \boldsymbol{b}_\Delta) \in \mathbb{R} \\
\boldsymbol{U}_t &= \boldsymbol{W}_U \boldsymbol{x}_{t-1} \\
\bar{\boldsymbol{A}}_t &= \exp(\Delta_t \, \boldsymbol{A}) \\
\bar{\boldsymbol{U}}_t &= \Delta_t \, \boldsymbol{U}_t,
\end{aligned} \tag{110}$$

where $\boldsymbol{A}$ is a *fixed*, learnable parameter which for Mamba-2 is restricted to $\boldsymbol{A} = a\,\boldsymbol{I}$ for a single learned scalar $a < 0$ per head. The recurrence parameters thus become functions of the input through the data-dependent scalar Δ_t and the data-dependent input vector $\boldsymbol{U}_t$, while $\boldsymbol{A}$ itself remains time-invariant. Setting $\Delta_t = \Delta$ and $\boldsymbol{U}_t = \boldsymbol{U}$ recovers a fixed-parameter linear recurrence equivalent to the LSSM (Eq. (3)).[5]

Relation to the LSSM Crucially, the recurrence in Eq. (109) is still *linear in the latent state* $\boldsymbol{z}_{t-1}$ for any fixed value of $\boldsymbol{x}_{t-1}$, which is what permits parallelization via parallel scan during training [18]. At the same time, the selection mechanism circumvents the $\mathrm{rank} \leq \min(N, M, L)$ bottleneck on the effective test-time recurrence (Eq. (4)) discussed in Sect. 3.1: because $\boldsymbol{U}_t$ varies with t, the model is no longer constrained to a single fixed product $\overline{\boldsymbol{W}} = \boldsymbol{U}\boldsymbol{B}$ of fixed rank, but can in principle realize a different effective input pathway at every step.

I.3.3 Fig. 3B hyperparameters

Optimization settings For experiments in Sect. 5.4 we used the following shared hyperparameters: For the forced Lorenz-96 experiments in 3**B**, we used batch size $B = 1$, sequence length $T = 81{,}920$ with warm-up $T_w = 16{,}384$ and a learning rate decay from $\eta_s = 5 \cdot 10^{-5}$ to 10^{-6} using a cosine decay schedule [53]. Training was performed for $150{,}000$ parameter updates.

Mamba-2 For the Mamba-2 models, we evaluated two parameter budgets, denoted Mamba-2-7k with $7{,}038$ and Mamba-2-17k with $16{,}830$ parameters, respectively. The former is matched as closely as possible to the shPLRNN/LSSM budget ($\approx$ 2,200–2,900 parameters) for direct comparison at fixed capacity without overly limiting Mamba-2's expressiveness; the latter probes whether additional capacity closes the gap. Both use a single Mamba-2 block with hidden dimension $D = 16$ (`d_model`) for Mamba-2-7k and $D = 32$ for Mamba-2-17k. All other parameters stay the same: State size $N = 16$ (`d_state`), `d_conv` $= 4$ and `expand` $= 2$. The Mamba-2 block is followed by the same MLP read-out as used in the LSSM (Eq. (3)) with $L = 128$. We used the Mamba-2 implementation from the original code repository `https://github.com/state-spaces/mamba`, licensed under the Apache License, Version 2.0.

RNN The RNN used $M = 10$ and $L = 128$ and $M_r = 4$. We used $\lambda_{\mathrm{MAR}} = 1$, $\lambda_1 = 1$ and $\lambda_2 = 10^{-4}$, however, we found the regularization of the singular values of $\boldsymbol{B}$ did not have a huge impact on training convergence and reconstruction performance. The forcing was $\alpha = 0.08$.

LSSM For the LSSM we kept core settings equal to the ones used in the RNN, i.e. $M = 10$, $L = 128$. Additionally, we apply MAR to the LSSM (see Appx. F) where we scanned $\lambda_{\mathrm{MAR}} \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ and found that all settings produced similar results, except $\lambda_{\mathrm{MAR}} = 10$ which degraded performance. Hence we reported $\lambda_{\mathrm{MAR}} = 1$ to make clear that MAR does not improve reconstruction for the LSSM. We further used $M_r = 4$ in this case, mirroring the setting of the RNN.

