Lyapunov-Guided Embedding for Hyperparameter Selection in Recurrent Neural Networks

Ryan Vogt
University of Washington
Yang Zheng
University of Washington
Eli Shlizerman (shlizee@uw.edu)
University of Washington https://orcid.org/0000-0002-3136-4531

Article

Keywords: Recurrent Neural Networks, Hyperparameter Optimization, Networked Dynamical Systems, Lyapunov Exponents, Deep Learning

Posted Date: April 7th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1433482/v1

Lyapunov-Guided Embedding for Hyperparameter Selection in Recurrent Neural Networks

Ryan Vogt[1], Yang Zheng[2] and Eli Shlizerman[1,2*]

[1*]Department of Applied Mathematics, University of Washington, Seattle, 98195, WA, US.
[2*]Department of Electrical and Computer Engineering, University of Washington, Seattle, 98195, WA, US.

*Corresponding author(s). E-mail(s): shlizee@uw.edu;
Contributing authors: ravogt95@uw.edu; zheng94@uw.edu;

Abstract

Recurrent Neural Networks (RNN) are ubiquitous computing systems for sequences and multivariate time series data. While several robust architectures of RNN are known, it is unclear how to relate RNN initialization, architecture, and other hyperparameters with accuracy for a given task. In this work, we propose to treat RNN as dynamical systems and to correlate hyperparameters with accuracy through Lyapunov spectral analysis, a methodology specifically designed for nonlinear dynamical systems. To address the fact that RNN features go beyond the existing Lyapunov spectral analysis, we propose to infer relevant features from the Lyapunov spectrum with an Autoencoder and an embedding of its latent representation (AeLLE). Our studies of various RNN architectures show that AeLLE successfully correlates RNN Lyapunov spectrum with accuracy. Furthermore, the latent representation learned by AeLLE is generalizable to novel inputs from the same task and is formed early in the process of RNN training. The latter property allows for the prediction of the accuracy to which RNN would converge when training is complete. We conclude that representation of RNN through Lyapunov spectrum along with AeLLE, and assists with hyperparameter selection of RNN, provides a novel method for organization and interpretation of variants of RNN architectures.

Keywords: Recurrent Neural Networks, Hyperparameter Optimization, Networked Dynamical Systems, Lyapunov Exponents, Deep Learning

Introduction

Recurrent Neural Networks (RNN) specialize in processing sequential data, such as natural speech or text, and broadly, multivariate time series data [1–3]. With such omnipresent data, RNN address a wide range of tasks in the form of prediction, generation, and translation for a myriad of applications which include stock prices, markers of human body joints, music composition, spoken language, and sign language [4–11]. While RNN are ubiquitous systems, these networks cannot be easily explained in terms of the architectures they assume, the parameters that they incorporate, and the learning process that they undergo. As a result, it is not straightforward to associate an optimally performing RNN with a particular dataset and a task. The difficulty stems from RNN characteristics making them intricate dynamic systems. In particular, RNN can be classified as *(i) nonlinear (ii) high-dimensional (iii) non-autonomous* dynamical systems with *(iv) varying*

parameters, which are either global (hyperparameters) or connectivity weights during training.

RNN are defined as multi-layered systems where each layer is viewed as a sequential composition of a function $\boldsymbol{f}$ for each time step of the input sequence. The equations describing a generic RNN are

$$h_{t+1} = \boldsymbol{f}(h_t, x_t) = \sigma(\mathbf{U}x_t + \mathbf{V}h_t + b), \quad (1)$$
$$y_t = \mathbf{W}h_t, \quad (2)$$

where h_t are the hidden states of the RNN, t is the time step, x_t is the input, U are the connectivity weight parameters for the input, V are the connectivity weight parameters for hidden (internal) states, b is the bias term, and σ is a non-linear activation function (e.g., sigmoid or tanh). The hidden states h_t are translated to the predicted output y_t, where W are the connectivity weight parameters for the output [12].

