Neural Networks as Geometric Chaotic Maps

Ziwei Li, and Sai Ravela

***Abstract*—The interest in using neural networks as models of nonlinear dynamics are rapidly expanding. Despite many prior successes, the ability and mechanism of neural networks to learn chaotic dynamics remain poorly understood. In this work, we show that parsimonious neural networks trained only on few data points suffice for the reconstruction of strange attractors, extrapolation outside the boundaries of training data, and accurate prediction of the local divergence rates on the attractor. To understand the reason behind this good performance, we decompose mappings in the neural network into a series of geometric stretching and compression operations that indicate topological mixing and, therefore, chaos. This indicates that after training, the neural network has learned to become structurally similar to the chaotic dynamical system on which it is trained. To quantify the necessary complexity of the neural network, we design simple neural networks by bounding the loss gradient spectra using polynomial regression.**

***Index Terms*—Neural networks, chaos, topological mixing, nonlinear dynamical systems.**

I. Introduction

CHAOTIC systems are ubiquitous [1], and their dynamics are usually governed by nonlinear equations or maps. However, finding the exact solutions is often impossible, largely due to the nonlinearities and the characteristic wherein two close-by trajectories exponentially diverge. Hence, approximate numerical solutions are usually sought for when studying these dynamical systems. To quantify errors, uncertainty, and predictability of the numerical solutions, modelers simulate an ensemble of initial conditions using discretized numerical models of the nonlinear governing equations. Doing so, however, is challenging because the equations are nonlinear, models are often high-dimensional, and the uncertainties are non-Gaussian [2]. As a result, the search for simple-yet-effective models for chaotic dynamics remains a crucial pursuit in engineering and physical sciences.

The application of artificial neural networks to model chaotic dynamics started at least 30 years ago [3]. Recently, there has been a surge of interest in using NNs to emulate chaotic systems [4]–[19], showing neural networks as promising dynamical models for such systems. We restrict term "neural network" in our paper to be the traditional feedforward neural network (hereby NN). We also focus our attention on the classic Lorenz-63 (L63) system [20], a commonly used chaotic system for both chaos and learning. Applying NN to emulate the dynamics of the L63, our first observation is that a single-hidden-layer NN equipped with only 4 neurons and trained using 40 data points is able to reconstruct the strange attractor. Another numerical experiment shows that NN can extrapolate from partial knowledge of the attractor. Further comparison between the L63 and NN systems using finite-time Lyapunov exponents also indicates that the neural model becomes as chaotic as the L63 system in terms of predictability. Although this success is consistent with previous efforts using NN to emulate L63 (e.g., [13], [15]), it's not clear how chaos emerges in the neural system. The typical explanation is to resort to the universal approximation theorem (UAP) [21]–[24]. However, UAP is unsatisfactory here because it neither explains the emergence of chaos in the network nor the efficacy with which the attractor is reconstructed.

Here, we show that a geometric interpretation of the L63 system [25] explains the emergence of chaos in the neural system. Similar to the geometric map of L63, the neural map alternately rotate, stretch, and compress, which are the defining characteristics of chaotic dynamics [26]. In fact, these transformations are not particular to L63, but are very common in chaotic systems, and we see them with more clarity on the simpler discrete Hénon map and the corresponding NN map. Possessing the geometric properties required by chaos theory enables NNs to efficiently reconstruct strange attractors and match the true systems' predictability.

To the best of our knowledge, this explanation of the neural learnability of chaos is new[1]. Due to the chaotic nature of the system from which it learns, the predictive skill of NN will always remain limited. Nevertheless, the NN matches the predictability of the underlying system. Our results formally reveal why NNs can emulate chaos, thereby justifying their usage as reduced models of chaotic dynamics. For the practical aspect of learning from a system that exhibits chaos but whose exact mathematical formulas are unknown or difficult to construct (i.e., reduced models of the atmosphere), our results help to establish the use of NN as a viable solution to learn from empirical data.

We further argue that the optimal learning system must efficaciously emulate chaos to avoid overfitting. Realizing that L63 is a polynomial system, we match the equilibrium norm of the learning dynamics of NN training and that of polynomial regression to provide a lower bound of the necessary number of neurons for emulation of the L63 system [28], [29].

The remainder of this paper is organized as follows. Section II describes related work. The experimental results on the predictability of NN is shown in section III, followed by a geometrical perspective in section IV. We then discuss the bounds on the complexity of neural network in section V and conclude in VI.

Ziwei Li and Sai Ravela are with the Department of Earth, Atmospheric, and Planetary Sciences, Massachusetts Institute of Technology, Cambridge, MA, 02139 (e-mail: ziweili@mit.edu, ravela@mit.edu).

[1]The term "learnability" is used here in the sense of neural system's fidelity to specified properties of a dynamical system, e.g., the predictability of the dynamical system quantified by its finite-time Lyapunov exponent. This is different from, e.g., Valiant's definition [27].

II. Related work

Prior works using neural networks to model chaotic dynamics can be categorized in two ways: designing primitive neural networks that achieve chaotic behavior (bottom-up), and learning from data generated by chaotic systems (top-down).

It is not surprising that low-dimensional neural networks can exhibit chaotic behavior, as multiple previous results have already shown. Models of 3 or 5 neurons with a truncated polynomial activation function were found to show chaotic behaviors [4], 3-D [6], [7] and 2-D [12] neural maps shows period-doubling bifurcations in particular parameter ranges to give onset of chaos, a 3-D cellular network can obtain horseshoe map for some weight matrices [9], and a delayed 2-D network has chaotic dynamics for certain parameter regimes [11].