[5] For clarity, Eqs. (109) & (110) describe the Mamba-2 recurrence at the level of abstraction relevant for comparison with the LSSM. In practice, a Mamba-2 block does not operate directly on $\boldsymbol{x}_{t-1}$: the input is first lifted to an expanded representation of dimension $D = E \cdot N$ via a linear projection, where E is the expansion factor, then split into H heads, and each head runs an independent SSM with its own data-dependent parameters $(\Delta_t, \boldsymbol{U}_t)$ shared across the channels within that head, while $\boldsymbol{A}$ is a learned scalar per head. Channel mixing across the observation dimensions therefore occurs in the 'peripheral' input/output projections rather than inside the recurrence itself. We refer the reader to [18] for the full block specification.

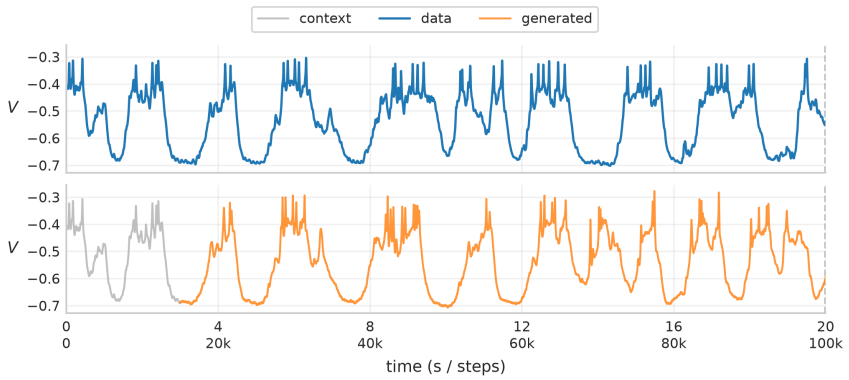


Figure A2: Example rollout of a shPLRNN trained through GTF-DEER (bottom) on physiological neuron recordings (top) [36]. To mimic process noise present in the data, we applied isotropic Gaussian process noise during the roll-out with strength $\sigma = 10^{-3}$, i.e. $\boldsymbol{z}_t = F_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}) + \sigma\boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{M\times M})$.

I.3.4 Predator–prey and neurophysiological data

Predator-Prey. For the 1D predator–prey system, we trained shPLRNNs ($M = 10, L = 128$) with GTF-DEER using $\alpha = 0.15$. For regularization losses, we used $M_r = 9$ with $\lambda_{\text{MAR}} = 0.1$, $\lambda_1 = 20$ and $\lambda_2 = 10^{-2}$. We used a learning rate of $\eta = 10^{-4}$, which decayed using a cosine schedule to reach 10^{-5} at the end of training. For the LSSM, we used $M = 20$ and $L = 128$, matching the parameter count of the shPLRNNs. We also started at a slightly higher learning rate of $\eta = 5 \cdot 10^{-4}$. The Mamba-2 model was the same as the small model used in experiments on the forced Lorenz-96, i.e. a single Mamba-2 block with $D = 16$ (`d_model`), state size $N = 16$ (`d_state`), `d_conv` $= 4$ and `expand` $= 2$, followed by an MLP with hidden size $L = 128$. For all models, parameter updates were calculated based on single sequences ($B = 1$) with common length $T = 49{,}152$, of which $T_W = 16{,}384$ are used for warm-up. For the shPLRNNs specifically, we also trained models with $T = 1{,}536$ and $B = 32$ (i.e. the same amount of total data per update), showing that the training benefits massively from sequences which span a long contiguous segment of the observed data.

Neurophysiological data. For the physiological voltage recordings from [36], we trained shPLRNNs ($M = 15, L = 128$) with GTF-DEER using $\alpha = 0.09$. For regularization losses, we used $M_r = 14$ with $\lambda_{\text{MAR}} = 0.01$, $\lambda_1 = 0.1$ and $\lambda_2 = 10^{-4}$. We also applied randomly drawn isotropic Gaussian noise to teacher forcing signals after sampling the training sequence with standard deviation $\sigma = 0.025$, which we found to stabilize training for this data. We used a learning rate of $\eta = 5 \cdot 10^{-5}$, which decayed using a cosine schedule to reach $5 \cdot 10^{-6}$ at the end of training. Additionally, we used a mean absolute value (MAE) loss, which we found to provide reconstructions of higher fidelity.

