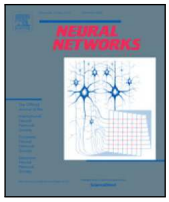
Neural Networks 173 (2024) 106152

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Full Length Article

Data-driven learning of chaotic dynamical systems using Discrete-Temporal Sobolev Networks

Connor Kennedy, Trace Crowdis, Haoran Hu, Sankaran Vaidyanathan, Hong-Kun Zhang *

Department of Mathematics & Statistics, University of Massachusetts, Amherst, MA 01003, USA

ARTICLE INFO

Keywords:
Lorenz system
Neural network
LSTM
Physical Informed Neural Network
Chaotic system
Prediction

ABSTRACT

We introduce the *Discrete-Temporal Sobolev Network (DTSN)*, a neural network loss function that assists dynamical system forecasting by minimizing variational differences between the network output and the training data via a temporal Sobolev norm. This approach is entirely data-driven, architecture agnostic, and does not require derivative information from the estimated system. The DTSN is particularly well suited to chaotic dynamical systems as it minimizes noise in the network output which is crucial for such sensitive systems. For our test cases we consider discrete approximations of the Lorenz-63 system and the Chua circuit. For the network architectures we use the *Long Short-Term Memory* (LSTM) and the Transformer. The performance of the DTSN is compared with the standard MSE loss for both architectures, as well as with the *Physics Informed Neural Network* (PINN) loss for the LSTM. The DTSN loss is shown to substantially improve accuracy for both architectures, while requiring less information than the PINN and without noticeably increasing computational time, thereby demonstrating its potential to improve neural network forecasting of dynamical systems.

1. Introduction

The study of chaotic dynamical systems universally leads to difficulties in numerical analysis and simulation. It is these difficulties in simulation that lead to the initial discovery of the field with Edward Lorenz's seminal work on the Lorenz-63 system (Lorenz, 1963). This simplified fluid convection model was found to have inherent complexities, such as the existence of nonzero Lyapunov exponents (Viswanath, 1998) and the presence of a strange attractor (Sparrow, 1982), which would become hallmarks of what we now understand to be chaotic systems. Despite its simplicity in comparison to other models that have been introduced since Lorenz's work, these properties prove to be so fundamentally challenging to numerical methods that the system still serves as an invaluable example to test new, more powerful computational tools. We also consider the Chua circuit as another test case chaotic dynamical system. The Chua circuit is a rare example of a physically realizable system in a lab setting that is chaotic and can be described via a simple set of ordinary differential equations (ODE) (Galias, 1997). The Chua circuit is also nonlinear via a lack of smoothness in a resistance term rather than via a product as in the Lorenz system, and thus provides a qualitatively different example of a chaotic system to consider.

In this paper, we consider the use of neural network models for dynamical system prediction. While basic feed-forward networks in principle are universal function approximators (Hornik, Stinchcombe, & White, 1989), a variety of different network architectures have been developed to achieve better accuracy on dynamical systems. One such method is the *Long Short-Term Memory* (LSTM) network, a particular form of *Recurrent Neural Network* (RNN) which uses the dependence between sequential data points to better perform predictions (Hochreiter & Schmidhuber, 1997). We also consider the Physics Informed Neural Network (PINN), which uses a loss function that incorporates knowledge of a system's governing equations to better improve training (Raissi, Perdikaris, & Karniadakis, 2017). The Transformer architecture has seen some use in dynamical systems as well, despite initially being developed for natural language processing tasks. The Transformer uses a self-attention scheme where the degree of connection between input data points is learned and stored in the form of attention vectors (Vaswani et al., 2017).

Many variants of these above methods have also been tested on the Lorenz system and related systems to demonstrate their efficacy. The LSTM has been combined with methods of incorporating partially available data to improve predictions on the Lorenz-63 system (Dubois, Gomez, Planckaert, & Perret, 2020). A probabilistic approach referred

* Corresponding author.
E-mail addresses: conmkennedy@umass.edu (C. Kennedy), tcrowdis@umass.edu (T. Crowdis), haoranhu@umass.edu (H. Hu), sankaranv@cs.umass.edu (S. Vaidyanathan), hongkunz@umass.edu (H.-K. Zhang).

https://doi.org/10.1016/j.neunet.2024.106152
Received 28 April 2023; Received in revised form 1 January 2024; Accepted 28 January 2024
Available online 1 February 2024

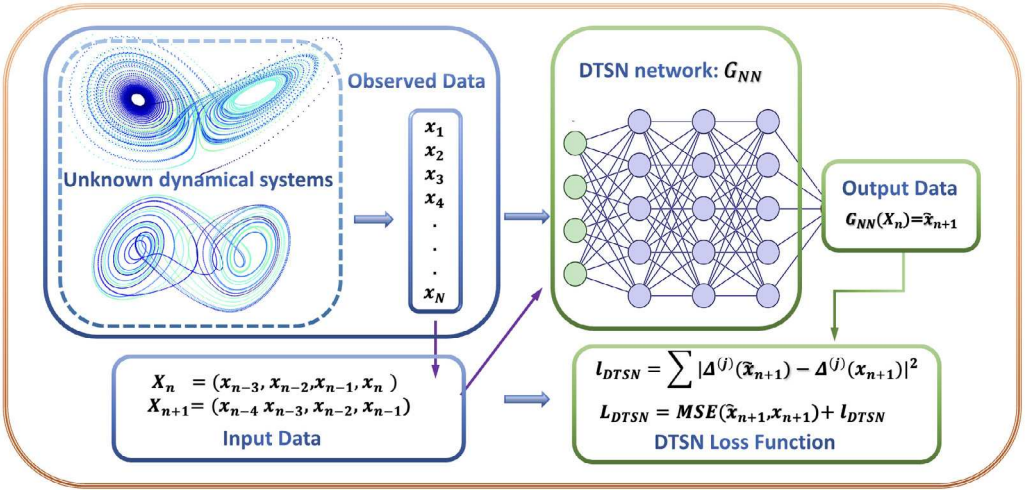


Fig. 1. A basic outline of the first-order DTSN network, which defines a system G_{NN}. The network assigns prediction labels to paired samples of input data independently and then compares the difference to the difference of the true labels. Further details are described in Section 3.1.

to as the *Mean Stochastic Model* has been applied to prevent deviation from the attractor (Vlachas, Byeon, Wan, Sapsis, & Koumoutsakos, 2018). The LSTM has also been compared to other methods such as reservoir computing methods in predictions on the related, but more complicated Lorenz-96 system (Chattopadhyay, Hassanzadeh, & Subramanian, 2020; Lorenz, 1995). There have also been some applications of the PINN to the Lorenz-63 system with notable success (Lu, Meng, Mao, & Karniadakis, 2019). These two methods need not be distinct, and there have been promising examples of using LSTM architectures with PINN loss functions (Zhang, Liu, & Sun, 2020), as well as PINN losses being applied to other RNNs (Yucesan & Viana, 2021). The PINN does however requires knowledge of the underlying governing equations for the given system, which are often unknown. The Transformer's applications to dynamical systems are less explored, but there has been some research (Cholakov & Kolev, 2021), with a few applications to the Lorenz system (Shalova & Oseledets, 2020), and at least one interesting reformulation using a Koopman embedding to process the data (Geneva & Zabaras, 2022). The Chua circuit has also seen some study via not the LSTM but a *Locally Recurrent Neural Networks* to learn about the properties of its attractor (Cannas, Cincotti, Marchesi, & Pilo, 2001), as well as earlier study via *Cellular Neural Networks* (Arena, Baglio, Fortuna, & Manganaro, 1995). The investigation of the Chua circuit via neural networks has been limited overall though.

We propose a new method, the Discrete-Temporal Sobolev Network (DTSN), which is fully data-driven and reduces the need for known prior information about the system. The DTSN includes a loss term that measures the mean square error (MSE) between the difference in successive steps of the network output and those same differences for the true data. A basic overview of the concept is included in Fig. 1. On the local level for a single prediction, this bears similarity to estimation via Taylor polynomials, where knowledge of the derivative at a known point is used to improve estimation. When averaged across a training set this acts similarly to minimization over a Sobolev norm with respect to time. We recall that Sobolev norms act as sums of L^p norms over the derivatives of two functions, measuring the difference in how the functions move as well as their overall values. Our usage of finite differences mimics the concepts of the Sobolev norm but without requiring knowledge of the derivatives from our given data set.

The DTSN also has several features that make it notably robust. It may be easily extended to higher orders by simply looking at higher order differences. These distinct loss terms for different orders may also have their relative importance set as learnable parameters, allowing terms of greater importance to contribute more substantially to the overall loss. The loss function also uses short to moderate lengths orbits of the system for training data, which are often more accessible than large clusters of initial points as needed in some PINN approaches (Lu et al., 2019). By using a collection of multiple short orbits for the system we can still prevent overfitting to a given data set while maintaining the needed data format.

The DTSN can be seen as an extension of RNN based models trained with the PINN loss. Similar to the RNN approach, we wish to use information from the interdependence of successive steps to improve the network's accuracy, only now we look at the interdependence of outputs from the network, rather than inputs from the data set. From the PINN we see that it is valuable to not only match a data set point for point in training, but to appropriately match the behavior of differences (or derivatives) for these points. In contrast with the PINN, the DTSN does not require prior knowledge about the system being modeled. Experiments conducted within this paper demonstrate that the DTSN loss applied to an LSTM architecture actually outperforms the PINN loss applied to this same architecture.

However, evaluating this architecture on a chaotic system presents challenges with data generation. The numerical issues with any chaotic system mean that even with exact knowledge of the equations it is difficult to generate medium to long length trajectories without an exceptionally high number of stored digits and very high order approximation schemes, this is explicitly well-known in the case of the Lorenz system (Estep & Johnson, 2011). One can rely on shadowing lemmas to guarantee generated trajectories match for some unknown initial condition but some chaotic systems like the Lorenz system contains singularities which make the process of verifying these shadows more complicated (Araujo & Pacifico, 2010; Coomes, Koçak, & Palmer, 1994; Hayes & Jackson, 2003). Thus we propose treating a simple discretization of a given chaotic system to be the ground truth itself for testing the network's capabilities. This substitute system still retains all the essential chaotic properties of the flow while being vastly easier to compute sample trajectories. This method of treating a discretization as the ground truth for evaluation of a prediction scheme may be useful for many dynamical systems where there are also difficulties in the generation of data. For such systems though one would likely wish to verify that any crucial qualitative properties of the system (positive Lyapunov exponents, topological equivalence of attractors, etc.) are preserved under the discretization, a process we verify for the Lorenz system in Section 4.1.2.

The overall structure of the paper is given as follows. In Section 2 we give an overview of background information and notation for the paper. Section 2.1 defines the form in which data is batched for our methods as well as outlining notation for all considered loss functions.

Section 2.2 briefly reviews the PINN. In Section 3.1 we explicitly define the DTSN network approach and in Section 3.2 we discuss in greater detail the connection with Sobolev norms. In Section 4.1 we discuss the issues with appropriately generating data for the Lorenz system and introduce an alternative discrete system that circumvents these problems. Section 4.2 briefly introduces the Chua circuit as another considered dynamical system to test the DTSN on. Section 5 outlines how the experiments were set up and how the baselines were defined. Section 6 presents the testing results of all the considered networks and compares them. In Section 7 final discussion and speculation on future directions of research are given. Appendix B provides some technical background on the Lorenz system and Lyapunov exponents while Appendix A reviews the shadowing lemma.