For RNN to be effective in handling different time scales, it is important to employ the concept of parameter sharing across inputs of different length. This allows the network to generalize across inputs and identify similar features. Since input data may exhibit these features in different orders or duration, the relationship between elements in a sequence must be learnt by the network hidden states through their connectivity parameters.

Such requirements introduce several fundamental bottlenecks that hinder the performance of generic RNN. The first bottleneck is due to *long-term propagation of gradients during training* which typically results in vanishing or exploding gradients [13]. In addition, even when the propagation of gradients is stabilized, another bottleneck stems from RNN being a dynamic map constituting the *composition of the function* $\boldsymbol{f}$ at each time step being a dynamic map. The long-term repetition of the composition may amplify small inaccuracies at individual time step to large contributions.

In order to have a network which can propagate and learn longer sequence in a robust way, it was shown that the eigenvalues of $\boldsymbol{f}$ need to be close to zero [14]. Furthermore, architectures of RNN, such as Long Short-Term Memory (LSTM) [15] and Gated Recurrent Units

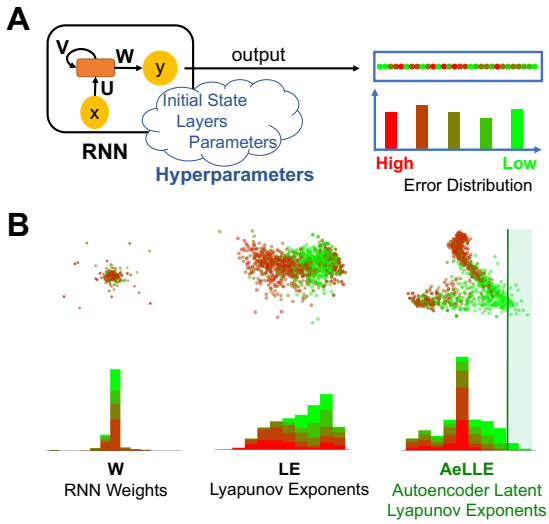

Fig. 1 RNN performance for the same task varies significantly with hyperparameter variation. A) RNN variants randomly initialized with initial states and hyperparameters exhibit a wide distribution of accuracy on the target learning task. B) Clustering of variants with various embeddings (left to right). PCA 2D embedding of the connectivity weights (left), W in Eq. 1 does not result in useful clustering. PCA 2D embedding of LE spectrum (middle) separates the variants based on accuracy better. AeLLE 2D embedding (right) is capable, by a vertical line classifier to separate variants and isolate high accuracy variants with high precision.

(GRU)[16, 17], have been proposed to mitigate the extreme effects of these long-term dependencies. LSTM and GRU remedy these problems by incorporating gates (self-loops) which allow the network to learn the amount of information that each component should preserve from previous time steps. By implementing this structure, LSTMs and GRUs are able to learn long-term dependencies, making them useful for a wide variety of applications [9, 18–25].

While such specific architectures are intuitive and robust, they represent singular points in the space of RNN. In addition, for these architectures, the accuracy still depends on hyperparameters, among which are the initial state of the network, initialization of the network connectivity weights, optimization settings, architectural parameters, and the input statistics. Each of these factors has the potential to impact network learning and as a consequence task performance. Indeed, by fixing the task and randomly sampling hyperparameters and inputs, the accuracy of otherwise identical

networks can vary significantly, as demonstrated in Fig. 1A.

Mitigation of the sensitivity on error gradient propagation and informing parameter selection typically involves studying the stability of the propagation of the recursive compositions of the hidden states, i.e., h_t [14]. Since vanishing and exploding gradients arise from long products of Jacobians of the hidden states dynamics whose norm could exponentially grow or decay, much effort has been made to mathematically describe the link between model parameters and the eigen- and singular-value spectra of long products [26–29]. For architectures used in practice, these approaches appear to have a limited scope [30, 31]. This is likely due to spectra having non-trivial properties reflecting intricate long-time dependencies within the trajectory, and due to the need to take into account the dynamic nature of RNN systems.