For the LSSM, we used $M = 29, L = 128$ to match the parameter count of the shPLRNN. We also applied MAR to all states and L2 regularization to the observation model with strength $\lambda = 0.01$, which we found through line search in $\lambda \in [10^{-5}, 10^{-4}, \ldots, 10^{-1}]$. We used the same learning rate schedule as for the shPLRNN trained through GTF-DEER.

Since the Mamba-2 model profited from more parameters in Tab. A3, this higher parameter count was also used in the present experiment. The model is again comprised of a single Mamba-2 block with hidden dimension $D = 32$ (`d_model`), state size $N = 16$ (`d_state`), `d_conv` $= 4$ and `expand` $= 2$. The Mamba-2 block is followed by an output MLP with hidden size $L = 128$.

Finally, all models used a common sequence length $T = 50{,}000$, of which $T_w = 10{,}000$ are used for warm-up, and batch size $B = 2$.

J Additional figures and tables

Table A3: Model comparisons on two simulated and one real-world benchmark. Columns are T: sequence length, B: batch size, long-term measure $D_{\text{stsp}}^{\text{DE}}$, short-term measure $\text{RMSE}(128)$, and number of parameters $|\boldsymbol{\theta}|$. All values are median $\pm$ MAD over 10 independent runs. While (d-)LSSMs and Mamba-2 models are trained using conventional TF, shPLRNNs are trained using GTF-DEER (highlighted in gray). Bold values denote the best result per dataset and metric within overlapping $\pm$ MAD intervals.

Dataset	**Model**	T	B	$D_{\text{stsp}}^{\text{DE}}$	RMSE(128)	$\lvert\boldsymbol{\theta}\rvert$
Forced Lorenz-96 (6D)	LSSM	81,920	1	8.8 ± 1.8	0.26 ± 0.01	2,256
	LSSM + MAR	81,920	1	9.1 ± 2.2	0.41 ± 0.02	2,256
	d-LSSM	81,920	1	7.8 ± 0.9	0.27 ± 0.01	2,346
	d-LSSM + MAR	81,920	1	9.9 ± 3.8	0.32 ± 0.01	2,346
	Mamba-2-7k	81,920	1	10.5 ± 2.6	0.21 ± 0.02	$7,038$
	Mamba-2-17k	81,920	1	8.7 ± 0.8	$\mathbf{0.14 \pm 0.02}$	16,830
	shPLRNN ($r = 6$)	81,920	1	7.7 ± 0.3	0.19 ± 0.01	2,316
	shPLRNN ($r = 7$)	81,920	1	$\mathbf{3.8 \pm 0.2}$	$\mathbf{0.16 \pm 0.01}$	2,454
	shPLRNN	81,920	1	4.3 ± 0.2	0.17 ± 0.01	2,868
Deng–Hines (1D)	LSSM	49,152	1	15 ± 12	0.22 ± 0.13	2,876
	Mamba-2	49,152	1	34.5 ± 0.5	0.093 ± 0.001	4,761
	shPLRNN	1,536	32	36 ± 3	11 ± 10	2,718
	shPLRNN	49,152	1	$\mathbf{0.6 \pm 0.4}$	$\mathbf{0.076 \pm 0.012}$	2,718
Neural recordings (1D)	LSSM	50,000	2	6.2 ± 2.5	$\mathbf{0.068 \pm 0.002}$	4,055
	Mamba-2	50,000	2	50.3 ± 0.6	$\mathbf{0.068 \pm 0.001}$	16,505
	shPLRNN	50,000	2	$\mathbf{0.68 \pm 0.02}$	$\mathbf{0.071 \pm 0.008}$	4,013