2. Preliminaries

2.1. Data formatting and notation

In this section, we introduce some relevant notation for the data sets considered and loss functions, as well as some key assumptions needed for batching. We consider a discrete map $F_h : \mathbb{R}^d \to \mathbb{R}^d$, which acts as some discrete approximation of continuous chaotic flow $\Phi_t : \mathbb{R}^d \to \mathbb{R}^d$ over a fixed time h. Using this discrete map F_h we generate a length N orbit denoted by:

$$\Gamma_N = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}, \tag{1}$$

where $\mathbf{x}_k = (x_{k,1}, x_{k,2}, \dots, x_{k_m}) \in \mathbb{R}^d$ is the kth location of the orbit. Our goal is to design a neural network G_{NN} to learn the dynamical system F_h, based on the information provided by one or more orbits of the form Γ_N.

We first fix a sample size $m \geq 1$. The samples in orbit Γ_N are grouped into overlapping samples $\mathbf{X}_i$ of m consecutive points. Explicitly this form is given by:

$$\mathbf{X}_i = (\mathbf{x}_i, \dots, \mathbf{x}_{i+m}), \quad 1 \leq i \leq N - m. \tag{2}$$

This collection of samples $\mathbf{X}_i$ is denoted as our data set $\mathcal{D}$. The usage of consecutive data values in the individual samples is necessary for the implementation of the LSTM and Transformer, as both architectures use the relationship between these iterates to improve prediction. We then split the data set $\mathcal{D}$ into the training set $\mathcal{D}_{train}$, validation set $\mathcal{D}_{val}$, and test set $\mathcal{D}_{test}$. For each sample $\mathbf{X}_i = (\mathbf{x}_i, \dots, \mathbf{x}_{i+m}) \in \mathcal{D}_{train}$, we denote the true label as $Y_{true}(\mathbf{X}_i) := F_h(\mathbf{x}_{i+m})$.

We randomly group samples $\mathbf{X}_i$ into batches denoted by $\mathcal{A}_{batch}$ in the usual manner for performing stochastic gradient descent (Zinkevich, Weimer, Li, & Smola, 2010). Our goal is to design an effective loss function $\mathcal{L}(\mathcal{A}_{batch}) := \mathcal{L}(\{G_{NN}(\mathbf{X}_i), Y_{true}(\mathbf{X}_i) : \mathbf{X}_i \in \mathcal{A}_{batch}\})$, which is defined on the predicted and true labels for a given batch. We compare this to the standard mean square error (MSE) loss. We recall the MSE loss measures the distance between the labels $G_{NN}(\mathbf{X}_i)$ and $Y_{true}(\mathbf{X}_i)$ by the following:

$$\mathcal{L}_{MSE}(\mathcal{A}_{batch}) := \frac{1}{|\mathcal{A}_{batch}|} \sum_{\mathbf{X}_i \in \mathcal{A}_{batch}} \|G_{NN}(\mathbf{X}_i) - Y_{true}(\mathbf{X}_i)\|_2^2. \tag{3}$$

Here $|\mathcal{A}_{batch}|$ is the cardinality of the batch. To implement our DTSN loss, as well as the PINN loss, we will need to constrain the types of batches considered. Any given batch $\mathcal{A}_{batch}$ must be able to be divided into sub-batches of the form $\mathbf{A}$, such that

$$\mathbf{A} = \{\mathbf{X}_i, \mathbf{X}_{i+1}\} = \{(\mathbf{x}_i, \dots, \mathbf{x}_{m+i}), (\mathbf{x}_{i+1}, \dots, \mathbf{x}_{m+i+1})\}. \tag{4}$$

We also define the set $\mathcal{A}_{batch,1}$ as:

$$\mathcal{A}_{batch,1} = \{\mathbf{X}_i \in \mathcal{A}_{batch} : \mathbf{X}_{i+1} \in \mathcal{A}_{batch}\}. \tag{5}$$

We thus have that for each sample $\mathbf{X}_i$ $in \mathcal{A}_{batch}$ we must include the sample immediately before it or after it. The set $\mathcal{A}_{batch,1}$ simply denotes all the samples from the orbit for which the next sample is still in the same batch, in order to avoid having to consider two cases in later definitions. We see this grouping together of samples into batches mirrors the grouping of points into samples from Eq. (2). Thus the similarity between RNN requirements on the input data structure and the DTSN requirement on the output data structure is made explicit.

Lastly, we generalize this notion of batching for cases where we wish to observe dependencies between l sequential outputs. In this case, we would require batches to be capable of being divided into sub-batches $\mathbf{A}$ of the form

$$\mathbf{A} = \{\mathbf{X}_i, \mathbf{X}_{i+1}, \dots \mathbf{X}_{i+l}\}. \tag{6}$$

Now the batches used for training the network always come in blocks that produce l sequential labels. We also define the set $\mathcal{A}_{batch,j}$ analogously to $\mathcal{A}_{batch,1}$ by

$$\mathcal{A}_{batch,j} = \{\mathbf{X}_i \in \mathcal{A}_{batch} : \mathbf{X}_{i+1}, \dots \mathbf{X}_{i+j} \in \mathcal{A}_{batch}\}. \tag{7}$$

Finally, we remark that if the network produces an output of l time steps for a given sample then no special formatting of the batches is needed, we already have successive iterates available to the loss function. We will work with networks that only perform prediction for one step at a time within this paper for simplicity, but we leave this remark to note the ease of implementation in cases of longer prediction times.

2.2. Physics informed neural network

We now briefly review the loss function from the Physics Informed Neural Network (PINN), with a slight modification we impose to work with systems of difference equations, like those used in discretizations of continuous systems considered in Section 4.1. A PINN is a neural network whose loss function has been modified to respect a physical law that the studied system is assumed to obey. Assume we have a d-dimensional ODE system (an analogous construction for partial differential equations PDEs also exists) given by Raissi et al. (2017):

$$G_{ODE}(\mathbf{x}) = \dot{\mathbf{x}} + F_{ODE}(\mathbf{x}) = \mathbf{0}. \tag{8}$$

Then, in addition to the standard MSE loss function, $\mathcal{L}_{MSE}$, a second term is included with the following form:

$$\mathcal{L}_{P,1}(\mathcal{A}_{batch}) = \frac{1}{|\mathcal{A}_{batch}|} \sum_{\mathbf{X}_i \in \mathcal{A}_{batch}} |G_{ODE}(G_{NN}(\mathbf{X}_i))|^2. \tag{9}$$

With total loss given by

$$\mathcal{L}_{PINN,1}(\mathcal{A}_{batch}) = \mathcal{L}_{MSE}(\mathcal{A}_{batch}) + \mathcal{L}_{P,1}(\mathcal{A}_{batch}). \tag{10}$$

The $\mathcal{L}_{P,1}$ term measures how well the output of the network adheres to the governing equations of the system. This type of loss function has been shown to be particularly powerful for systems where underlying equations are known (Cuomo et al., 2022). Because this is solely a modification of the loss function, it can be paired with several types of neural network architectures. In particular, its combination with LSTM networks, called the LSTM-PINN, is one of several examples that have had notable success (Yucesan & Viana, 2021; Zhang et al., 2020). To implement this method on a discretized dynamical system, we will need to modify it slightly, as the underlying system we wish the network to learn is given by a system of difference equations. Assume $F : \mathbb{R}^d \to \mathbb{R}^d$ is a known function defining a system of difference equations by

$$\mathbf{x}_{n+1} = F(\mathbf{x}_n). \tag{11}$$

We assume a data set $\mathcal{D}$ with sub-batches of the form in Eq. (4) when we perform stochastic gradient descent. We also define a pair of hyperparameters $\lambda_1, \lambda_2 > 0$ to act as weights. We then finally may define $\mathcal{L}_P$, a discrete version of $\mathcal{L}_{P,1}$, by:

$$\mathcal{L}_P(\mathcal{A}_{batch,1}) = \frac{1}{|\mathcal{A}_{batch,1}|} \sum_{\mathbf{X}_i \in \mathcal{A}_{batch,1}} |(f(G_{NN}(\mathbf{X}_i)) - G_{NN}(\mathbf{X}_{i+1}))|^2. \tag{12}$$

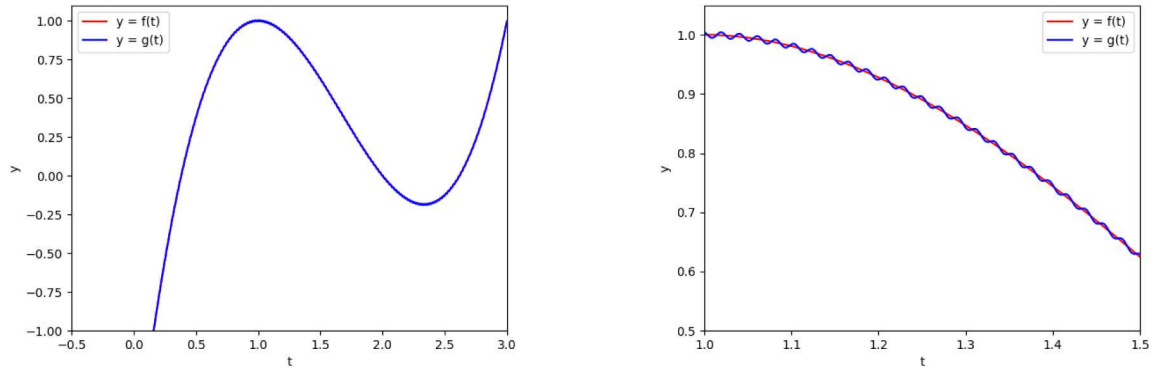


Fig. 2. Left: Here we compare a test signal of a simple polynomial $f(x) = (x-2)^3 + (x-2)^2 - x + 2$ with a signal perturbed by $p(x) = 0.005\sin(300x - 0.7)$. Right: The perturbed signal has very rapid oscillation despite the closeness in the L^2 norm. These are the perturbations that the Sobolev norm picks up.

With total loss given by

$$\mathcal{L}_{PINN}(\mathcal{A}_{batch}) = \lambda_1 \mathcal{L}_{MSE}(\mathcal{A}_{batch}) + \lambda_2 \mathcal{L}_P(\mathcal{A}_{batch}). \tag{13}$$

We see this is essentially the same as the definition for $\mathcal{L}_{P,1}$, but for a system of difference equations. We note that while in realistic settings we could use knowledge of the equations to directly perform predictions, the usage of this term is primarily to serve as a point of comparison to the DTSN. We shall see in Section 6 that the LSTM-PINN outperforms the LSTM-MSE network, and variation of the weights λ_1 and λ_2 during training can improve the accuracy further.