Neural networks of different structures and complexities were used to learn from data and reproduce the dynamics of chaotic systems. Ref. [5] used multilayer perceptron models in addition to linear models to propagate embedded data, and ref. [10] established the equivalence between a class of neural networks and Devaney's definition of chaos. The dynamics of the L63 system can be emulated with simple feedforward NNs [13], [14], with tensor recurrent networks [15] and with LSTM [17]. NNs similar to our setup were also thought of as nonlinear ODE propagators [18], and were used in quantifying Lyaponuv exponents in higher dimensional chaotic systems [16], [19].

Our work bridges a gap between bottom-up simple theoretical NN models and top-down complex operational models by fitting a parsimonious feedforward NN onto the L63 system, revealing its efficacy in modeling chaotic dynamics. A geometric perspective further reveals the structural similarity between NN maps and chaotic maps, which makes NN a suitable and potentially explainable model for data-driven problems.

III. Neural Lorenz-63 emulation

The L63 model was originally used to describe 2-D Rayleigh-Bénard convection. The spectral components of the dynamical fields are truncated to a set of ordinary differential equations [20]:

$$\begin{aligned} \dot{X} &= \sigma(Y - X), \\ \dot{Y} &= \rho X - Y - XZ, \\ \dot{Z} &= -\beta Z + XY, \end{aligned} \tag{1}$$

where X and Y are the strengths of the streamfunction and temperature modes, and Z represents the deviation of the vertical temperature profile from linearity. Consistent with [20], we set $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$. The solutions of L63 are known to be dissipative (volume in phase space contracts rapidly), and chaotic (sensitive to initial perturbations).

We define L63 as a discrete map from the current state of the system $\mathbf{x}_n = (X, Y, Z)^{\mathrm{T}}$ to the state of the next timestep $\mathbf{x}_{n+1}$:

$$\mathbf{\Phi}_{\mathrm{L63}}(\mathbf{x}_n) \mapsto \mathbf{x}_{n+1}. \tag{2}$$

We choose the discrete form both for L63 and NN maps because it provides a straightforward geometric connection between L63 and the dynamics in the neural net, as we shall discuss in section IV. Since the exact form of (2) for L63 is unknown, the discrete map is obtained by numerically integrating (1) and sampling at increment $\mathrm{d}t = 0.01$.

A. Compact neural model

We use single-hidden-layer feedforward neural networks to learn the dynamics of L63. The functional form of the NN is

$$\mathbf{\Phi}_{\mathrm{NN}}(\mathbf{x}_n) = \mathbf{W_2}\, g(\mathbf{W_1}\mathbf{x}_n + \mathbf{b_1}) + \mathbf{b_2}, \tag{3}$$

in which the 3×1 input vector ($\mathbf{x}_n$) is left-multiplied by an $L \times 3$ weight matrix ($\mathbf{W_1}$), and is added to an $L \times 1$ bias term ($\mathbf{b_1}$), where L is the number of neurons. The resulting vector is then element-wise "compressed" by a sigmoid function $g(\cdot)$, which takes the form of $\tanh$ in our setup. Left-multiplication by a $3 \times L$ matrix ($\mathbf{W_2}$) followed by addition of another bias term ($\mathbf{b_2}$) finishes a mapping iteration.

We use Matlab function *ode45* to numerically solve for the discrete maps of L63 as training data. To obtain data on the attractor, we randomly initialize 1000 trajectories from region $[-20, 20] \times [-20, 20] \times [0, 50]$ with uniform distribution. Each trajectory is integrated for 2500 timesteps. We abandon the first 2000 timesteps to remove the transient parts, which are typically much shorter than 2000 steps. The remaining 500 timesteps of the 1000 trajectories are aggregated as pairs $(\mathbf{x}, \mathbf{x}')$ that satisfy $\mathbf{x}' = \mathbf{\Phi}_{L63}(\mathbf{x})$ to form the training data pool.

The locations of the training data ($\mathbf{x}$) can be treated as a representation of the L63 attractor ($\mathcal{A}_{\mathrm{L63}}$), and each consecutive location pair provides information about the L63 flow. We then randomly sample a specific number of location pairs from the data pool to train NNs. Each NN is trained for 10^3 epochs with Bayesian regularization [30].

B. Comparisons of predictability

To compare the local divergence rates of NN and L63 and quantify their similarities in predictability, we use the maximum finite-time Lyapunov exponent (FTLE) [31]. The FTLE is computed from forward-propagating two nearby trajectories that originate from the vicinity of the $\mathcal{A}_{\mathrm{L63}}$ attractor. Formally, it is defined as

$$\lambda_{\max} := \frac{1}{N_t} \ln \frac{\left| \max_{\delta \mathbf{x}_0} \delta \mathbf{x}_{N_t} \right|}{|\delta \mathbf{x}_0|} = \frac{1}{N_t} \ln \sqrt{\sigma_{\max}}, \tag{4}$$

where $\lambda_{\max}$ denotes the maximum FTLE, $\delta \mathbf{x}_0$ is the initial perturbation between two trajectories, and $\delta \mathbf{x}_{N_t}$ denotes their difference after N_t steps. The FTLE relates to the original Lyapunov exponent when $N_t \to \infty$ and $\delta \mathbf{x}_0 \to 0$ [32]. We calculate $\lambda_{\max}$ using the largest eigenvalue ($\sigma_{\max}$) of $\mathbf{J}_{N_t}^{\mathrm{T}} \mathbf{J}_{N_t}$, where $\mathbf{J}_{N_t}$ is the Jacobian matrix evaluated using perturbations around $\mathbf{x}_0$ (see S1 for details). The perturbation is 10^{-9} for all 3 directions.