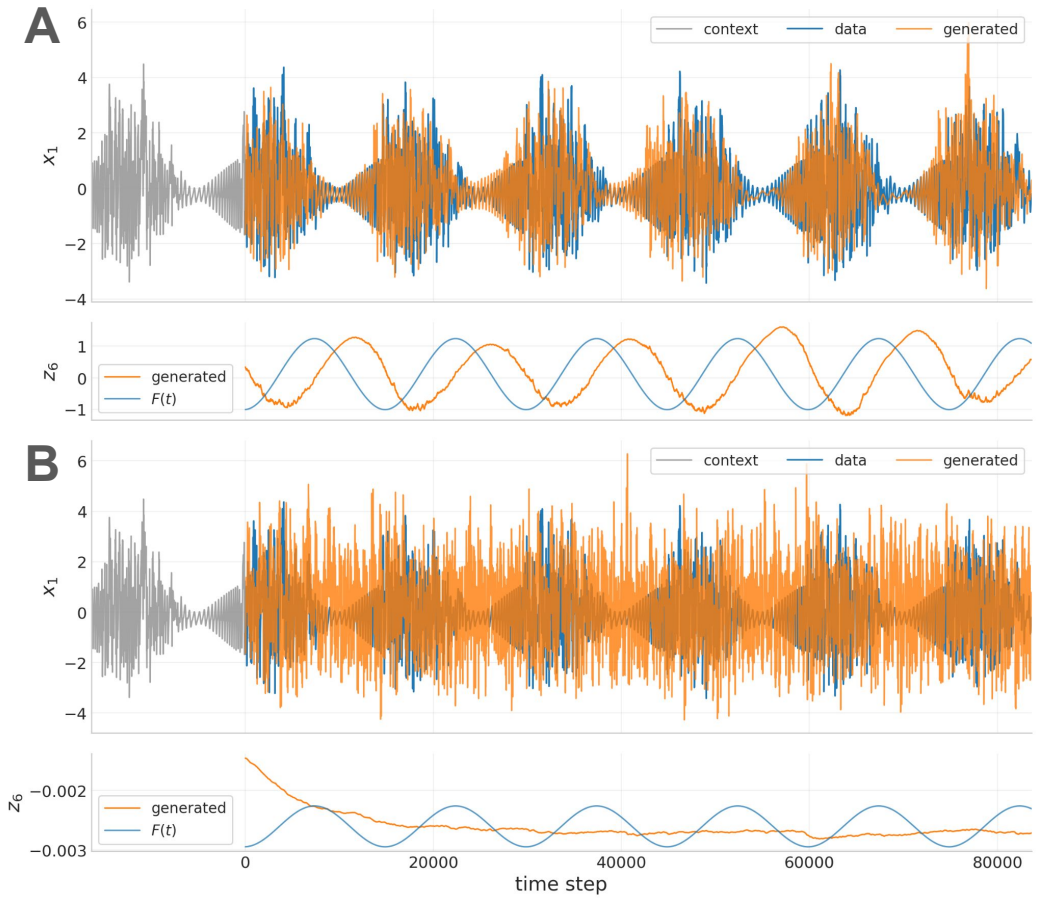


Figure A3: Generated dynamics of a shPLRNN ($M = 10, L = 128$) trained on the forced Lorenz-96 system. **A**: Access to long sequences during training enables the model to capture the latent long-period sinusoidal forcing $F(t)$ in the MAR regularized units, and hence to produce accurate long-term rollouts. **B**: A model trained on short sequence lengths w.r.t. the intrinsic time scales of the data fails to capture and reproduce the latent forcing during auto-regressive roll-outs.