3. The discrete-temporal Sobolev neural network

3.1. Defining the DTSN neural network

We now define the DTSN loss function. We assume the sub-batch restriction of (4), which allows the relationship between outputs of the network to be used in our loss function. We then define Δa_i on a sequence $\{a_i\}_{i=1}^N$ by $\Delta a_i = a_{i+1} - a_i$. We may now define our first new loss function term, $\mathcal{L}_{D_1}$, by:

$$\mathcal{L}_{D_1}(\mathcal{A}_{batch}) = \frac{1}{|\mathcal{A}_{batch,1}|} \sum_{\mathbf{X}_i \in \mathcal{A}_{batch_1}} |\Delta G_{NN}(\mathbf{X}_i) - \Delta Y_{true}(\mathbf{X}_i)|^2. \tag{14}$$

We see that this is similar to the form of $\mathcal{L}_P$, but $\mathcal{L}_{D_1}$ compares how the differences of network outputs match the differences of the training data, rather than how they match a set of known underlying equations. We call this network utilizing this new loss term a Discrete-Temporal Sobolev Network (DTSN), due to it functioning as an approximation of a discrete-temporal Sobolev norm (further details are outlined in Section 3.2). We construct the total loss function $\mathcal{L}_{DTSN,1}$, in a similar manner to $\mathcal{L}_{PINN}$ with similar weighting terms λ_1, λ_2:

$$\mathcal{L}_{DTSN,1}(\mathcal{A}_{batch}) = \lambda_1 \mathcal{L}_{MSE}(\mathcal{A}_{batch}) + \lambda_2 \mathcal{L}_{D_1}(\mathcal{A}_{batch}). \tag{15}$$

This narrows the field of possible solutions by disallowing examples where the data points are matched closely but the rate of change $|\Delta G_{NN}(\mathbf{X}_i) - \Delta Y_{true}(\mathbf{X}_i)|$ is substantially off (see Fig. 2 for an example). Preventing these inaccurate fluctuations can also reduce the accumulation of small errors, which we know for a chaotic system can rapidly build over the course of estimation. In addition, we note that the DTSN loss is fully data-driven, removing the substantial requirements of prior knowledge of the PINN. Eq. (15) describes the first order form of the DTSN, which we show in Section 6 notably improves prediction accuracy.

We now draw some parallels between the DTSN and the classic Taylor expansion to give some intuition to the DTSN's capabilities as well as how it may be generalized to higher orders. We recall that if a function $f : \mathbb{R} \to \mathbb{R}$ is l times differentiable at point a, then there exists $q_l : \mathbb{R} \to \mathbb{R}$ with $\lim_{x \to a} q_l(x) = 0$ such that:

$$f(t) = f(a) + f'(a)(t-a) + \frac{f''(a)}{1}(t-a)^2 + \cdots \frac{f^{(l)}(a)}{l!}(t-a)^l + q_l(t)(t-a)^l. \tag{16}$$

We also consider a slight modification of the theorem to better fit our current situation. Given an orbit of points $\Gamma = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ for a map Φ_h (the fixed time h map for a flow Φ_t) we see that

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \partial_t \Phi_t(\mathbf{x}_{n-1})(h) + \frac{\partial_t^2 \Phi_t(\mathbf{x}_{n-1})}{2}(h)^2 + \cdots \frac{\partial_t^l \Phi_t(\mathbf{x}_{n-1})}{l!}(h)^l + q_l(t)(h)^l. \tag{17}$$

Our initial form of the loss $\mathcal{L}_{D_1}$ can be thought of as similar to the first-order version of the expansion, going only up until the $\partial_t \Phi_t$ term. With this viewpoint, we see a natural way to define a higher order version of the loss. This requires the stronger restrictions on sub-batches given by Eq. (6) for some chosen l. With this restriction, we can apply a discrete equivalent of the lth derivative on a sequence $\{a_i\}_{i=1}^N$ to the samples. This discrete derivative is given by:

$$\Delta^{(l)} a_i := \Delta^{(l-1)} a_{i+1} - \Delta^{(l-1)} a_i, \qquad \Delta^{(1)} a_i := a_{i+1} - a_i. \tag{18}$$

We can then define $\mathcal{L}_{D_j}$, a generalization of $\mathcal{L}_{D_1}$, by:

$$\mathcal{L}_{D_j}(\mathcal{A}_{batch}) = \frac{1}{|\mathcal{A}_{batch,j}|} \sum_{\mathbf{X}_i \in \mathcal{A}_{batch,j}} |\Delta^{(j)} G_{NN}(\mathbf{X}_i) - \Delta^{(j)} Y_{true}(\mathbf{X}_i)|^2. \tag{19}$$

Finally, with a selection of weights $\{\lambda_j\}_{j=0}^{j=l}$, the total loss function for the lth order DTSN is given by:

$$\mathcal{L}_{DTSN} = \sum_{\mathcal{A}_{batch}} \lambda_0 \mathcal{L}_{MSE}(\mathcal{A}_{batch}) + \sum_{j=1}^{l} \lambda_j \mathcal{L}_{D_j}(\mathcal{A}_{batch}). \tag{20}$$

We see this allows us to further narrow the range of explored solutions by only allowing possibilities that closely match all differences up to order l. This may prove especially crucial for systems where there is an underlying differentiable system that has large higher order derivatives. In that case substantial future changes may be predicted by high values of 3rd to 4th-order derivatives that have yet to be clearly visible in the data points directly or the first derivative. We found that the simpler loss function in (15) was sufficient for prediction on our considered dynamical systems, but we still introduce this more general framework here to be considered for future study on other dynamical systems.

3.2. Analysis of the DTSN neural network

In this section we provide a deeper analysis of the DTSN. We begin by considering the usual pointwise MSE loss, which can be thought of

as trying to make the network converge to the true underlying map F in the simple L^2 norm; here we assume the considered phase space is $\Omega \subset \mathbb{R}^d$ equipped with normalized Lebesgue measure μ.

$$\|Y_{true} - G_{NN}\|_{L^2}^2 := \int_\Omega |Y_{true}(\mathbf{X}_i) - G_{NN}(\mathbf{X}_i)|^2 \mu. \tag{21}$$

Because we do not have information about F over the entire phase space, we use the MSE loss to approximate this integral by sampling a collection of known points via generated trajectories. We must then extrapolate from our limited trajectory data to guide predictions of the behavior across the whole space. The MSE loss crucially does not see any relationship between separate outputs of the network. While the learning process may consider multiple outputs simultaneously, it does not treat the outputs $G_{NN}(\mathbf{X}_1), G_{NN}(\mathbf{X}_{100})$ as being any more or less related than the outputs $G_{NN}(\mathbf{X}_1), G_{NN}(\mathbf{X}_2)$ for a given collection of samples. In contrast, the DTSN loss function utilizes the relationships between successive outputs.

We now consider the first-order DTSN loss. The simple MSE loss minimized the difference between the network and the true underlying system in the L^2 norm. Is there a corresponding norm or semi-norm which the DTSN loss minimizes distance in? We will show that the DTSN minimizes distance in a discrete form of the Sobolev time–space semi-norm (Abdeljawad & Grohs, 2022). We first consider the standard Sobolev norm to build up to this more complex notion. The definition of the standard Sobolev norm is given by:

Definition 1. Assume Ω is an open subset of $\mathbb{R}^d$, $d \in \mathbb{N}$, $1 \le p \le \infty$. The Sobolev space $W^{k,p}(\Omega)$ is the set of all functions $f \in L^p(\Omega)$ such that for any multi index α with $|\alpha| \le k$, $D^\alpha f$ exists and $D^\alpha(f) \in L^p(\Omega)$. We recall that D^α is defined as:

$$\left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \left(\frac{\partial}{\partial x_2}\right)^{\alpha_2} \cdots \left(\frac{\partial}{\partial x_d}\right)^{\alpha_d}. \tag{22}$$

The Sobolev norm is then defined as

$$\|f\|_{W^{k,p}} := \Big(\sum_{0\le|\alpha|\le k} (\|D^\alpha f\|_{L^p(\Omega)}^p)\Big)^{\frac{1}{p}}. \tag{23}$$

Before we move on to more complex formulations, we first explore what information the Sobolev norm has that the standard L^p norm does not. The metric defined by the Sobolev norm measures not only the difference between two functions, but also how close their rates of change are. We note that closeness in the Sobolev norm immediately implies closeness in the corresponding L^p norm. Meanwhile one can simply construct an example even in the 1-D case of two functions that are arbitrarily close in the $\|\cdot\|_{L^2}$ norm and arbitrarily far in the $\|\cdot\|_{W^{1,2}}$ norm.

Consider a function $f(t) \in C^1$ on the interval $[0, T]$ and a periodic function $p(t) \in C^1$ with period T. Let $\epsilon > 0$ be a small constant and m_0 be a large positive integer. We then consider $g(t) = f(t) + \epsilon p(m_0 t)$. It is simple to see that

$$\|f - g\|_{L^2} = \|\epsilon p(m_0 t)\|_{L^2} \le \epsilon C_1, \tag{24}$$

here C_1 is a constant based on p. Meanwhile, we have that

$$\|f' - g'\|_{L^2}^2 = \epsilon m_0 \|p'(m_0 t)\| \ge \epsilon m_0 C_2. \tag{25}$$

Where C_2 is another constant based on p'. So if the perturbation is small in magnitude, but high in frequency, then $g(t)$ is close to $f(t)$ in the $\|\cdot\|_{L^2}$ norm while being far in the $\|\cdot\|_{W^{1,2}}$ norm. This more rigorously quantifies the notion that to be close in the Sobolev norm an approximation must not oscillate too frequently. We construct a specific example of a function and corresponding perturbed function in this way in Fig. 2. We also note that searching for an approximation via this stronger norm may also prevent trapping in local minima, as a local minimum in L^2 may not be a local minimum in $W^{1,2}$ but again the reverse cannot happen as $\|\cdot\|_{W^{1,2}}$ is a stronger norm than $\|\cdot\|_{L^2}$.

Now that we have illustrated the significance of the norm for our purposes, we begin to broaden the norm to match the context of our dynamical systems. We first extend this definition in the following way (Evans, 2022).

Definition 2 (*Sobolev and Bochner Space*)**.** Assume Ω is an open subset of $\mathbb{R}^d$, $d \in \mathbb{N}$, $1 \le p \le \infty$, and B is some Banach space. The Bochner space $L^p(\Omega, B)$ is defined as the set of all functions $F : \Omega \to B$ such that the following norm is finite on them.

$$\|F\|_{L^p(\Omega,B)} = \|(\|F\|_B)\|_{L^p(\Omega)}. \tag{26}$$

The corresponding Sobolev space $W^{k,p}(\Omega, B)$ is then defined as the set of functions $F \in L^p(\Omega, B)$ such that for α with $|\alpha| \le k$, $D^\alpha F$ exists and $D^\alpha(F) \in L^p(\Omega, B)$. The expanded Sobolev norm is then simply defined by

$$\|F\|_{W^{k,p}(\Omega,B)} := \Big(\sum_{0\le|\alpha|\le k} (\|(\|D^\alpha F\|_B)\|_{L^p(\Omega)}^p)\Big)^{\frac{1}{p}}. \tag{27}$$

The notation will be written simply as $\|F\|_{W^{k,p}}$ if there is no ambiguity for the codomain B.

We first use this when the codomain B is $\mathbb{R}^d$, which allows us to apply Sobolev norms to vector valued functions. More complex codomains will be considered when we approach the time–space norm.