Our numerical experiment shows that NN can learn the chaotic dynamics of L63 efficiently with a small number of data and neurons. The quadratic prediction error is reported in [14], and will not be the main focus of this paper. We instead compare the short-term and long-term behaviors of the two

systems. Specifically, we show that the dynamics represented by NN has similar predictability compared to L63 as quantified by FTLE, and NNs are able to extrapolate into regions that are unknown in the training data.

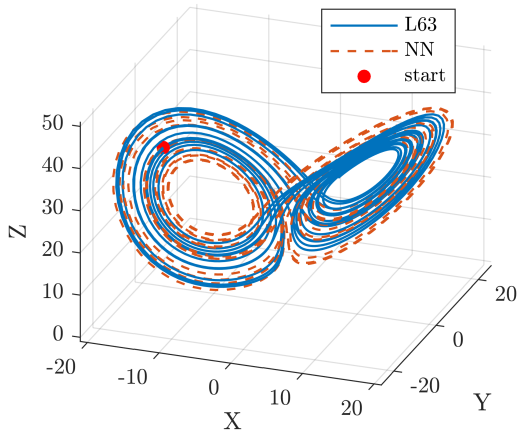

Fig. 1. Two trajectories produced by L63 (blue) and the 4-neuron NN trained on 40 data points sampled from the whole attractor (red-dashed). They start from the same location on the L63 attractor (red dot), and are both 2000 timesteps long.

We analyze a 4-neuron network trained on 40 randomly sampled data points. Its trained parameters are shown in table I. Fig. 1 depicts two trajectories, one of which follows the L63 flow and the other follows the flow of the trained network. They interlace with each other, tracing out the well-known Lorenz attractor. NN trajectories starting from other locations on the attractor follow the same behavior, and we have not seen any trajectory that diverge from the strange attractor. The close resemblance between the two structures indicates that the dynamics of this 4-neuron NN trained on 40 data points is similar to that of L63, confirming that NNs are able to learn chaotic dynamics efficiently.

TABLE I
PARAMETERS OF THE 4-NEURON NN TRAINED WITH 40 DATA POINTS OF L63. THE MATRICES ($\mathbf{W_1}$, $\mathbf{W_2}$, $\mathbf{b_1}$, $\mathbf{b_2}$) ARE AS IN (3), WHILE $\mathbf{S}$ IS AS IN (7)

Matrix	Values
$\mathbf{W_1}$	0.0091 0.0008 −0.0004 0.0140 0.0063 −0.0016 0.0061 0.0023 −0.0049 0.0085 0.0036 0.0041
$\mathbf{b_1}^{\mathrm{T}}$	0.1697 −0.6054 −0.0449 0.1773
$\mathbf{W_2}$	94.6004 8.7248 −8.0364 3.0535 −349.8684 11.3885 207.0634 227.4161 32.1244 93.9784 −214.6608 11.9787
$\mathbf{b_2}^{\mathrm{T}}$	−12.1241 34.2950 33.6097
diag($\mathbf{S}$)	2.7988 1.2134 0.6438 0.0000

To calculate FTLE, we generate points following the NN flow using the same generation process as in section III-A. The generated points in the phase space represents the NN attractor ($\mathcal{A}_{\mathrm{NN}}$). We then randomly initialize 2000 trajectories on $\mathcal{A}_{\mathrm{L63}}$. Every trajectory from $\mathcal{A}_{\mathrm{L63}}$ is paired with another trajectory that starts from the closest point on $\mathcal{A}_{\mathrm{NN}}$, and in each pair of trajectories, the former follows the L63 flow while the latter follows the NN flow.

The FTLE of the trajectory pairs are compared under different integration steps: $N_t = 5, 50, 100, 500$ (Fig. 2). When $N_t = 5, 50$, NN accurately reproduces local divergence rates over the whole attractor, showing that the short-term predictability of the two systems agree with each other. As N_t increases, the correspondence diverges (N_t = 100), and converges again (N_t = 500) to the classical largest Lyapunov exponent of L63 (roughly 0.91 as in [33]). The convergence of FTLE under large timesteps implies similarity in the long-term behavior of the two systems.

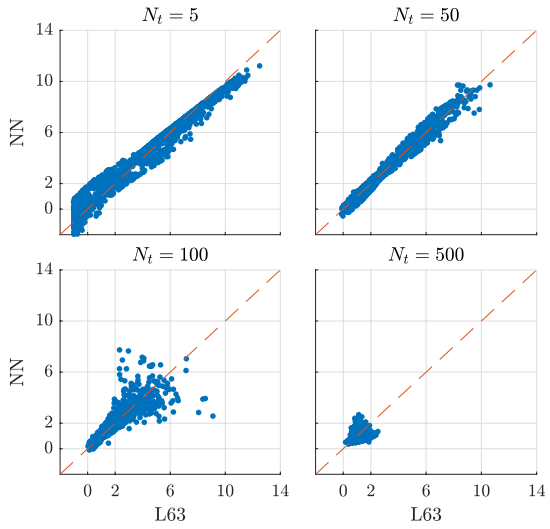

Fig. 2. One-to-one scatter plot of FTLE with L63 (x axis) and NN (y axis). The NN used in this plot is the same as that in Fig. 1. The panels (from left to right, top to bottom) correspond to increasing integration steps, N_t.

The agreement in FTLE generally improves under increasing numbers of neurons and training data points (Fig. 3). This trend is similar to the decreasing trend of the root-mean-square prediction errors (not shown), and it is expected if we invoke the bias and variance trade-off [34]: increased complexity in learning models such as neural networks generally translates into lower bias in prediction, provided that regularization techniques prevent the learning algorithm from entering the high-variance regime. Although it's the prediction errors that is being minimized during training, the errors in FTLE is also reduced simultaneously.