The identification of RNN as dynamical systems and as such developing appropriate analyses appears as a prospective direction. Recently, dynamical systems methodology has been applied to introduce constraints to RNN architecture to achieve better robustness, such as orthogonal (unitary) RNN [32–38] and additional architectures such as coupled oscillatory RNN [39] and Lipschitz RNN [40]. These approaches set network weights to form dynamical systems which have the desired Jacobians for long-term information propagation. In addition, analyses such as stochastic treatment of the training procedure have been shown to stabilize various RNN [41]. Furthermore, a universality analysis of fixed points of different RNN, proposed in [42], hints that RNN architectures could be organized in similarity classes such that despite having different architectural properties would exhibit similar dynamics when optimally trained. In light of this analysis, it remains unclear to which similarity classes in the space of RNN the constrained architectures belong and what is the distribution of the architectures within each class. These unknowns warrant development of dynamical systems tools that *characterize and classify RNN variants*.

A powerful dynamical system method for characterization and predictability of dynamical systems is *Lyapunov Exponents (LE)* [43, 44]. LE capture the information generation by the system's dynamics through measurement of the separation rate of infinitesimally close trajectories. The collection of all LE is called LE spectrum (see Fig. 2). When LE are computed from observed evolution of the system, Oseledets theorem guarantees that LE characterize each ergodic component of the system, i.e., when long enough evolution of a trajectory in an ergodic component is sampled, the computed LE spectrum is guaranteed to be the same (see Methods section for details of computation) [45, 46]. Efficient approaches and algorithms have been developed for computing LE spectrum [47]. These have been applied to various dynamical systems including hidden states of RNN and variants such as LSTM and GRU [48]. Multiple features of LE spectrum provide an insight into an underlying dynamical system. For example, the maximal LE will determine the linear stability of the system [49]. Further, a system having LE greater than zero represents chaotic dynamics with the magnitude of the first exponent indicating the degree of chaos; when the magnitude decreases and approaches zero, the degree of chaos decreases as well. When all exponents are negative, the dynamics converge towards a fixed point attractor. Zero exponents represent limit cycles or quasiperiodic orbits [50, 51]. Many additional features of LE, even non-direct, correspond to properties of the dynamical system. For example, the mean exponent determines the rate of contraction of full volume elements and is similar to the KS entropy [52]. The LE variance measures heterogeneity in stability across different directions and can reflect the conditioning of the product of many Jacobians. In the study of turbulence [53], LE close to and below zero in a chaotic system have been showed to align with the inertial subrange of the turbulent system [54].

These features are descriptive and hold for nonlinear autonomous dynamical systems which share with RNN the characteristics of being *(i) nonlinear* and *(ii) high-dimensional*. However, RNN possess two additional characteristics of *(iii) non-autonomous* systems due to the hidden states dynamics h_t driven by inputs x_t, and *(iv) varying parameters*. Computing meaningful LE spectrum and preserving its features might still be plausible for RNN with non-autonomous inputs and fixed parameters, as described in [55]. This approach relies on the theory of random dynamical systems

which establishes LE spectrum even for a system driven by a noisy random input sequence sampled from a stationary distribution [56]. Analytical foundations employ uncorrelated Gaussian inputs, however, the framework is expected to apply to a wider range of well-behaved input statistics. This includes those with finite, low-order moments and finite correlation times like character streams from written language and sensor data from motion capture systems. The time course of the driven dynamics depends on the specific input realization, but critically, the theory guarantees that the stationary dynamics for all input realizations share the same stability properties which will, in general, depend on the input distribution, e.g., its variance.