For a dynamical flow we first consider the simple case where we fix an initial point $\mathbf{x}_0$. We may then consider the flow $\boldsymbol{\Phi}_t$ to be a function mapping from a considered finite time interval $[0, T]$ $T \in (0, \infty)$ to the space $\mathbb{R}^d$. In this context we may consider the following Sobolev norm

$$\|\boldsymbol{\Phi}_t\|_{W^{1,2}([0,T],\mathbb{R}^d)}^2 = \frac{1}{T}\int_0^T \|\boldsymbol{\Phi}_t(\mathbf{x}_0)\|^2 dt + \frac{1}{T}\int_0^T \|\partial_t \boldsymbol{\Phi}_t(\mathbf{x}_0)\|^2 dt. \tag{28}$$

Our network is estimating a discrete approximation of a flow though. In this case, we may modify (28) to a discrete form where we denote $I_N := \{1, 2, \ldots, N\}$ to be the considered discrete time interval.:

$$\|F\|_{W^{1,2}(I_N,\mathbb{R}^d)}^2 = \frac{1}{N}\sum_{i=0}^N \|F^i(\mathbf{x}_0))\|^2 + \frac{1}{N}\sum_{i=0}^N \|\Delta(F^i(\mathbf{x}_0))\|^2. \tag{29}$$

We see this exactly corresponds to the DTSN loss function (15) for a single batch consisting of one orbit so we have a simple context where the DTSN loss function and a Sobolev norm correspond well. We see this notion of closeness in terms of variations still holds with the discrete form, so long as the variations are not to a degree substantially smaller than our scale of discretization.

We wish to consider arbitrary starting points over a finite region though, and thus the 1-D domain is not fully suited for our application. In the complete context we have a separated time–space domain for our system with temporal derivatives. When we consider the full phase space we introduce the Sobolev time–space semi-norm presented in Abdeljawad and Grohs (2022).

Definition 3 (*Sobolev Time–Space*)**.** Let $1 \le p, q \le \infty, m, n \in \mathbb{N}, I \subset\subset \mathbb{R}$, and $\Omega \subset\subset \mathbb{R}^d$. Let $W_{m,q}^{n,p}(I, \Omega)$ defined as follows

$$W_{m,q}^{n,p}(I,\Omega) = \{F \in L^q(I, W^{n,p}(\Omega)) : \partial_t^k F \in L^q(I, W^{n,p}(\Omega)), \forall k \le m\}. \tag{30}$$

The corresponding semi-norm is then defined as

$$\|F\|_{W_{m,q}^{n,p}(I,\Omega)} := \sum_{k\le m} \|\partial_t^k F\|_{L^q(I,W^{n,p}(\Omega))} = \sum_{k\le m} \|(\|\partial_t F\|_{W^{n,p}(\Omega)})\|_{L^q(I)}. \tag{31}$$

Now we see the significance of defining Bochner spaces in Definition 2. For each chosen length of time considered, we can treat a flow $\boldsymbol{\Phi}_t$ as mapping times from a chosen interval I to maps acting on the space Ω. Each such map has a $W^{n,p}$ norm, and those norms are then integrated with respect to t. We present our discrete-temporal form of this below:

Definition 4 (*Discrete-Temporal Sobolev*)**.** Assume $F : \Omega \to \mathbb{R}^d$ is a map such that $F \in W^{n,p}(\Omega, \mathbb{R}^d)$. We may then define the discrete-temporal Sobolev norm on this map for up to $N \in \mathbb{N}$ iterations by

$$\|F\|_{\bar{W}^{n,p}_{m,q}(I_N,\Omega)} = \sum_{k\leq m} \frac{1}{N} \Big(\sum_{j=1}^{N} \|\Delta^{(k)} F^{(j)}\|^q_{W^{n,p}(\Omega)}\Big)^{1/q}. \tag{32}$$

If we do not wish to consider spatial derivatives then this simplifies to

$$\|F\|_{\bar{W}^{0,p}_{m,q}(I_N,\Omega)} = \sum_{k\leq m} \frac{1}{N} \Big(\sum_{j=1}^{N} \|\Delta^{(k)} F^{(j)}\|^q_{L^p(\Omega)}\Big)^{1/q}. \tag{33}$$

We see the DTSN loss function (20) approximates Eq. (33) in the case of $\bar{W}^{0,2}_{l,1}(\{1\}, \Omega)$, if there is an appropriately large number of sample points used to approximate the L^2 norms. The norm could be approximated for varying values of N by averaging the DTSN loss applied to F^j for multiple values of j. This could be used to construct a network that attempts to directly allow approximations for higher iterates of the map without having to feed outputs of the network back to the network. We do not pursue this method within the present paper to focus on verifying the core concepts of the DTSN but it is an interesting direction for further research. If spatial derivative information is available, one could also simply incorporate that to allow approximation of the norm in Eq. (32).

With these definitions in place, we compare our approach to the existing work in applying Sobolev norms to neural networks. The work of Abdeljawad and Grohs (2022) considers the ability of a network to act as a universal approximator in the norm of (30), proving well-posedness similar to the classic universal approximator property of multilayer networks proven in Hornik et al. (1989). The proof of Abdeljawad and Grohs (2022) does not propose a particular method of training to achieve such approximation though as we are. In the Sobolev Neural Network (SNN) (Czarnecki, Osindero, Jaderberg, Swirszcz, & Pascanu, 2017), there is no discretization of a flow, just immediately the system is already a map, the derivatives are spatial rather than temporal, and derivative data is assumed to be known. Our method differs from the SNN most crucially in that last way, in how little information is assumed to be known; only the trajectory is necessary, with no direct information on any derivatives. The DTSN loss can also be applied to systems which are simply a map from a space to itself rather than a discretization for a flow, still using the method of differences between successive iterations. Some substantial complications may arise in such systems, as generally discrete chaotic systems have singularities, often dense singularities, and thus methods stochastic gradient descent suffer from instability issues. There may still be useful ways to apply the DTSN concept to such systems, which we hope to expand on in future work. Overall the DTSN acts as a broadly viable method of applying concept of Sobolev theory to the loss function of a neural network.

We additionally see how the DTSN contrasts with the PINN. While there is an obvious advantage that the DTSN requires less knowledge of the system, there is also the subtle improvement over the PINN's tendency to "violate causality" (Wang, Sankaran, & Perdikaris, 2022). This is a problem where PINNs tend to match later portions of a trajectory but not earlier steps, creating inconsistency in its estimates of trajectories. The DTSN however enforces consistency from step to step, and thus we expect it to be especially well-suited to preserving causality. Indeed, even the first order DTSN can still see dependencies between any two points in a trajectory via a chain of one step dependencies, and thus we expect that perturbations in the early steps should appropriately be reflected in the network's prediction of later steps. There are still many questions about the nature of the DTSN loss function and the work done so far points to some exciting possibilities for further exploration.

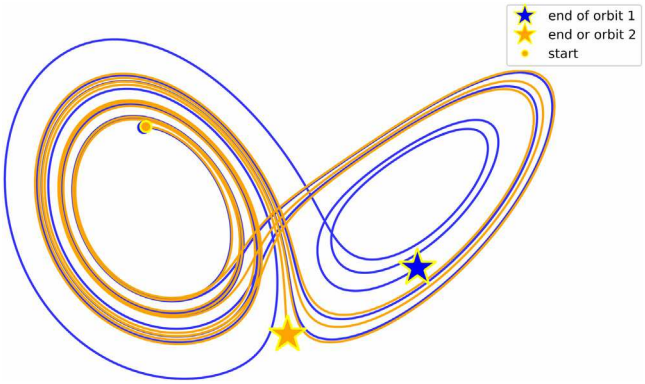


Fig. 3. Plot of solutions near the Lorenz attractor, using the same initial point $(-10, -7, 35)$, generated by two numerical methods from scipy.integrate package: `odeint` for orbit 1 and `solv_iv` for orbit 2.

4. Considered dynamical systems

4.1. Discrete Lorenz map

4.1.1. The Lorenz flow

The Lorenz-63 system of temporal variable t, three spatial variables x, y, z and parameters σ, ρ, $\beta > 0$, is given by Lorenz (1963):

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= x(\rho - z) - y, \\ \frac{dz}{dt} &= xy - \beta z. \end{aligned} \tag{34}$$

We recall that the system is chaotic for certain values of the parameters, and throughout this paper we take them to be the classic example $(\sigma, \rho, \beta) = (10, 28, 8/3)$. For a further review of the system's properties please refer to Appendix B. In order to test the prediction accuracy of an approximation scheme, we first must generate a known orbit Γ for the system. This however proves to be problematic due to the chaotic nature of the system. We first demonstrate this with a specific example.

We choose an initial point, $P = (-10, -7, 35)$, and simulate the flow from this point for 2000 iterations with time step $h = 0.01$ using the Lorenz flow equation. This is done via two different approximation methods for solving differential equations provided by the SciPy package in Python: `odeint` and `solv_iv` (Virtanen et al., 2020). The first algorithm, `odeint`, acts as a first-order initial value problem solver for the Lorenz system. This function is built on LSODA from FORTRAN's ODEPACK which dynamically switches between the non-stiff Adams–Moulton method and the stiff Backwards Differentiation Formula method with default relative tolerance of 1e-3 and absolute tolerance e^{-6} (Petzold, 1983). Meanwhile, the default integration method of `solv_iv` is the Runge–Kutta–Fehlberg method (RK45).

For $t \in (0, 10)$, we can see these two solutions arrive at completely different points after 400 of the 2000 iterations, see Fig. 3. We see in Fig. 4 that even as early as $t = 5$ we get vastly different values for the position of the particle between the two approximation schemes. These findings are consistent with general results on the difficulty of consistent generation of orbits for the Lorenz system (Estep & Johnson, 2011).

This is a fundamental difficulty in the study of chaotic dynamical systems, especially flows. To simulate a given flow Φ_t, we are restricted to estimating repeated applications of the flow over a fixed time interval h, given by Φ_h. The exact form of Φ_h cannot be analytically found, so we must consider some form of approximation F_h. We may consider this approximation to essentially be a perturbation of the original time h map Φ_h.

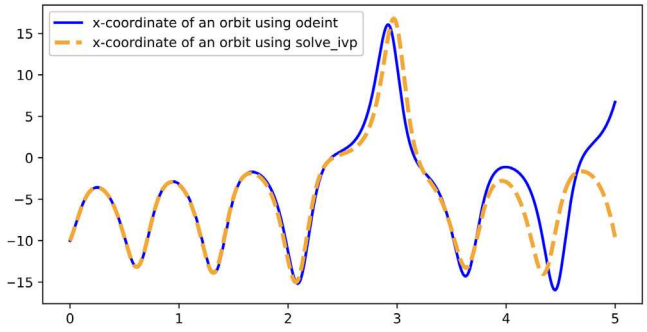


Fig. 4. Plot of the x-coordinate solutions of the Lorenz system, using the same initial condition $(-10, -7, 35)$, generated by two numerical methods from scipy.integrate package: `odeint` (blue) and `solv_iv` (orange).

This creates potential issues with the validity of available machine learning methods when using a data set that is artificially generated from simulations of the Lorenz system or any chaotic dynamical system. Improvements in computability can be made with higher order approximations and more preserved digits, however, the computational cost becomes very expensive (Estep & Johnson, 2011). One could potentially appeal to a form of the shadowing lemma in this case, which we further describe in Appendix A. A form of the lemma suggests that the two approximation methods considered here may both be close to true orbits, but for two distinct starting points, which may both be different from the true starting point $(-10, -7, 35)$. Even with the shadowing lemma however, for the Lorenz system such approximated trajectories that "shadow" true trajectories may only do so for a limited length of time, dependent on the error at each step (Hayes & Jackson, 2003).