Remarkably, NN can extrapolate from an incomplete training dataset sampled from part of the attractor. Similar to Fig. 1, Fig. 4 shows a comparison of two trajectories predicted by NN and L63. The NN in this case has 5 neurons, and is trained on 100 data points sampled from the $X > -5$ part of $\mathcal{A}_{\mathrm{L63}}$, which amounts to knowing about 73% of the attractor structure. The two trajectories in Fig. 4 originate from the unknown region. They are close in the first 100 timesteps, and then bifurcate onto the two branches of the attractor. Despite starting from an unknown region, the NN trajectory still traces out a well-behaved object that closely resembles the original attractor in the extrapolated region of $X \leq -5$. The one-to-one

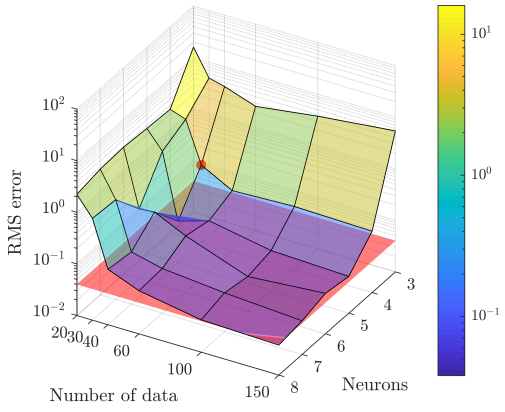

Fig. 3. The RMS error in FTLE of neural networks for each neuron and number-of-data configuration. The FTLE is calculated with $N_t = 50$ and averaged over 2000 trajectories that are randomly initialized on the attractor. The red dot represents the example configuration used in Figs. 1 and 2. The red surface is located at $z = 0.04$.

correspondence of FTLEs between L63 and the NN trained on the incomplete dataset is similar to Fig. 2 (see Fig. S2).

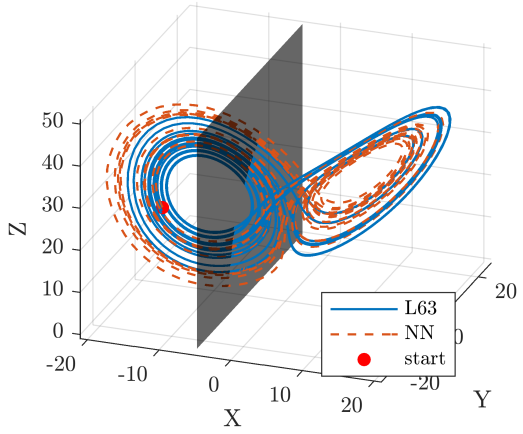

Fig. 4. Similar to Fig. 1, but the red-dashed trajectory is produced by a 5-neuron NN trained on 100 data points sampled from the $X > -5$ part of the attractor. The region to the right of the grey partition is the training data range, and the region to the left is unknown to the NN.

IV. A GEOMETRIC PERSPECTIVE OF THE NN FLOW

We showed from the previous section that the neural learnability on the L63 dynamics is very good. However, the theoretical approach to understand this learnability is unknown. Although the UAP states that mapping $\mathbf{\Phi}_{\mathrm{L63}}$ can be approximated by NN arbitrarily well, it does not explain the NN's efficacy in reconstructing the strange attractor with few neurons, nor its skill of extrapolation. Inspired by the exact mathematical correspondence between the geometric Lorenz flow and L63 [25], [35] (see section S2 for details), we give our geometric understanding of the NN flow.

A. Mathematical formulation

The dynamics of NN (3) can be seen as a mapping in a multi-dimensional Riemann space (this interpretation was also used in classification problems [36]). In the discrete map of the simple 4-neuron network discussed above, the input vector $\mathbf{x}$ in the 3-D phase space is mapped into a 4-D *neuron space*, and then mapped back to the phase space. Let an N_t-step trajectory be $\mathcal{L}_0^{N_t} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{N_t}\}$, $N_t \geq 2$. From step n to $n+1$ ($n \in \{0, 1, ..., N_t - 1\}$), there exists a 4-D intermediate vector $\mathbf{y}$ in the neuron space:

$$\mathbf{y}_{n+1} = g(\mathbf{W_1}\mathbf{x}_n + \mathbf{b_1}), \quad (n = 0, 1, ..., N_t - 1). \tag{5}$$

We refer to $\mathbf{y}$ as the *neuron vector*. The recurrence relation of $\mathbf{y}$ is then:

$$\mathbf{y}_{n+1} = g(\mathbf{W}^*\mathbf{y}_n + \mathbf{b}^*), \quad (n = 1, 2, ..., N_t - 1). \tag{6}$$

where $\mathbf{W}^* = \mathbf{W_1}\mathbf{W_2}$ is a 4-by-4 matrix, and $\mathbf{b}^* = \mathbf{W_1}\mathbf{b_2} + \mathbf{b_1}$ is a 4-by-1 vector. $\mathbf{W}^*$ can be decomposed as $\mathbf{W}^* = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$ using singular-value decomposition. $\mathbf{U}$ and $\mathbf{V}$ are both 4-D orthonormal matrices, and $\mathbf{S}$ is a diagonal matrix of rank 3. Eq. (6) can be written more explicitly as

$$\mathbf{y}_{n+1} = g(\mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}\mathbf{y}_n + \mathbf{b}^*), \tag{7}$$

which we call the *neuron map*. Equation (7) encodes the entire dynamics learnt by NN, since it is different from (3) by a homomorphism, i.e., (5). Therefore, understanding the neuron map is equivalent to understanding the dynamics of NN.