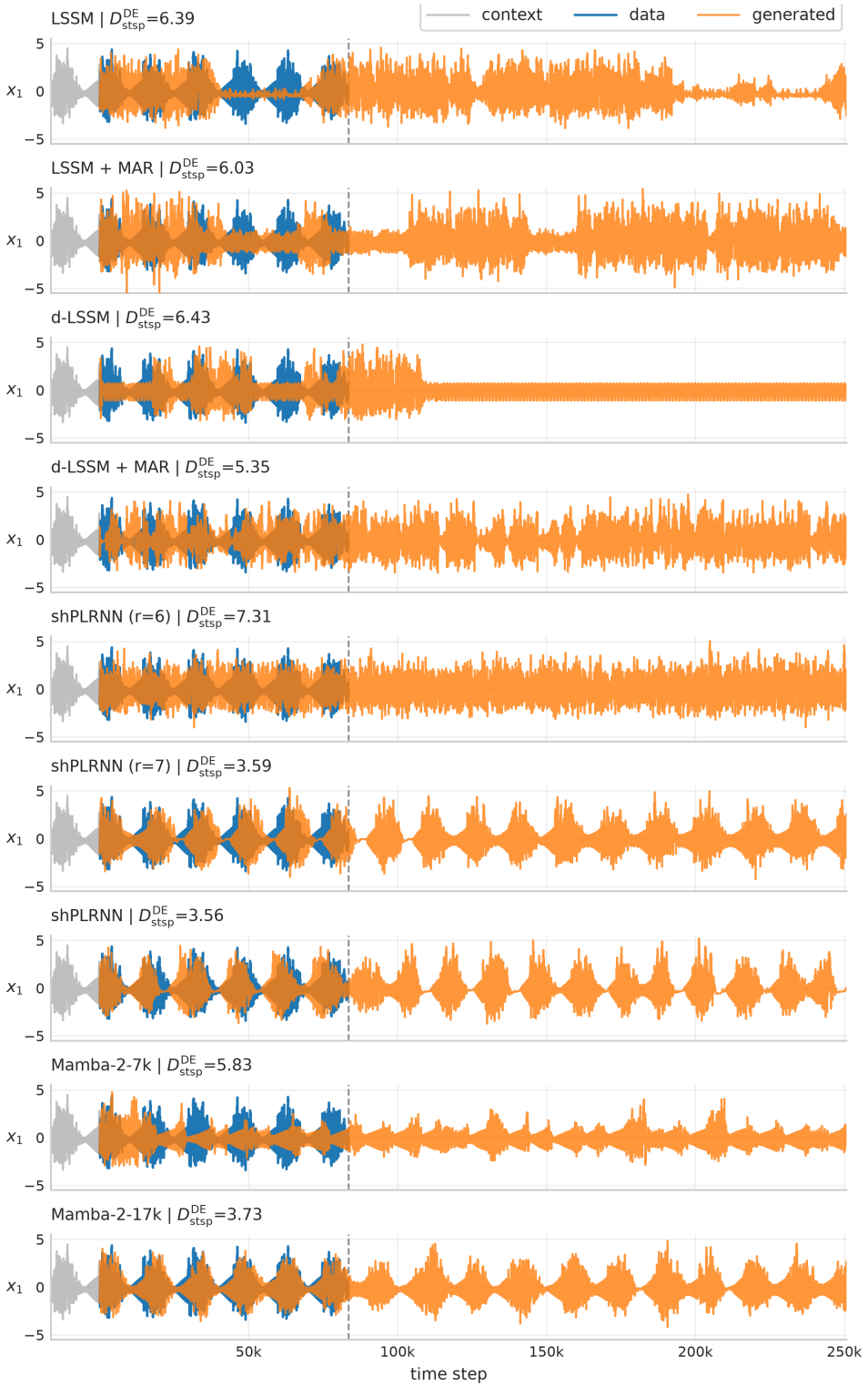


Figure A4: Generated dynamics across $3\times$ the length of available ground truth data from models evaluated in Sect. 5.4. shPLRNNs with their rank limited to the number of observed variables $r = N = 6$ and all LSSM variants fail to learn the latent sinusoidal forcing. Increasing the rank to 7 in the shPLRNN enables the model to learn the underlying forcing and to produce convincing limiting behavior; however, the model still struggles to infer the correct phase from the context. The Mamba-2 based model benefits from more parameters and can achieve good reconstructions (Mamba-2-17k), but lacks the robustness of the shPLRNN trained through GTF-DEER (see Fig. 3**B** and Tab. A3). Each generated trace was produced using the best model out of 10 per configuration.