We consider the following general framing for the usage of artificially generated data used to benchmark a neural network's capabilities. Assume a chaotic flow Φ_t and an approximation scheme F_h of the time h map Φ_h. We assume that the following holds for some given function g.

$$\|\Phi_h - F_h\| < O(g(h)) \tag{35}$$

Here $\|\cdot\|$ is simply whatever norm is considered most relevant, usually the supremum norm, and $g(h)$ denotes whatever form the considered error has, usually a power of h. If testing shows the neural network can learn the approximation F_h up to order $O(\tilde{g}(h))$ for $0 \leq \tilde{g} \leq g$, we have:

$$\|\Phi_h - G_{NN}\| \leq \|\Phi_h - F_h\| + \|G_{NN} - F_h\| < O(g(h)) + O(\tilde{g}(h)) = O(g(h)). \tag{36}$$

Thus the testing can only tell us if the network can approximate the original system better than F_h. Therefore the network is inherently limited by the accuracy of the discretization scheme used to train the network.

To circumvent these difficulties, we propose an alternative conceptualization for empirically evaluating learned models of a dynamical system. We take a given discrete approximation F_h, to be the ground truth on which we test the network. We thus have a direct form for this discrete system and are only limited by floating point arithmetic error in generating our data set; this is further addressed in Appendix B. As will be shown in Section 4.1.2, this approximate system still preserves the most crucial properties in the Lorenz system case, and thus still serves as a useful method to perform benchmark testing for a predictive neural network. This conceptualization is ultimately important to the study of any chaotic flow, as these complications are in no way unique to the Lorenz system. Any attempt at generating training data from a chaotic flow model encounters these same issues and thus requires careful consideration of what particular approximation scheme is chosen, as that approximation is what the network is technically learning from in artificial data testing.

4.1.2. Discretizing the Lorenz system

We choose the following simple map $F_h : \mathbb{R}^3 \to \mathbb{R}^3$ to act as our discrete approximation of the Lorenz system, with $h = 0.01$ fixed throughout the paper. The map is given by $(x_1, y_1, z_1) = F_h(x_0, y_0, z_0)$, with (x_1, y_1, z_1) defined as:

$$\begin{aligned} x_1 &= x_0 + \sigma(y_0 - x_0)h, \\ y_1 &= (1 - h)y_0 + x_0(\rho - z_0)h, \\ z_1 &= (1 - h\beta)z_0 + hx_0y_0. \end{aligned} \tag{37}$$

We can see that this is the simple Euler approximation for Eqs. (34), and thus F_h converges to Φ_h in the small h limit. We can verify that all of the major dynamical properties of the Lorenz flow are still preserved under this simple, first-order approximation. The Lyapunov spectrum of the system is estimated to be $1.041, -0.001, -14.992$ (for further details, see Appendix B). The existence of a positive Lyapunov exponent crucially tells us that the system is still chaotic, while the existence of a negative exponent tells us that the time-reversed version of the system is chaotic. The sum of the exponents is negative, meaning that the system is dissipative and any collection of initial conditions has its volume tend to zero under repeated iterations of the map. These comprise the most important local properties of the Lorenz flow.

The system also preserves the attractor of the system, at least on the topological level. The discretization is known to have an attractor that is topologically equivalent to that of the original Lorenz system for $h < 0.0265$ (see Eq. (18) of Letellier & Mendes, 2005). This attractor is close to the case for the flow with the alternate set of chaotic parameter values $(80, 3, 0.25)$. Thus the discrete system retains all properties which are dependent on the topological character of the attractor.

While the choice of the Euler approximation was made due to the simplicity with which these properties could be verified, we note that other approximations such as the Runge–Kutta method could also be considered for the Lorenz system or other systems. The significance is in verifying certain properties of the chosen approximation scheme.

4.2. Chua circuit

We also consider the Chua circuit as an example of a chaotic system to test the DTSN on. The Chua circuit is the first example of a broader category of systems referred to as "Multiscroll Attractors" (Galias, 1997; Lü & Chen, 2006). The name "Multiscroll" is due to the shape of trajectories looking like multiple discs, somewhat similar to the Lorenz system, though not equivalent. The Chua circuit is explicitly the simplest case of an electrical circuit meeting the basic criteria to display chaotic behavior (Kennedy, 1993).

- one or more nonlinear elements,
- one or more locally active resistors,
- three or more energy-storage elements.

The circuit exists as a more concrete example of chaos than the Lorenz system, as it is a physical system which can be precisely described via a simple ODE system with chaotic behavior, as opposed to the Lorenz system acting as a heavily simplified model of an atmospheric system. The original Chua Circuit is described physically in Fig. 5.

The governing ODE system is then given by the following: (Lü & Chen, 2006)

$$\frac{dv_{C_1}}{dt} = \frac{1}{RC_1}(v_{C_1} - v_{C_2}) - \frac{1}{C_1}g(v_{C_1}), \tag{38}$$

$$\frac{dv_{C_2}}{dt} = \frac{1}{RC_2}(v_{C_1} - v_{C_2}) - \frac{1}{C_1}i_L, \tag{39}$$

$$\frac{di_L}{dt} = -\frac{1}{L}v_{C_2}. \tag{40}$$

Constants C_1, C_2 are the capacitance of each capacitor, R the resistance of the linear resistor, L the inductance. The function g is given by

$$g(v_{C_1}) = m_1 v_{C_1} + \frac{1}{2}(m_0 - m_1)(|v_{C_1} - b_1| + |v_{C_1} - b_2|). \tag{41}$$

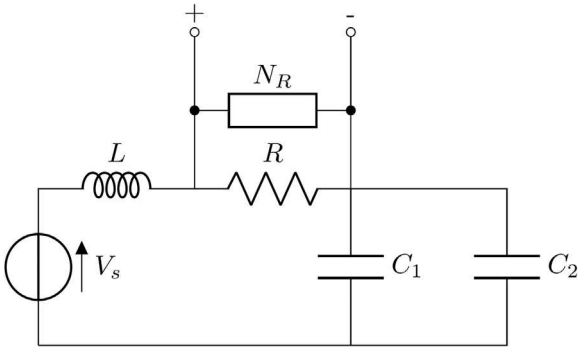


Fig. 5. We demonstrate a simple diagram of the basic Chua Circuit. L is an inductor, C_1, C_2 are capacitors, R a resistor, N_R a nonlinear resistor, and V_s is the source voltage for the system.

The simple nonlinearity of g, being given by a piecewise linear function, is enough to create chaotic dynamics for this system as shown through positive topological entropy of the Poincare map (Galias, 1997). It also has estimated positive Lyapunov exponents (Parlitz, 1993) and a variety of other analyses of its chaotic behavior have been conducted such as bifurcation analyses (Madan, 1993).

We present the above equations to intuitively show how the governing equations arise from the circuit, though in simulations we use a change of coordinates to an equivalent system which is easier to work with. Analyzing the circuit using Kirchhoff's circuit laws, the dynamics of Chua's circuit can be accurately modeled in simulations by means of a system of three nonlinear ordinary differential equations in the variables $x(t)$, $y(t)$, and $z(t)$, which represent the voltages across the capacitors C_1 and C_2 and the electric current in the inductor L (Kuznetsov et al., 2022) respectively. The Chua's circuit equations, which produce the Chua attractor in certain parameter regimes, are then given by:

$$\frac{dx}{dt} = \alpha\,(y - x - g(x)),$$
$$\frac{dy}{dt} = x - y + z,$$
$$\frac{dz}{dt} = -\beta y.$$

Where $g(x)$ is a piecewise linear function:

$$g(x) = m_1 x + 0.5(m_0 - m_1)\,(|x+1| - |x-1|)\,.$$

The system is discretized into a simple Euler approximation in the same manner as in Section 4.1.2 due to similar complications in the ability to directly approximate this chaotic system. We treat this discretized system as our considered form of the Chua circuit for the remainder of the paper. We make the following choice of parameters for the system which are known to create a chaotic regime: $\alpha = 15.6$, $\beta = 1/L = 28$, $m_2 = -1.143$, $m_1 = -0.714$. A diagram of a generated trajectory for this discretization is included in Fig. 6. With our dynamical systems well defined we now move on to experiment design.

5. Experiment design

For our experiments we consider two cases for artificially generating data. In the standard case, we generate a single trajectory of our dynamical system, with time step size $h = 0.01$ over the time interval $[0, 100]$, see Fig. 7. The orbit is further split so that the first 60% is used for training (interval $[0, 60]$), the next 20% for validation (interval $[60, 80]$), and the final 20% for test data (interval $[80, 100]$. Fig. 8 demonstrates the three data sets. Moreover, in the second case, we consider 400 distinct orbits, each taken with time step size $h = 0.01$ and time interval $[0, 5]$. 100 orbits are chosen to form the training set, 200 for the validation set, and 100 for the testing set. In this multiple orbit case batches may be formed using samples from multiple orbits simultaneously, thus allowing the network to perform training where it matches more than one orbit at a time. The batch size during the training is taken to be 500 for all the neural networks. We still require the sub-batch restriction as outlined in (4) in order to still apply the DTSN loss. This approach functions essentially as a compromise which has the sequential data needed for the LSTM-DTSN architecture while still considering the broader phase space rather than a single orbit. In contrast with the method of Lu et al. (2019), where a lattice of initial conditions was used to train a PINN, our approach is more easily applicable for real physical systems due to the greater ease in sampling several trajectories for a short time as opposed to a large number of initial conditions.

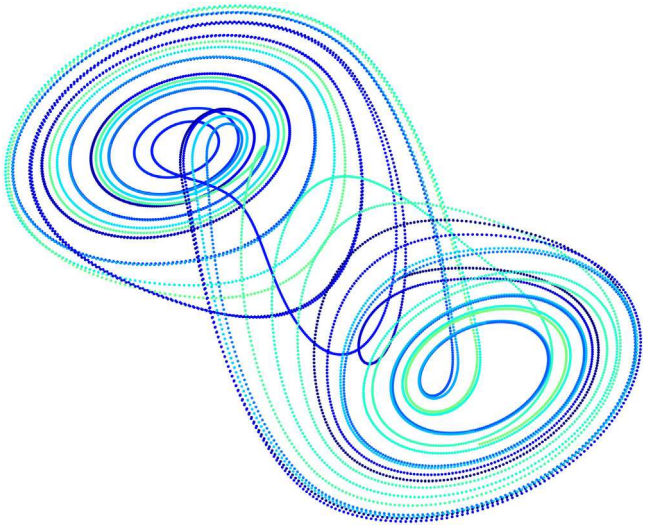


Fig. 6. Generated trajectory for the Chua circuit, note the two "scrolls" appearing in its attracting set. A single orbit over $t \in [0, 100]$ with initial condition $(0.7, 0, 0)$ and time step 0.001. The coloration changes each increment of 20 time units for ease of visibility.

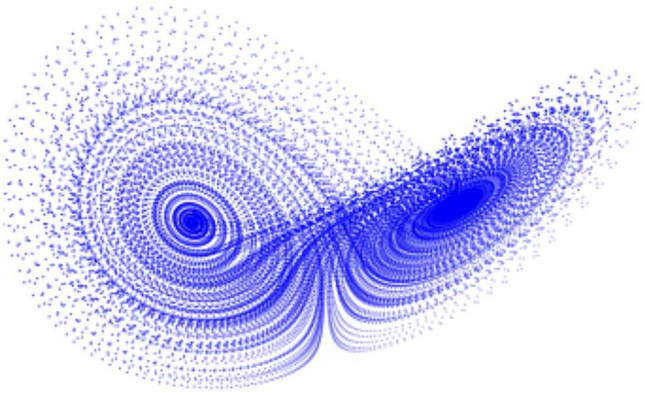


Fig. 7. A single trajectory for the Lorentz System, over $t \in [0, 100]$ with initial condition $(10, 10, 10)$ and time step 0.01.