The neuron map has 4 sub-steps: rotation, stretch, rotation, and compression. *Rotation* in this paper takes the generalized sense of orthogonal transformation, and they are carried out by orthonormal matrices $\mathbf{V}^{\mathrm{T}}$ and $\mathbf{U}$ in the neuron map. Since the sigmoid function only has a compressing effect due to its gradient being smaller than or equal to 1, $\mathbf{S}$ must have at least one diagonal element larger than 1 in order to obtain one or more unstable directions as required by chaos [37]. For the 4-neuron NN at question, $\mathbf{S}$ applies expansion in two dimensions as two of its diagonal elements are greater than 1 (table I).

Through the growth of perturbations, the effects of compression and expansion exerted by NN are seen more clearly. Let $\delta\mathbf{y}$ be the difference between two initial points near $\mathbf{y}$, we know from (6) that its value at the next timestep, $\delta\mathbf{y}'$, is

$$\delta\mathbf{y}' = g'\left(\mathbf{W}^*\mathbf{y} + \mathbf{b}^*\right) \odot \mathbf{W}^*\delta\mathbf{y}, \tag{8}$$

where we neglected second- and higher-order terms, and $\odot$ denotes element-wise product. We set $G_{jj} = g'\left(\sum_{i=1}^{L} W^*_{ji}y_i + b^*_j\right)$, then $g'\left(\mathbf{W}^*\mathbf{y}_n + \mathbf{b}^*\right) \odot \mathbf{W}^*\delta\mathbf{y} = \mathbf{G}\mathbf{W}^*\delta\mathbf{y}$, where $\mathbf{G} = \mathrm{diag}\{G_{11}, G_{22}, ...\}$. The squared error is then

$$|\delta\mathbf{y}'|^2 = \delta\mathbf{y}^{\mathrm{T}}(\mathbf{W}^*)^{\mathrm{T}}\mathbf{G}^2\mathbf{W}^*\delta\mathbf{y}, \tag{9}$$

where $(\mathbf{W}^*)^{\mathrm{T}}\mathbf{G}^2\mathbf{W}^*$ is symmetric positive semi-definite. From (9), it's clear that the singular values of $\mathbf{W}^*$ that are larger than 1 will expand the perturbation since $\mathbf{G}$ compresses the perturbation as $g'(x) \in (0, 1]$, $\forall x \in \mathbb{R}$. With the information of $\mathbf{y}$, $\mathbf{G}$ controls the degrees of compression in each direction of the 4-D neuron space. The orientations of

compression and expansion can be in different directions as they are controled by $\mathbf{U}$ and $\mathbf{V}$.

The above framework can be easily generalized into an N-hidden-layer network. The dynamics of the neural vector will be ambiguous as there are multiple layers of hidden neurons, hence we apply the same treatment to perturbations in the phase space. For a perturbation of $\delta\mathbf{x}$ around $\mathbf{x}$, its squared length at the next timestep is

$$|\delta\mathbf{x}'|^2 = |\mathbf{W}_{N+1}\mathbf{G}_N\mathbf{W}_N...\mathbf{G}_1\mathbf{W}_1\delta\mathbf{x}|^2, \tag{10}$$

where $\mathbf{G}_i = \mathrm{diag}\{g'(\mathbf{W}_i\mathbf{y}_{i-1} + \mathbf{b}_i)\}$, $\mathbf{y}_{i-1}$ is the neuron vector of the ith layer for $i > 1$, and $\mathbf{y}_0 = \mathbf{x}$. The weight and gradient matrices then consecutively parameterize multiple stretching and compressing operations in a single NN map.

B. Topological mixing in NN with the Hénon map

The stretch and compression sub-steps in neuron maps are frequently thought of as the typical way to give rise to topological mixing and chaos (although strictly speaking, it is neither the necessary nor the sufficient condition [38]). The ability to obtain these geometric operations makes NN very good at approximating discrete chaotic mappings. For example, a 2-neuron NN can be trained to faithfully recreate the Hénon map. The Hénon map is a discrete 2-D chaotic map designed such that the phase space is stretched in one direction and compressed in the other [39]. The map can be decomposed into three steps: an area-preserving stretch, a compression, and a reflection along $x^{(1)} = x^{(2)}$:

$$\begin{aligned}
(x_1^{(1)}, x_1^{(2)}) &= (x_0^{(1)}, 1 - a(x_0^{(1)})^2 + x_0^{(2)}), && \text{(stretch)}\\
(x_2^{(1)}, x_2^{(2)}) &= (bx_1^{(1)}, x_1^{(2)}), && \text{(compression)}\\
(x_3^{(1)}, x_3^{(2)}) &= (x_2^{(2)}, x_2^{(1)}), && \text{(reflection)}
\end{aligned} \tag{11}$$

where we choose $a = 1.4$, $b = 0.3$. We follow the same training procedure as for L63, and train this 2-neuron network with only 20 randomly-sampled data points. The reconstructed strange attractor is virtually indistinguishable from the original Hénon attractor (Fig. S3). Table II shows the parameters of this network after training.

TABLE II
PARAMETERS OF THE 2-NEURON NN TRAINED WITH 20 DATA POINTS OF THE HÉNON MAP.