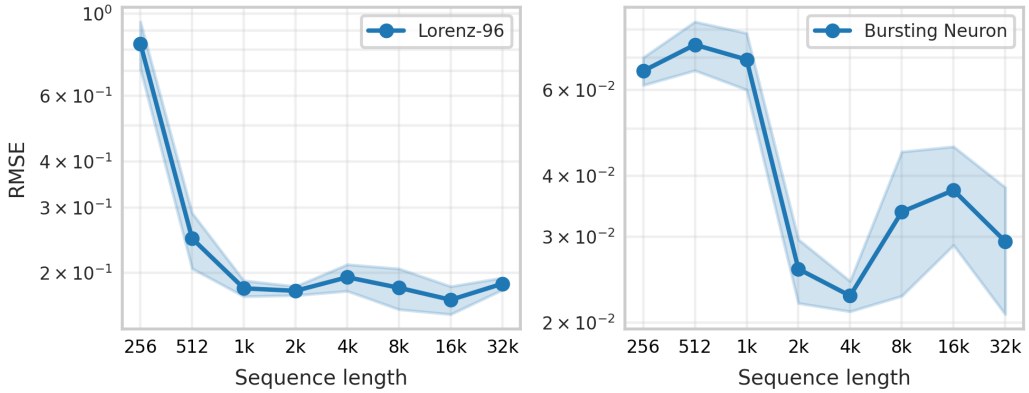


Figure A5: RMSE(128) as a function of sequence length for shPLRNNs trained on the forced Lorenz-96 and bursting neuron systems, complementary to the D_{stsp} plot in Fig. 3. Curves show Median $\pm$ MAD, 10 runs per setting. The RMSE(128) already saturates for shorter sequence lengths, as expected by design of this short-term measure (mean $\pm$ SEM over all sequence lengths and runs: 0.27 ± 0.02 for the Lorenz-96 and 0.047 ± 0.003 for the bursting neuron system, respectively.)

K Training algorithms

Algorithm 1 Linear SSM training (using parallel associative scan)

Require: Data $\boldsymbol{X} \in \mathbb{R}^{T_{\text{obs}} \times N}$, optional external inputs $\boldsymbol{S} \in \mathbb{R}^{T_{\text{obs}} \times K}$, linear recurrence $F_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t-1}, \boldsymbol{s}_t)$, decoder $G_{\boldsymbol{\psi}}(\boldsymbol{z}_t)$, batch size B, sequence length T, warm-up length T_w

Ensure: Trained parameters $\boldsymbol{\theta}, \boldsymbol{\psi}$

repeat
  Sample sequences $\boldsymbol{x}_{0:T} \in \mathbb{R}^{B \times (T+1) \times N}$ from $\boldsymbol{X}$, $\boldsymbol{s}_{1:T} \in \mathbb{R}^{B \times T \times K}$ from $\boldsymbol{S}$
  ▷ *Implicit vectorized batch processing below, e.g. via* `jax.vmap` ◁
  Initialize $\boldsymbol{z}_0 = \boldsymbol{0}$
  $\boldsymbol{z}_{1:T} \leftarrow \text{associative_scan}(F_{\boldsymbol{\theta}}, \boldsymbol{z}_0, \boldsymbol{x}_{0:T-1}, \boldsymbol{s}_{1:T})$ ▷ *Solve in parallel on GPU*
  $\hat{\boldsymbol{x}}_{1:T} = G_{\boldsymbol{\psi}}(\boldsymbol{z}_{1:T})$
  $\mathcal{L} \leftarrow \text{MSE}(\boldsymbol{x}_{T_w+1:T}, \hat{\boldsymbol{x}}_{T_w+1:T})$ ▷ *Exclude warm-up steps from loss*
  $(\boldsymbol{g}_{\boldsymbol{\theta}}, \boldsymbol{g}_{\boldsymbol{\psi}}) \leftarrow \text{grad}(\mathcal{L};\ \boldsymbol{\theta}, \boldsymbol{\psi})$
  $(\boldsymbol{\theta}, \boldsymbol{\psi}) \leftarrow \text{optimize}(\boldsymbol{\theta}, \boldsymbol{\psi}, \text{g}_{\boldsymbol{\theta}}, \text{g}_{\boldsymbol{\psi}})$
until convergence

Algorithm 2 Initial value / trajectory matching training with generalized teacher forcing

Require: Data $\boldsymbol{X} \in \mathbb{R}^{T_{\text{obs}} \times N}$, optional external inputs $\boldsymbol{S} \in \mathbb{R}^{T_{\text{obs}} \times K}$, recurrent model $F_{\boldsymbol{\theta}}$, observation matrix $\boldsymbol{B} \in \mathbb{R}^{N \times M}$, forcing strength $\alpha \in [0, 1]$, batch size B, sequence length T