Our first considered architecture for testing is a standard LSTM network (Hochreiter & Schmidhuber, 1997), which serves as a well understood and natural choice for the prediction of time-series data. The code of Madondo and Gibbons (2018) serves as the base for our particular implementation. Our choice of hyperparameters is given by Table 1. We consider the size m for samples $\mathbf{X}_i$ passed to the LSTM to be 4 (as defined in Eq. (2)) which gives us the dimension for our

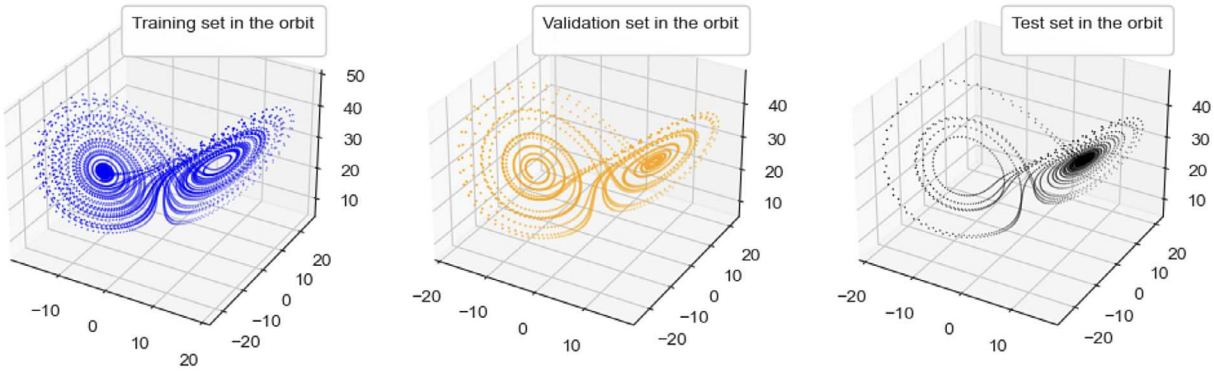


Fig. 8. The Training, Validation and Test sets created from the single orbit.

Table 1
Hyperparameters for LSTM-DTSN network.

Layer type	Output dimension	No. of parameters
Input Layer	(4,3)	0
LSTM1	100	44 000
LSTM2	100	80 400
Dense Output Layer	(1,3)	909

input layer as 4 time steps, each corresponding to a point in $\mathbb{R}^3$ hence our input dimension $(4, 3)$ corresponding to $\mathbb{R}^3$ tensored with itself 4 times. We then have 100 LSTM neurons, each made up of 4 LSTM cells each corresponding to one input time step, with the neuron producing a single output. Each neuron in the next layer "LSTM2" has 100 cells, corresponding to the 100 outputs of the previous layer, and each neuron producing a single output. Finally the "Dense Output Layer" is a simple fully connected layer that outputs a single time step prediction for the network's final output.

The implementation of the LSTM with the MSE loss function (3) is denoted as LSTM-MSE, the one applying the PINN loss function (13) as LSTM-PINN, and the one applying the DTSN loss (15) as LSTM-DTSN. In all cases the Adam optimizer is used to perform training (Kingma & Ba, 2014), and the learning rate is fixed at 0.0001 for the LSTM-MSE and LSTM-DTSN.

In early testing to find baselines we also considered the "Mean Absolute Error" (MAE) as an alternative to the MSE, the MAE simply being given by the same definition as the MSE but with no squaring on terms. We found that this loss performed significantly less well in training the network than the MSE in all cases, which is consistent with general findings on the usage of neural networks in time-series prediction (Jadon, Patil, & Jadon, 2022). We thus consider only the MSE as the standard baseline loss function. Similarly we also note that the LSTM is shown to consistently outperform simple feed forward neural networks in time-series prediction (Hochreiter & Schmidhuber, 1997), thus we do not consider the feed forward network in this context.

We found that the LSTM-PINN performed well with a change in the values of λ_1, λ_2 every 10 epochs during training, with a change in the learning rate after 30 epochs; the values of these parameter changes are given in Table 3. We found that starting with a higher value of λ_1 and then shifting the weight to λ_2 allowed the network to achieve rapid improvement in the early stages of training, while reaching high final accuracy at the end of training.

The Transformer (Vaswani et al., 2017) is also considered here. This neural network was originally developed for natural language processing tasks, which has had several adaptations to prediction in dynamical systems. For our implementation, we use 128 Time Distributed Dense linear layers and then split into 8 heads for multi-head attention. We then calculate the attention weights for scaled dot product attention, and finally after concatenating the heads we apply one final Time Distributed Dense linear layer. For the linear layers, we tried both Dense layers and Time Distributed Dense layers and found the latter to produce smoother, more accurate results. The encoder layer consists of multi-head attention and Dense layers followed by dropout. The output of the network is still a single step, to maintain parity with the LSTM method. The dimension for the query, key, and value vectors is $d_q = d_k = d_v = 64$. We use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1e-9$ and the custom learning rate scheduler given in Vaswani et al. (2017):

$$\text{learning rate} = d_{\text{model}}^{-0.5} * min(\text{step\_num}^{-0.5}, \text{step\_num} * \text{warmup\_steps}^{-1.5}). \tag{42}$$

Two main alterations were applied to this network method to improve its performance. First, we use a model-agnostic vector representation of time, Time2Vector (Kazemi et al., 2019), to better handle the time-series data of our dynamical system. We also replace the standard ReLU activation function with Sigmoid weighted Linear Units (SiLU) (Elfwing, Uchibe, & Doya, 2018). This was found to remove the need to rescale the data set during training, and improved the accuracy of the network's predictions.

We also considered the Transformer-based architecture from Geneva and Zabaras (2022), where a Koopman observable embedding was used to convert the physical space into a 1D vector representation. We however found substantial issues in reproducing the performance of this method, with our attempts to train the network failing to create adequate improvements in the network's performance. This may be due to a difference in the power of computer hardware available to us or an unknown error in the available code. We therefore used the more basic implementation of the Transformer outlined previously.

The first order DTSN loss function given by (15) was applied to both the LSTM and Transformer architectures. In both cases, the hyperparameter values are fixed at $\lambda_1 = 0.5, \lambda_2 = 0.5$, though we did not see noticeable changes in the results when varying these hyperparameters. For our experiments, we did not test higher order variation of the DTSN loss outlined in Eq. (20), as the first-order implementation already achieved noticeably improved results.

6. Results

Our results on the Lorenz system are summarized in Table 2. In all cases, the implementation of the DTSN loss function did not result in any noticeable increase in training time. We see that the DTSN achieves an order of magnitude improvement over the PINN approach, which already achieves about one order of magnitude improvement over the simple MSE loss function. This is despite the PINN using substantially more information about the system's behavior. We also note that even in the Transformer case which performed much less well we still had a significant improvement in the performance of the network, though

Table 2
Comparison of all considered network approaches for Lorenz system, are recorded in the form of mean ± standard deviation.

Case	Lorenz Test MSE
LSTM-MSE (one orbit)	$.00467 \pm 0.0005$
LSTM-PINN (one orbit)	$.00016 \pm 0.00004$
Transformer-MSE (one orbit)	0.52192557 ± 0.3
Transformer-DTSN (one orbit)	0.11781909 ± 0.1
LSTM-DTSN (one orbit)	$2 * 10^{-5} \pm 4 * 10^{-6}$
LSTM-DTSN (multi-orbit)	$\mathbf{4.3 * 10^{-6} \pm 6 * 10^{-7}}$

Table 3
Results during the training and testing of the LSTM-PINN network.

λ_1	λ_2	Learning rate	Train MSE	Test MSE
0.8	0.2	.0005	.00527	.00691
0.5	0.5	.0005	.00115	.00153
0.2	0.8	.0005	.00080	.00088
0.2	0.8	.0001	.00014	.00018
0.1	0.9	.0001	.00013	.00016

Table 4
Final error results from transformer prediction.

Loss function	Input steps	Test MSE
MSE	3	0.53802687
MSE	6	0.7338447
MSE	9	0.52192557
DTSN	3	0.26652685
DTSN	6	0.35463473
DTSN	9	0.11781909

Table 5
Comparison of LSTM neural network using different loss function for the Chua system, on dataset generated by a single orbit. The losses are recorded in the form of mean ± standard deviation.

Neural network	LSTM-MSE	LSTM-PINN	LSTM-DTSN
Chua test MSE	$.0076 \pm 0.0004$	$3.8 * 10^{-4} \pm 3 * 10^{-5}$	$\mathbf{6.7 * 10^{-5} \pm 4 * 10^{-6}}$

not the full multiple orders of magnitude difference seen going from the LSTM-MSE to the LSTM-DTSN.

Table 3 further describes the results of the LSTM-PINN. Each change in hyperparameter values was applied after 10 epochs of training, where at each of these cutoffs the network's training and testing MSE had reached a stable point. We see that notable improvements were able to be made in the early stages, but eventually a limit was found at the final set of hyperparameter values where further adjustments did not produce improved results.

Next we consider the multiple orbit case. Under this regime the DTSN as the test MSE drops all the way to the order of 10^{-6}. We note this task is theoretically more difficult due to the wider range of the phase space the network must learn to predict on multiple distinct orbits. This method is also especially important for highly chaotic cases, as the shorter orbit lengths mean a given discretization may be much more accurate to the true underlying system than in cases of longer orbits. We include a log plot of the testing/validating loss in Fig. 9. In over 10 conducted trials we found that the plots to reach the same values in the same number of epochs, thus this figure represents an average picture of the performance, rather than any exceptional result. We note how the network rapidly achieved low loss values in a small number of epochs. The short number of epochs may be explained by the usage of multiple distinct orbits, which allows the network to much more rapidly learn general information about the system. We find the results of the DTSN to be extremely promising, especially considering how much room there is for further improvement by including higher order variants.

When initially selecting a greater number of training orbits compared to validation orbits, we observed that the validation loss was lower than the training loss. To address this, we adjusted the distribution to 100 orbits for training and 200 orbits for validation. This change resulted in a validation loss that is consistently higher than the training loss, as illustrated in Fig. 9. Notably, this pattern persisted even when using the standard MSE loss metric, confirming that the observed behavior is not an artifact of the DTSN loss function. We know that RNN architectures, and the LSTM in particular, have some potential limitations in how accurately they are able to capture chaotic dynamics (Mikhaeil, Monfared, & Durstewitz, 2022) and some unusual behaviors may arise as artifacts of how the architecture interacts with this type of complex dynamics. Overall we believe that this unusual result is primarily a sign of the incredible inherent challenges of predictions on chaotic systems and we hope to find new approaches to further refine our methods in future work.