Matrix	Values		Bias	Values
$\mathbf{W}_1$	0.0960	0.0043	$\mathbf{b}_1^{\mathrm{T}}$	$0.8688, 0.9188$
	-0.0866	0.0041		
$\mathbf{W}_2$	220.7978	263.0327	$\mathbf{b}_2^{\mathrm{T}}$	$-344.5050, 0.5593$
	3.0292	-3.6975		

With this simple example, we show how the extension and compression take place in the neuron map. Let $\mathcal{H}$ be a group of points that form a straight line in the phase space. $\mathcal{H}$ is initialized into the neuron space by (5) as $\mathcal{H}^0$, shown as the blue dots in Fig. 5. Then $\mathcal{H}^0$ undergoes a series of geometric mapping following the NN flow. The location vector of each point in $\mathcal{H}^0$ is left-multiplied by $\mathbf{V}^{\mathrm{T}}$ and becomes $\mathcal{H}^1$. $\mathbf{V}^{\mathrm{T}}$ is a rotational matrix of 130.0° counter-clockwise. $\mathbf{S}$ linearly expands $\mathcal{H}_1$ in $y^{(1)}$ and contracts it in $y^{(2)}$, yielding $\mathcal{H}^2$. $\mathcal{H}^2$ is then reflected by $\mathbf{U}$ along a line with an angle of 69.0° relative to line $y^{(2)} = 0$, giving $\mathcal{H}^3$. $\mathcal{H}^3$ is transformed into $\mathcal{H}^4$ from bias by $\mathbf{b}^*$, and element-wise compression by $g(\cdot)$. $\mathcal{H}^4$ initiates the next step of the neuron mapping. Compared with $\mathcal{H}^0$, $\mathcal{H}^4$ extends along the principle direction of the point manifold and wraps around the lower-right tip of $\mathcal{H}^0$, which is effectively a horseshoe transformation.

V. LOWER-BOUNDING THE NUMBER OF NEURONS

The number of neurons that are necessary to reproduce the strange attractors of the Lorenz and Hénon maps are surprisingly small compared to previous theoretical results. Since the Euler-forward scheme of (1) is a 3-D ($n = 3$) polynomial with a degree of at most $d = 2$, we can use previous theoretical results on learning polynomials with NNs [40], [41] to establish lower bounds on the necessary number of neurons. In effect, we assume that the dynamics are characterized by polynomials but the learning system doesn't know the exact coefficients for each term. The number of neurons (L) for learning a polynomial with root-mean-square error target ϵ is bounded by $L = \Omega(n^{6d}/\epsilon^3)$ according to [41]. This is a rather coarse estimate as more than 5×10^5 nodes are needed when $\epsilon \sim 1$ (for Hénon map, this estimate is 4×10^3). On the other hand, matching the equilibrium norms of neural and polynomial regression [29] gives a more reasonable estimate: a full polynomial (n, d) needs $L = \binom{n+d}{d} - (n+1) \sim 6$ hidden nodes [28], [29] for an asymptotic match, and the standard network (3) has an asymptotic bound of $L \sim \frac{n}{2n+1}\left[\binom{n+d}{d} - 1\right] \sim 5$ neurons. Note that this match in learning dynamics is just an order-of-magnitude estimation and does not provide an error guarantee.

A more direct but less rigorous bound can be obtained via a Taylor-expansion of the sigmoid function to the third order: $\tanh(x) = x - x^3/3 + O(x^5)$, which allows (3) to be modeled as a polynomial of degree 3 (NN polynomial). We further require all coefficients of the NN polynomial to be equal to those in (1). Then for an NN with L hidden nodes, biases, and n-dimensional input/output, a total of $2nL + n + L$ parameters should satisfy $3\binom{n+3}{3}$ constraining equations. The parameters in NN should be under-determined for a good fit, i.e., $2nL + n + L \geq 3\binom{n+3}{3}$. Hence, at least $L = \lceil (3\binom{n+3}{3} - n)/(2n+1) \rceil = 9$ hidden nodes are needed. To obtain an error estimate, we substitute table I into the NN polynomial to obtain $\mathbf{\Phi}_{\mathrm{NN-poly}}$, and calculate the expected error over data sampled from the L63 attractor: $\epsilon^2 = \langle (\mathbf{\Phi}_{\mathrm{NN-poly}} - \mathbf{\Phi}_{\mathrm{L63}})^2 \rangle_{\mathcal{A}_{\mathrm{L63}}}$. 5000 random samples give a normalized error of $\epsilon \sim 0.14$. Therefore, this estimation gives a lower bound of 9 neurons at the error level of at most 0.14.

These theoretical lower-bounds give an estimate on the necessary complexity of neural networks needed to model polynomial systems. However, they overlook the geometric nature of the two chaotic systems discussed above, despite the fact that both of the systems are polynomials. This might be the reason why the lower-bounds are still equal or larger than the necessary number of neurons needed to reconstruct the strange attractors. The similarity between NN and chaotic