Ensure: Trained parameters $\boldsymbol{\theta}, \boldsymbol{B}$

repeat
  Compute $\boldsymbol{B}^+ \leftarrow \text{pinv}(\boldsymbol{B}), \quad \boldsymbol{P} \leftarrow \boldsymbol{I}_M - \alpha \boldsymbol{B}^+ \boldsymbol{B}$
  Sample sequences $\boldsymbol{x}_{0:T} \in \mathbb{R}^{B \times (T+1) \times N}$ from $\boldsymbol{X}$, $\boldsymbol{s}_{1:T} \in \mathbb{R}^{B \times T \times K}$ from $\boldsymbol{S}$
  ▷ *Implicit vectorized batch processing below, e.g. via* `jax.vmap` ◁
  Compute teacher signals $\bar{\boldsymbol{z}}_{0:T} = \boldsymbol{B}^+(x_{0:T})$
  Initialize $z_0 \leftarrow \bar{z}_0$
  for $t = 1, \ldots, T$ **do**
    $\tilde{\boldsymbol{z}}_{t-1} \leftarrow \boldsymbol{P}\, \boldsymbol{z}_{t-1} + \alpha\, \bar{\boldsymbol{z}}_{t-1}$ ▷ *Build weighted forced state*
    $\boldsymbol{z}_t \leftarrow F_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_{t-1}, \boldsymbol{s}_t)$
    $\hat{\boldsymbol{x}}_t \leftarrow \boldsymbol{B} \boldsymbol{z}_t$
  $\mathcal{L} \leftarrow \text{MSE}(\boldsymbol{x}_{1:T}, \hat{\boldsymbol{x}}_{1:T})$
  $(\boldsymbol{g}_{\boldsymbol{\theta}}, \boldsymbol{g}_{\boldsymbol{B}}) \leftarrow \text{grad}(\mathcal{L};\ \boldsymbol{\theta}, \boldsymbol{B})$ ▷ *Backpropagation through time*
  $(\boldsymbol{\theta}, \boldsymbol{B}) \leftarrow \text{optimize}(\boldsymbol{\theta}, \boldsymbol{B}, \boldsymbol{g}_{\boldsymbol{\theta}}, \boldsymbol{g}_{\boldsymbol{B}})$
until convergence

Algorithm 3 GTF-DEER: Generalized teacher forcing with DEER parallel solver

Require: Data $\boldsymbol{X} \in \mathbb{R}^{T_{\text{obs}} \times N}$, optional external inputs $\boldsymbol{S} \in \mathbb{R}^{T_{\text{obs}} \times K}$, recurrent model $F_{\boldsymbol{\theta}}$, observation matrix $\boldsymbol{B} \in \mathbb{R}^{N \times M}$, forcing strength $\alpha \in [0, 1]$, batch size B, sequence length T, tolerance ε

Ensure: Trained parameters $\boldsymbol{\theta}$, $\boldsymbol{B}$

repeat

  Compute $\boldsymbol{B}^+ \leftarrow \text{pinv}(\boldsymbol{B}), \quad \boldsymbol{P}_\alpha \leftarrow \boldsymbol{I}_M - \alpha \boldsymbol{B}^+ \boldsymbol{B}, \quad \boldsymbol{P}_1 \leftarrow \boldsymbol{I}_M - \boldsymbol{B}^+ \boldsymbol{B}$

  Sample sequences $\boldsymbol{x}_{0:T} \in \mathbb{R}^{B \times (T+1) \times N}$ from $\boldsymbol{X}$, $\boldsymbol{s}_{1:T} \in \mathbb{R}^{B \times T \times K}$ from $\boldsymbol{S}$

  ▷ *Implicit vectorized batch processing below, e.g. via* `jax.vmap` ◁

  Compute forcing targets $\bar{\boldsymbol{z}}_t \leftarrow \boldsymbol{B}^+ \boldsymbol{x}_t$ for $t = 0, \ldots, T$

  Initialize $\boldsymbol{z}_0 \leftarrow \bar{\boldsymbol{z}}_0$ ▷ *Below, fix* $\boldsymbol{z}_0^{(k)} = \boldsymbol{z}_0 \; \forall k$

  Initial guess $\boldsymbol{z}_{1:T}^{(0)} \leftarrow \bar{\boldsymbol{z}}_{1:T}$ ▷ *Warm start from data*

  for $k = 0, 1, \ldots$ **until** $\|\Delta \boldsymbol{z}_{1:T}^{(k+1)}\|_\infty < \varepsilon$ **do** ▷ *DEER iteration*

    $\tilde{\boldsymbol{z}}_t^{(k)} \leftarrow \boldsymbol{P}_1 \boldsymbol{z}_t^{(k)} + \bar{\boldsymbol{z}}_t$ for $t = 1, \ldots, T_w$ ▷ *Warm-up*

    $\tilde{\boldsymbol{z}}_t^{(k)} \leftarrow \boldsymbol{P}_\alpha \boldsymbol{z}_t^{(k)} + \alpha \bar{\boldsymbol{z}}_t$ for $t = T_w + 1, \ldots, T$ ▷ *Forced inputs*