We also comment on the performance of the Transformer baseline. Several values for the number of input steps m were considered, as the architecture is known to work well on larger collections of input data. The results for several choices of input steps are compiled together in Table 4. One can see that even compared to the simple LSTM-MSE network, the performance of the Transformer is substantially worse. It may be partially explained by the dynamical system being "memoryless", where the true value of the next point in the sequence is determined entirely by the previous point. In this case, learning long term dependencies may not be very useful. Given the performance of Geneva and Zabaras (2022), which we could not reproduce ourselves, it is also possible that these poor results are due to the data not being preprocessed in such a way that the network can more easily train on it. The Transformer initially considered systems where all data points come from a finite library of possible tokens. For a system with continuous phase space like the discrete time Lorenz system, the choice of how to discretize the space may be extremely important to how well the Transformer can learn the system. In any case, this suggests that the Transformer cannot be simply applied to predicting chaotic dynamical systems.

Each of our considered architecture and loss function combinations were also applied to data generated by the Chua circuit. Performance of the transformer was found to still be quite poor as well as computationally costly, and is thus omitted. We also applied the same variation of hyperparameters as seen in Table 3, The final results on the Chua circuit are presented in Table 5.

The obtained results on the Chua circuit are very similar in orders of magnitude to the results on the Lorenz system in Table 2. Thus we find across these two different chaotic systems the LSTM-DTSN with multiple orbits is capable of a three orders of magnitude reduction in test MSE compared to a standard LSTM-MSE network.

7. Discussion

In this paper we proposed the Discrete-Temporal Sobolev Network loss function to improve neural network predictions for time series, particularly for chaotic dynamical systems. We demonstrated its performance on the discretized Lorenz System and the Chua circuit. This showed exceptionally good improvements in performance, with a minimum of one order of magnitude smaller test MSE than the PINN approach, two orders of magnitude less than the MSE loss function approach, and when multiple orbits were used the test MSE was reduced by another order of magnitude. The DTSN is also highly flexible, able to be easily adapted to a wide variety of network architectures, though we restrict to two of the most important architectures for time-series here to simplify initial analysis. It is especially promising that the DTSN does not require any substantial increase in computational load for the network's training.

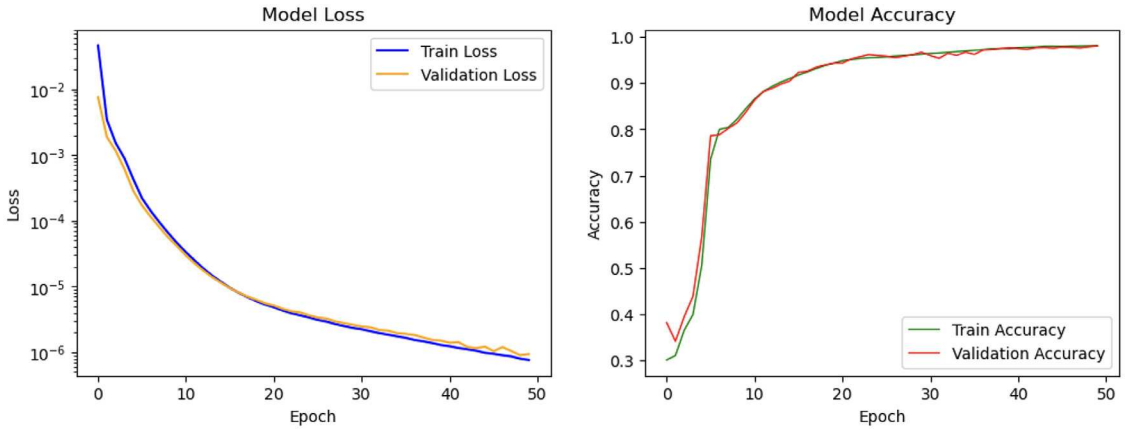


Fig. 9. The log plot of the training and validation loss, see (a), as well as the log plot of the training and validation accuracy, see (b), for the LSTM-DTSN neural network on multiple orbits of the Lorenz system over 50 epochs.

The most immediate direction for future work would be to empirically evaluate the DTSN with more classes of dynamical systems. We could consider some inherently discrete dynamical systems such as the Henon map to determine if the method still applies well in a system which is not a small time discrete approximation. We may also consider systems which contain many singularities, such as chaotic billiard flows which have dense singular sets (Nikolai Chernov, 2006), to evaluate the network's robustness under lower regularity assumptions. The existence of singularities though, which we note nearly all commonly considered discrete systems possess, does create substantial complications in training the network. On such a system stochastic gradient descent may cause approximations to cross singularity lines and produce dramatically inaccurate results. Such systems would still be worthy of further research, but would require a more substantial reevaluation of the network's fundamental training. Simply testing on less difficult chaotic systems may also be useful just for simply building the body of empirical evidence.

One significant idea for future improvement would be in modifications of our learning algorithm. As shown in the work of Mikhaeil et al. (2022) there are inherent challenges to the usage of certain RNN architectures to forecast chaotic systems. The restrictions imposed in the LSTM to prevent exploding gradients may also limit the expressibility of the network. The work of Mikhaeil et al. (2022) however proposes some means to address these issues in the training of a RNN through what they call "sparsely forced Back-Propagation Through Time". Applying this method in tandem with the DTSN would provide a useful test of how robust our loss function is under an alternative learning algorithm.

Testing on PDE systems may also be significant as that is the context where PINNs have seen some of their greatest success (Cuomo et al., 2022). The DTSN is capable of outperforming a PINN method in the discretization of ODE systems we present, so it would be interesting to see how it fairs on PDEs or PDE derived systems. We would also like to consider some practical work on concrete systems to show some applications of this work. We have some preliminary results on application of the DTSN to stock price prediction that look promising. We found the complexity of the stock price modeling substantial enough to be beyond the scope of the current paper though and leave it for a future work.

In addition to broader testing of the DTSN, we also consider future directions on improving the loss function itself. While we used the first order variant of the DTSN for the experiments in this paper, future work could make use of the general form of the loss function given in (20). Broadly speaking, the usage of higher order terms tends to improve approximation methods up to some point of diminishing returns and the DTSN loss function should be no exception. A key challenge in considering higher orders though will be in finding the appropriate cutoff. A general method for choosing λ_j terms could also substantially improve results on more complex systems where high values of the higher order derivatives (or discrete derivatives) could play a more significant role.

It may also be useful to expand our loss function to approximate the more complex forms of our norms given in Eqs. (32) and (33). In the present work we assume no information on spatial derivatives is accessible, but such information could be easily incorporated into a new loss function approximating (32). Even without any additional information assumptions it may also be useful to consider the approach of varying the number of steps that are being approximated, equivalent to varying N in either of these two norms. The direct approximation of larger step sizes may prove useful in estimating orbits on larger timescales, though it may introduce further complications as well. Overall the DTSN loss function is an approach to network training with many directions for further improvement, as well as many possibilities for further application.

CRediT authorship contribution statement

Connor Kennedy: Conceptualization, Formal analysis, Methodology, Supervision, Writing – original draft, Writing – review & editing. **Trace Crowdis:** Data curation, Investigation, Software, Validation. **Haoran Hu:** Data curation, Investigation, Software, Validation, Visualization, Writing – original draft. **Sankaran Vaidyanathan:** Data curation, Resources, Software, Validation. **Hong-Kun Zhang:** Conceptualization, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors did not use any generative AI and AI-assisted technologies in the writing process. The authors take full responsibility for the content of the publication.

Acknowledgments

H.-K. Zhang is partially supported by the National Science Foundation, United States (DMS-2220211), as well as Simons Foundation Collaboration Grants for Mathematicians, United States (706383).

Appendix A. Shadowing lemmas

For any dynamical system with a positive Lyapunov exponent, difficulties arise in the simulation of trajectories. The effect is an exponential growth in distance between trajectories even if they start arbitrarily close to each other. This causes small estimation errors, or even simple floating point arithmetic rounding, to eventually lead to substantial deviation in predicted trajectories (Katok, 1995). In the case of the Lorenz system there are some additional stability factors, such that the growth of error may not be quite as large as the large positive Lyapunov exponent might imply, but this still results in the requirement of extremely high order for estimation methods, as well as a high number of digits for variable storage, to generate longer trajectories (Estep & Johnson, 2011).

The shadowing lemma serves as one method to address this issue, albeit at a significant cost. Its original form is stated as follows (Katok, 1995) (a corresponding version for a flow is also present in the source):

Lemma 1. *Given a map $F : M \to M$ of a metric space (M, d) to itself, define a δ-pseudo-orbit as a sequence $(\mathbf{x}_n)$ of points such that $\mathbf{x}_{n+1}$ belongs to a δ-neighborhood of $F(\mathbf{x}_n)$.*

Then, near a hyperbolic invariant set, the following statement holds: Let Λ be a hyperbolic invariant set of a diffeomorphism F. There exists a neighborhood U of Λ with the following property: for any $\epsilon > 0$ there exists $\delta > 0$ such that for any (finite or infinite) δ-pseudo-orbit that stays in U, the orbit also stays in an ϵ-neighborhood of some true orbit. (we refer to staying within this range of a true orbit to be ϵ-shadowing a true orbit). In more compact notation:

$$\forall(\mathbf{x}_n), \mathbf{x}_n \in U, d(\mathbf{x}_{n+1}, F(\mathbf{x}_n)) < \delta \quad \exists(\mathbf{y}_n), \mathbf{y}_{n+1} = F(\mathbf{y}_n),$$
$$\text{such that} \quad \forall n, \mathbf{x}_n \in U_\epsilon(\mathbf{y}_n). \tag{A.1}$$

This means that for a system with a hyperbolic invariant set (usually the entire phase space with some equilibria removed), if we have an estimation method generating the sequence $\mathbf{x}_n$, with some small error δ, then the predicted orbit is always within ϵ distance of some true orbit, though this may be the orbit of something other than our original starting point. So we guarantee that the estimation is always close to some real orbit, but we lose control over which specific orbit we are observing. This works well enough to characterize certain features of a system, but does pose potential issues with attempting to train a network to learn how to accurately learn predictions for a given system.

For the Lorenz system however the situation is even worse, as it does not precisely satisfy the assumptions of the original Shadowing Lemma. The Lorenz system has singularities present in its hyperbolic invariant set, and thus does not have the uniform hyperbolicity necessary to satisfy the Shadowing Lemma (Araujo & Pacifico, 2010). There is an alternate version of the shadowing lemma which can still hold for such systems with singular hyperbolicity, but the shadowing result then only holds for finite trajectories, with further calculations needed to verify whether shadowing orbits still exist and no general guarantee for their length (Coomes et al., 1994). It is partially due to these complications that we introduce the discretized Lorenz system and limit the length of trajectories considered, so that the only error present is from the floating point arithmetic and we do not exceed the boundaries of computability.

Appendix B. Properties of the Lorenz system

In this section we briefly review the general properties of the Lorenz system and Lyapunov exponents. To provide physical context for the system we recall that the variable x is proportional to the intensity of the convection movement, y is proportional to the temperature difference between updrafts and downdrafts, and z is proportional to the deviation of the vertical temperature profile from a linear profile. Meanwhile for the parameters σ corresponds to the Prandtl number, a dimensionless quantity calculated from the ratio between the momentum diffusivity of an entity and its thermal diffusivity. ρ corresponds to the Rayleigh number, which characterizes the heat transfer within a fluid. Finally, β is a value depending on the layer of the atmosphere the system is assumed to be in Lorenz (1963).