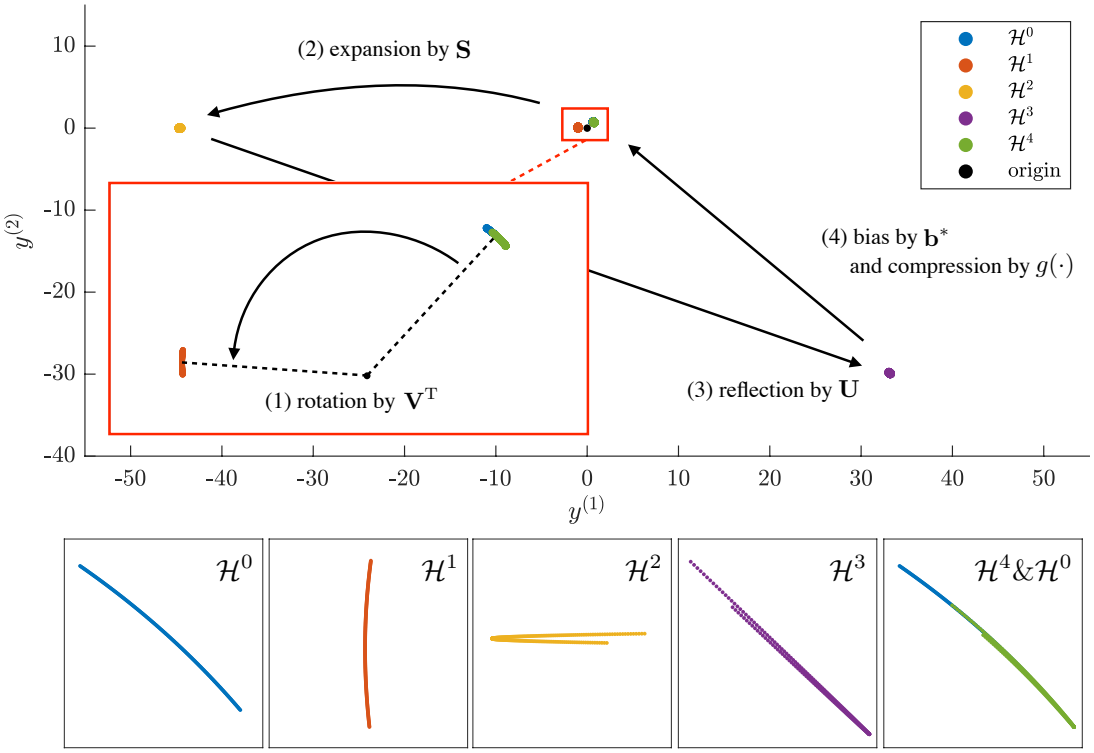

Fig. 5. Schematic of an iteration of the neuron map described by the 2-neuron NN trained on the Hénon map. The positions of points in the neuron space at each sub-step is shown in the upper figure, and the detailed structure is sketched in the lower panels. $\mathcal{H}^0$ (blue) is rotated counter-clockwise to $\mathcal{H}^1$ (orange), stretched and contracted to $\mathcal{H}^2$ (yellow), reflected to $\mathcal{H}^3$ (purple), and then compressed to $\mathcal{H}^4$ (green) which occupies the same region as $\mathcal{H}^0$. The first step is magnified in the inset.

maps in terms of the stretching and compression operations should therefore be stressed to explain NN's surprising efficacy in modeling chaotic dynamics.

VI. Conclusion and Discussion

We have shown that single-hidden layer feedforward neural networks are able to reconstruct the chaotic dynamics of the Lorenz-63 map and Hénon map with surprising efficacy. This success in modeling chaotic dynamics is associated with neural network's structural similarity to the chaotic maps in terms of the stretching and compression operations. Our work suggests that NN may be a good candidate to learn from data and represent a broad range of chaotic dynamics with good generalization skills. With the flow-like dynamics and gradient-descent algorithm, it may serve as a non-parametric model for chaotic systems without explicit expressions. Neural networks may be especially useful in data-driven problems with its efficacy in terms of the necessary complexity and number of data points that are needed to model chaotic dynamics. The explainability from the geometric viewpoint might even warrant neural networks as proper mathematical models for these systems.

Conversely, one could also consider neural networks as a more generalized class of chaotic systems. Apart from compression and expansion operations that are necessary for chaos, the higher-dimensional rotations are also important in creating the flow-like dynamics in modeling L63. We may further posit that the neural networks could be a unifying formulation for dissipative chaotic dynamics as it at least reproduces the Hénon map and the Lorenz map under the same mathematical framework.

On the other hand, the compression operation imposed by the sigmoid function makes NN of the form of (3) preferable to emulate low-dimensional dissipative systems; its ability to model chaotic Hamiltonian dynamics and systems of much higher dimensionality is yet to be tested. It's also interesting to test whether the choice of activation function, $g(\cdot)$, results in performance changes. For example, it may not be possible to model systems like L63 using only ReLU. Because ReLU is piece-wise linear in the phase space, it cannot obtain the nonlinear folding behavior. Indeed, linear models have been found to fail dramatically when modeling chaotic dynamics [8]. More work is also needed, possibly with the aid of Riemann geometry, to fundamentally understand the geometric operations in the high-dimensional neuron space.

Acknowledgment

Supports from ONR grant N00014-19-1-2273, the MIT Environmental Solutions Initiative, the John S. and Maryann Montrym Fund, and the MIT Lincoln Laboratory are gratefully acknowledged.

References

[1] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, 2nd ed. Boca Raton, FL: CRC Press, 2015.