While including non-autonomous inputs in the analysis of RNN LE spectrum appears feasible, very little is known about LE spectrum features for systems with varying parameters, especially systems like RNN which have a high number of parameters making them sensitive to parameter variation and impractical to analyze on a per-parameter basis. In fact, contradictory examples in [55] and in Fig. 2 suggest that the known features of LE spectrum are not directly correlated with RNN robustness and accuracy. Such inconsistency motivates our work where we develop a data driven methodology, called *AeLLE*, to infer LE spectrum features and associate them with RNN performance. The methodology implements an Autoencoder (Ae) which learns through its *L*atent units a representation of *LE* spectrum (LLE) and correlates the spectrum with the accuracy of RNN for a particular task. The latent representation appears to be low dimensional and interpretable such that even a simple linear embedding of the representation, denoted as AeLLE, corresponds to a classifier for selection of optimally performing parameters of RNN based on LE spectrum. We show that once AeLLE is trained, it holds for novel inputs and we also investigate the correlation between AeLLE classification accuracy and needed RNN training.

Methods

The proposed AeLLE methodology consists of three steps: 1) Computation of LE spectrum, 2) Autoencoder for LE spectrum, and 3) Embedding of Autoencoder Latent representation.

Computation of LE [48, 55]

We compute LE by adopting the well-established algorithm [57, 58] and follow the implementation in [48, 55]. For a particular task, each batch of input sequences is sampled from a set of fixed-length sequences of the same distribution. We choose this set to be the validation set. For each input sequence in a batch, a matrix $\mathbf{Q}$ is initialized as the identity to represent an orthogonal set of nearby initial states. The hidden states h_t are initialized as zeros.

To track the expansion and the contraction of the vectors of $\mathbf{Q}$, the Jacobian of the hidden states at step t, $\mathbf{J}_t$, is calculated and then applied to the vectors of $\mathbf{Q}$. The Jacobian $\mathbf{J}_t$ can be found by taking the partial derivatives of the RNN hidden states at time t, h_t, with respect to the hidden states at time at $t-1$, h_{t-1}

$$[\mathbf{J}_t]_{ij} = \frac{\partial \mathbf{h}_t^j}{\partial \mathbf{h}_{t-1}^i}. \tag{3}$$

Beyond the hidden states, the Jacobian will depend on the input x_t. This dependence allows us to capture dynamics of a network as it responds to input. The expansion factor of each vector is calculated by updating $\mathbf{Q}$ by computing the QR decomposition at each time step

$$\mathbf{Q}_{t+1}, \mathbf{R}_{t+1} = QR(\mathbf{J}_t \mathbf{Q}_t). \tag{4}$$

If r_t^i is the expansion factor of the i^{th} vector at time step t – corresponding to the i^{th} diagonal element of $\mathbf{R}$ in the QR decomposition– then the i^{th} LE λ_i resulting from an input signal of length T is given by

$$\lambda_k = \frac{1}{T} \sum_{t=1}^{T} \log(r_t^k) \tag{5}$$

The LE resulting from each input x^m in the batch of input sequences are calculated in parallel and then averaged. For each experiment, the LE were calculated over a fixed number of time steps with n different input sequences. The mean of n resulting LE spectra is reported as the LE spectrum. To normalize the spectra across different network sizes and consequently the number of LE in the spectrum, we interpolate the spectrum such that it retains the shape of the largest network size.