    $\boldsymbol{J}_t^{(k)} \leftarrow \frac{\partial F_\theta}{\partial \boldsymbol{z}}\big|_{\tilde{\boldsymbol{z}}_t^{(k)}} \boldsymbol{P}_1$ for $t = 1, \ldots, T_w$ ▷ *Warm-up Jacobians*

    $\boldsymbol{J}_t^{(k)} \leftarrow \frac{\partial F_\theta}{\partial \boldsymbol{z}}\big|_{\tilde{\boldsymbol{z}}_t^{(k)}} \boldsymbol{P}_\alpha$ for $t = T_w + 1, \ldots, T$ ▷ *Forced Jacobians*

    $\boldsymbol{r}_t^{(k)} \leftarrow \boldsymbol{z}_t^{(k)} - F_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_{t-1}^{(k)})$ for $t = 1, \ldots, T$ ▷ *Residuals*

    Solve $\Delta \boldsymbol{z}_t^{(k+1)} = \boldsymbol{J}_{t-1}^{(k)} \Delta \boldsymbol{z}_{t-1}^{(k+1)} - \boldsymbol{r}_t^{(k)}$ for $t = 1, \ldots, T$ via associative scan

    $\boldsymbol{z}_{1:T}^{(k+1)} \leftarrow \boldsymbol{z}_{1:T}^{(k)} + \Delta \boldsymbol{z}_{1:T}^{(k+1)}$

  $\hat{\boldsymbol{x}}_{T_w+1:T} \leftarrow \boldsymbol{B} \boldsymbol{z}_{T_w+1:T}^{(\text{final})}$

  $\mathcal{L} \leftarrow \text{MSE}(\boldsymbol{x}_{T_w+1:T}, \hat{\boldsymbol{x}}_{T_w+1:T})$ ▷ *Loss excluding warm-up states*

  $(\boldsymbol{g}_{\boldsymbol{\theta}}, \boldsymbol{g}_{\boldsymbol{B}}) \leftarrow \text{grad}(\mathcal{L};\ \boldsymbol{\theta}, \boldsymbol{B})$ ▷ *Single associative scan via IFT*

  $(\boldsymbol{\theta}, \boldsymbol{B}) \leftarrow \text{optimize}(\boldsymbol{\theta}, \boldsymbol{B}, \boldsymbol{g}_{\boldsymbol{\theta}}, \boldsymbol{g}_{\boldsymbol{B}})$

until convergence

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract clearly states the challenges that motivated our contributions, the main idea we introduced to address these challenges, and our conclusion about the benefits of our method drawn from our experiments. All claims have structured counterparts in the main text, which faithfully backs them up with evidence.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations, in particular regarding scaling with latent dimension and scope of our proposition, are stated in the main text and are again summarized in a dedicated paragraph in the Discussion section.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The relevant definitions and assumptions for our proposition are clearly stated in the main text, and a proof is provided in the appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The model architectures, algorithms, hyperparameters, regularizations, datasets, and evaluation measures needed for reproducing our results are all clearly and comprehensively described in the main text and appendices.

Guidelines:

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

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: For the review process, the anonymized paper version includes a link to an anonymous repository with our code. We will make our code publicly available upon publication.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Hyperparameter settings for model training, as well as training and evaluation methods, are comprehensively documented in the main text and appendices.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Where applicable, figures include error bars, and the text states confidence intervals. Together with the detailed description of our evaluation measures in the appendix, these elements allow for the critical confirmability of our judgment on the statistical significance of our results.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Details on our compute resources are stated in the appendix on additional experimental details.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our research conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

Justification: Our methods for dynamical systems reconstruction will be valuable to the scientific community working in this field, but we do not expect any broader societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: All of our datasets were generated using mathematical equations, so there is no potential for misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the papers that describe the methods and code upon which our work is based, as well as the papers associated with the code used for comparisons. Where applicable, the appendix on additional experimental details provides URLs and information on licenses under which models have been made available.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The code provided alongside the paper is documented.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: No crowdsourcing experiments or research with human subjects were conducted.

    Guidelines:

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

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

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

    Answer: [N/A]

Justification: Our paper involves no crowdsourcing or research with human subjects.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: Our core method development in this research does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.