[2] S. Ravela, *Tractable Non-Gaussian Representations in Dynamic Data Driven Coherent Fluid Mapping*. Cham: Springer International Publishing, 2018, pp. 29–46.
[3] K. Aihara1, T. Takabe, and M. Toyoda, "Chaotic neural networks," *Physics Letters A*, vol. 144, pp. 333–340, 1990.
[4] A. Garliauskas, "Neural network chaos analysis," *Nonlinear Analysis: Modelling and Control*, vol. 3, pp. 43–57, 1998.
[5] R. Bakker, J. C. Schouten, C. Lee Giles, F. Takens, and C. M. Van den Bleek, "Learning chaotic attractors by neural networks," *Neural Computation*, vol. 12, pp. 2355–2383, 2000.
[6] A. Das, A. Roy, and P. Das, "Chaos in a three dimensional neural network," *Applied Mathematical Modelling*, vol. 24, no. 7, pp. 511 – 522, 2000.
[7] A. Das, P. Das, and A. B. Roy, "Chaos in a three-dimensional general model of neural network," *International Journal of Bifurcation and Chaos*, vol. 12, pp. 2271–2281, 2002.
[8] S. V. Dudul, "Prediction of a Lorenz chaotic attractor using two-layer perceptron neural network," *Applied Soft Computing*, vol. 5, pp. 333–355, 2005.
[9] X.-S. Yang and Q. Li, "Horseshoe chaos in cellular neural networks," *International Journal of Bifurcation and Chaos*, vol. 16, pp. 157–161, 2006.
[10] J. M. Bahi, J. F. Couchot, C. Guyeux, and M. Salomon, "Neural networks and chaos: Construction, evaluation of chaotic networks, and prediction of chaos with multilayer feedforward networks," *Chaos*, vol. 22, p. 013122, 2012.
[11] Z. Song and J. Xu, "Bifurcation and chaos analysis for a delayed two-neural network with a variation slope ratio in the activation function," *International Journal of Bifurcation and Chaos*, vol. 22, p. 1250105, 2012.
[12] A. Zerroug, L. Terrissa, and A. Faure, "Chaotic dynamical behavior of recurrent neural network," *Annual Review of Chaos Theory, Bifurcations and Dynamical Systems*, vol. 4, pp. 55–66, 2013.
[13] L. Zhang, "Artificial neural networks model design of Lorenz chaotic system for EEG pattern recognition and prediction," in *2017 IEEE Life Sciences Conference*, 2017, pp. 39–42.
[14] ——, "Artificial neural network model design and topology analysis for FPGA implementation of Lorenz chaotic generator," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering*, 2017, pp. 30–33.
[15] R. Yu, S. Zheng, and Y. Liu, "Learning chaotic dynamics using tensor recurrent neural networks," in *Proceedings of the ICML 17 Workshop on Deep Structured Prediction*, 2017.
[16] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data," *Chaos*, vol. 27, p. 121102, 2017.
[17] M. Madondo and T. Gibbons, "Learning and modeling chaos using LSTM recurrent neural networks," in *Proceedings of the Midwest Instruction and Computing Symposium*, 2018.
[18] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *32nd Conference on Neural Information Processing Systems*, 2018.
[19] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach," *Physical Review Letters*, vol. 120, p. 024102, 2018.
[20] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, vol. 20, pp. 130–141, 1963.
[21] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.
[22] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.
[23] D. R. Seidl and R. D. Lorenz, "A structure by which a recurrent neural network can approximate a nonlinear dynamic system," in *Proceedings of the International Joint Conference on Neural Networks 1991*, vol. 2. IEEE, 1991, pp. 709–714.
[24] K. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, pp. 801–806, 1993.
[25] W. Tucker, "A rigorous ODE solver and smale's 14th problem," *Foundations of Computational Mathematics*, vol. 2, pp. 53–117, 2002.
[26] P. Berge, Y. Pomeau, and C. Vidal, *Order within chaos*. Wiley, 1987.
[27] L. G. Valiant, "A theory of the learnable," *Commun. ACM*, vol. 27, pp. 1134–1142, 1984.
[28] M. Trautner and S. Ravela, "Neural integration of continuous dynamics," *arXiv*, no. 1911.10309, 2019. [Online]. Available: https://arxiv.org/pdf/1911.10309.pdf
[29] S. Ravela, Z. Li, M. Trautner, and S. Reilly, "Spectral matching for theory-driven neural computation," *Preprint*, 2019.
[30] F. Dan Foresee and M. T. Hagan, "Gauss-Newton approximation to bayesian learning," in *Proceedings of International Conference on Neural Networks*, 1997.
[31] G. Haller, "Distinguished material surfaces and coherent structures in three-dimensional fluid flows," *Physica D: Nonlinear Phenomena*, vol. 149, pp. 248–277, 2001.
[32] L. Barreira and Y. Pesin, "Lectures on Lyapunov exponents and smooth ergodic theory," in *University Lecture Series*. Providence, RI: American Mathematical Society, 2002, vol. 23, p. 151.
[33] D. Viswanath, "Lyapunov exponents from random fibonacci sequences to the Lorenz equations," Ph.D. dissertation, Cornell University, Ithaca, NY, USA, 1998.
[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.
[35] J. Guckenheimer and R. F. Williams, "Structural stability of Lorenz attractors," *Publ. Math. IHES*, vol. 50, pp. 307–320, 1979.
[36] M. Hauser and A. Ray, "Principles of Riemannian geometry in neural networks," in *31st Conference on Neural Information Processing Systems*, 2017, p. 10.
[37] R. L. Devaney, *An introduction to chaotic dynamical systems*, 2nd ed. Reading, MA: Addison-Wesley, 1989.
[38] D. Ruelle, "What is a strange attractor?" *Notices of the AMS*, vol. 53, pp. 764–765, 2006.
[39] M. Hénon, "A two-dimensional mapping with a strange attractor," *Communications in Mathematical Physics*, vol. 50, pp. 69–77, 1976.
[40] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, pp. 930–945, 1993.
[41] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang, "Learning Polynomials with Neural Networks," in *Proceedings of the 31st International Conference on International Conference on Machine Learning*, 2014.

Ziwei Li Ziwei Li is a PhD candidate in the Department of Earth, Atmospheric and Planetary Sciences at the Massachusetts Institute of Technology. He is interested in discovering the dynamics of the atmosphere using simple physical and stochastic models. Ziwei Li received his Bachelor of Science degree in 2016 from Peking University.

Sai Ravela Sai Ravela directs the Earth Signals and Systems Group (ESSG) in the Earth, Atmospheric and Planetary Sciences at the Massachusetts Institute of Technology. His primary research interests are in dynamic data-driven stochastic systems theory and machine intelligence methodology with application to Earth, Atmospheric and Planetary Sciences. Ravela received a PhD in Computer Science in 2003 from the University of Massachusetts at Amherst.