For the classical parameter values of the system, $(\sigma, \rho, \beta) = (10, 28, 8/3)$ the system has an attracting set with several interesting properties. The attractor has fractional dimension estimated to be 2.06 ± 0.01 (Benzi, Paladin, Parisi, & Vulpiani, 1984). Attractors with such dimension are commonly referred to as *strange attractors* though the more significant properties of such attractors lie outside the scope of this paper. The attractor's shape roughly corresponds to two annular regions around 2 hyperbolic fixed points for the system (though the true shape is much more complex due to the dimensionality). Trajectories on the attractor are aperiodic and the attractor is invariant (Sparrow, 1982). We crucially note that in Section 4.1.2 that the discretized system considered has a topologically equivalent attractor, thus while the exact shape may differ all of these properties are preserved.

The other key property of the system's dynamics is the general sensitive dependence on initial conditions for the system. This is expressed quantitatively by the presence of a positive maximal Lyapunov exponent (Sparrow, 1982). (While we do not explicitly discuss the Chua circuit in this section the same concepts of Lyapunov exponents and estimations still applies to the Chua circuit (Parlitz, 1993).) We recall the general definition of a maximal Lyapunov exponent for a dynamical flow (Katok, 1995). Given a flow Φ_t we fix a point $\mathbf{x}_0$ and let $\mathbf{x}_1$ be an unfixed point. We define $\|\delta(t)\| := \|\Phi_t(\mathbf{x}_0) - \Phi_t(\mathbf{x}_1)\|$. Then the definition of the maximal Lyapunov exponent is given by:

$$\lambda_e(\mathbf{x}_0) = \lim_{t\to\infty} \lim_{\delta(0)\to 0} \frac{1}{t} \ln \frac{|\delta(t)|}{|\delta(0)|} \tag{B.1}$$

The quantity is also approximated by this equation for small values of $\delta(0)$ and t:

$$\|\delta(t)\| \approx \|\delta(0)\| e^{\lambda_e t}. \tag{B.2}$$

It is this approximation that explains the sensitive dependence property. Infinitesimally close trajectories diverge from each other exponentially quickly with a rate given by the maximal Lyapunov exponent. The exact details are more complex but this serves as a useful intuition for their behavior. Now for a discrete dynamical system, given by a differentiable map F, the maximal Lyapunov exponent at point $\mathbf{x}_0$ is defined by

$$\lambda_e(\mathbf{x}_0) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln |F'(F^i(x_0))|. \tag{B.3}$$

It is this definition that must be used for our discretized systems defined in Section 4 as there is no equivalent notion of small values for t in this case. We note that both of these definitions are only given for a single point at a time, however for the original Lorenz system and the discrete system the corresponding Jacobian matrices are constant, thus the value of all Lyapunov exponents are constant (Katok, 1995).

We also consider the general definition of a Lyapunov exponent (Viswanath, 1998). Assume we have an n-dimensional dynamical system (discrete or continuous) with Jacobian J and initial condition $\mathbf{x}_0$ for which the orbit is not periodic. The full set of Lyapunov exponents $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_n$ for a n-dimensional system are defined by

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \log \sigma_i(J_F(t)). \tag{B.4}$$

where σ_i is the ith singular value of the Jacobian. This essentially describes the expanding, or contracting, behavior of the Jacobian across a collection of vectors spanning all directions. For the discrete map given by (37) the estimated Lyapunov exponents are $1.041, -0.001, -14.992$, found by averaging on 100 orbits, each of which has length 100000. The maximal Lyapunov exponent being positive is what lets us know the system is chaotic, while the summation of the Lyapunov exponents being negative tells us the system is dissipative. This means that any volume of initial trajectories tends to zero as time passes, the system shrinks volumes over time, even though it stretches close trajectories apart from each other.

References

Abdeljawad, Ahmed, & Grohs, Philipp (2022). Approximations with deep neural networks in Sobolev time-space. *Analysis and Applications*, *20*, 499–541.

Araujo, Vitor, & Pacifico, Maria (2010). *Three-dimensional flows*: *vol. 53*, Springer.

Arena, P., Baglio, S., Fortuna, L., & Manganaro, G. (1995). Chua's circuit can be generated by CNN cells. *IEEE Transactions on Circuits and Systems I*, *42*(2), 123–125.

Benzi, R., Paladin, G., Parisi, G., & Vulpiani, A. (1984). On the multifractal nature of fully developed turbulence and chaotic systems. *Journal of Physics A: Mathematical and General*, *17*(18), 3521–3531.

Cannas, Barbara, Cincotti, Silvano, Marchesi, Michele, & Pilo, Fabrizio (2001). Learning of chua's circuit attractors by locally recurrent neural networks. *Chaos, Solitons, and Fractals*, *12*(11), 2109–2115.

Chattopadhyay, Ashesh, Hassanzadeh, Pedram, & Subramanian, Devika (2020). Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, *27*(3), 373–389.

Cholakov, Radostin, & Kolev, Todor (2021). Transformers predicting the future. Applying attention in next-frame and time series forecasting, CoRR.

Coomes, Brian A., Koçak, Hüseyin, & Palmer, Kenneth J. (1994). Shadowing orbits of ordinary differential equations. *Journal of Computational and Applied Mathematics*, *52*(1), 35–43.

Cuomo, Salvatore, Cola, Vincenzo Schiano Di, Giampaolo, Fabio, Rozza, Gianluigi, Raissi, Maizar, & Piccialli, Francesco (2022). Scientific machine learning through physics-informed neural networks: Where we are and what's next, CoRR. abs/2201.05624.

Czarnecki, Wojciech M., Osindero, Simon, Jaderberg, Max, Swirszcz, Grzegorz, & Pascanu, Razvan (2017). Sobolev training for neural networks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*: *vol. 30*, Curran Associates, Inc.

Dubois, Pierre, Gomez, Thomas, Planckaert, Laurent, & Perret, Laurent (2020). Data-driven predictions of the Lorenz system. *Physica D: Nonlinear Phenomena*, *408*, Article 132495.

Elfwing, Stefan, Uchibe, Eiji, & Doya, Kenji (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, *107*, 3–11, Special issue on deep reinforcement learning.

Estep, Donald, & Johnson, Claes (2011). The computability of the lorenz system. *Mathematical Models & Methods in Applied Sciences*, *08*.

Evans, Lawrence C. (2022). *Partial differential equations*: *vol. 19*, American Mathematical Society.

Galias, Zbigniew (1997). Positive topological entropy of Chua's circuit: A computer assisted proof. *International Journal of Bifurcation and Chaos*, *7*(02), 331–349.

Geneva, Nicholas, & Zabaras, Nicholas (2022). Transformers for modeling physical systems. *Neural Networks*, *146*, 272–289.

Hayes, Wayne B., & Jackson, Kenneth R. (2003). Rigorous shadowing of numerical solutions of ordinary differential equations by containment. *SIAM Journal on Numerical Analysis*, *41*(5), 1948–1973.

Hochreiter, Sepp, & Schmidhuber, Jürgen (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780.

Hornik, Kurt, Stinchcombe, Maxwell, & White, Halbert (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Jadon, Aryan, Patil, Avinash, & Jadon, Shruti (2022). A comprehensive survey of regression based loss functions for time series forecasting. arXiv preprint arXiv:2211.02989.

Katok, Anatole (1995). *Introduction to the modern theory of dynamical systems*. Cambridge ; New York, NY: Cambridge University Press.

Kazemi, Seyed Mehran, Goel, Rishab, Eghbali, Sepehr, Ramanan, Janahan, Sahota, Jaspreet, Thakur, Sanjay, et al. (2019). Time2vec: Learning a vector representation of time. arXiv preprint arXiv:1907.05321.

Kennedy, M. P. (1993). Three steps to chaos. I. Evolution. *IEEE Transactions on Circuits and Systems I*, *40*(10), 640–656.

Kingma, Diederik, & Ba, Jimmy (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kuznetsov, Nikolay V., Mokaev, Timur N., Ponomarenko, Vladimir I., Seleznev, E P, Stankevich, Nataliya V., & Chua, Leon Ong (2022). Hidden attractors in Chua circuit: mathematical theory meets physical experiments. *Nonlinear Dynamics*, *111*, 5859–5887.

Letellier, Christophe, & Mendes, Eduardo (2005). Robust discretizations versus increase of the time step for the Lorenz system. *Chaos (Woodbury, N.Y.)*, *15*, 13110.

Lorenz, Edward N. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, *20*(2), 130–141.

Lorenz, E. N. (1995). Predictability: a problem partly solved. In *Seminar on predictability, 4-8 September 1995*: *vol. 1*, (pp. 1–18). Shinfield Park, Reading: ECMWF.

Lü, Jinhu, & Chen, Guanrong (2006). Generating multiscroll chaotic attractors: theories, methods and applications. *International Journal of Bifurcation and Chaos*, *16*(04), 775–858.

Lu, Lu, Meng, Xuhui, Mao, Zhiping, & Karniadakis, George E. (2019). DeepXDE: A deep learning library for solving differential equations, CoRR. abs/1907.04502.

Madan, Rabinder N. (1993). *Chua's circuit: a paradigm for chaos*. WORLD SCIENTIFIC.

Madondo, Malvern, & Gibbons, Thomas E. (2018). Learning and modeling chaos using LSTM recurrent neural networks. In *MICS 2018 proceedings*.

Mikhaeil, Jonas M., Monfared, Zahra, & Durstewitz, Daniel (2022). On the difficulty of learning chaotic dynamics with RNNs.

Nikolai Chernov, Roberto Markarian (2006). *Chaotic billiards*. American Mathematical Society.

Parlitz, Ulrich (1993). Lyapunov exponents from Chua's circuit. *Journal of Circuits, Systems, and Computers*, *3*(02), 507–523.

Petzold, Linda (1983). Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM Journal on Scientific and Statistical Computing*, *4*(1), 136–148.

Raissi, Maziar, Perdikaris, Paris, & Karniadakis, George E. (2017). Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, CoRR. abs/1711.10561.

Shalova, Anna, & Oseledets, Ivan (2020). Tensorized transformer for dynamical systems modeling.

Sparrow, Colin (1982). *The lorenz equations: bifurcations, chaos, and strange attractors*. New York, NY: Springer.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Virtanen, Pauli, Gommers, Ralf, Oliphant, Travis E., Haberland, Matt, Reddy, Tyler, Cournapeau, David, et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272.

Viswanath, Divakar (1998). *Lyapunov exponents from random fibonacci sequences to the Lorenz equations* (Ph.D. thesis), Cornell University.

Vlachas, Pantelis, Byeon, Wonmin, Wan, Zhong, Sapsis, Themistoklis, & Koumoutsakos, Petros (2018). Data-driven forecasting of high-dimensional chaotic systems with long-short term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, *474*.

Wang, Sifan, Sankaran, Shyam, & Perdikaris, Paris (2022). Respecting causality is all you need for training physics-informed neural networks.

Yucesan, Yigit A., & Viana, Felipe A. C. (2021). Hybrid physics-informed neural networks for main bearing fatigue prognosis with visual grease inspection. *Computers in Industry*, *125*, Article 103386.

Zhang, Ruiyang, Liu, Yang, & Sun, Hao (2020). Physics-informed multi-LSTM networks for metamodeling of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering*, *369*, Article 113226.

Zinkevich, Martin, Weimer, Markus, Li, Lihong, & Smola, Alex (2010). Parallelized stochastic gradient descent. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems*: *vol. 23*, Curran Associates, Inc..