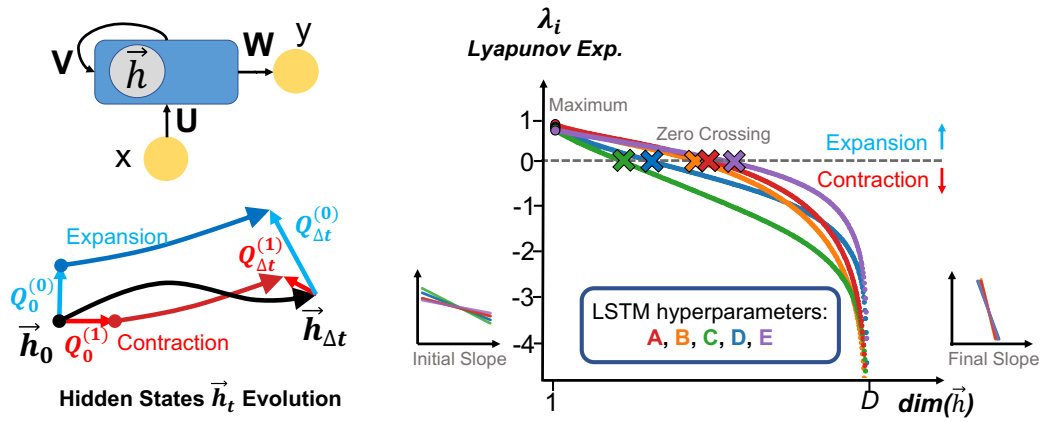

Fig. 2 LE Spectrum of RNN variants. RNN (top left) hidden states h evolution is tracked to calculate the LE spectrum of the network. The exponents are found by calculating the expansion and the contraction of nearby trajectories over time (bottom left). LE spectra correspond to points on LE spectrum curves (right). Variation of hyperparameters will correspond to distinct spectra curves. Basic known LE features such as Maximum, Initial Slope, Final Slope, Number of exponents greater than zero marked on the plot, do not directly correlate to task accuracy.

Through this interpolation, we can represent the LE spectra as curves. Spectra curves will have the same number of LE points for small networks and for larger networks.

Autoencoder for LE spectrum

As illustrated in Fig. 1, the computed LE spectrum appears to organize RNN variants and correlate them with accuracy better than direct quantities of RNN architecture, e.g., the connectivity weights of the output layer W and their embeddings. However, it is also evident that the correlation is subtle and depends on the choice of the embedding (features) of LE spectrum. In fact, when hyperparameters are varied, we observe that direct LE spectrum features are not correlated with performance (see Fig. 2 insets of maximum, zero-crossing, initial slope and final slope) . This warrants association of a robust embedding that will extract key correlative features between LE spectrum and RNN accuracy. For such purpose, Autoencoder methodology has been introduced as a methodology for learning robust and nonlinear embeddings from data and employed in numerous feature extraction applications [59–63].

Autoencoders are multilayered neural networks consisting of two components: an *encoder* network ϕ and a *decoder* network ψ. The encoder receives an input data $\boldsymbol{Z}$ and includes multiple layers that shrink in dimension as the input propagates through the layers. The last layer of the encoder, and typically the most compact in dimension, is denoted as the Latent layer. The decoder is a multi-layered network as well, receiving input from the Latent layer and is typically a reflection of the encoder, such that each layer expands in dimension up to the decoder's last (output) layer of same dimension as the first (input) layer of the encoder, see Fig. 3. The Autoencoder is trained for the task of reconstruction, namely the output of the decoder is optimized to closely match the input into the encoder. Over the course of training, the Latent layer becomes representative of the variance in the input data and extracts key features that might not immediately be apparent in the input. In addition to the reconstruction task, it is possible to include constraints on the optimization by the formulation of a loss function for the Latent layer values (Latent space), e.g., a classification or prediction criterion. This can constrain the organization of values in the Latent space [13, 64]. We propose to adopt the Autoencoder methodology for correlating LE spectra and RNN task accuracy. In this setup, the input into the encoder of the Autoencoder of LE spectrum of RNN (Ae) is the LE spectrum and denoted as $\boldsymbol{Z}$. Ae consists of a fully-connected encoder network ϕ, a decoder network ψ, and a prediction network θ defined by

$$\begin{aligned} \hat{Z} &= (\psi \circ \phi)Z, \\ \hat{T} &= (\theta \circ \phi)Z, \end{aligned} \tag{6}$$

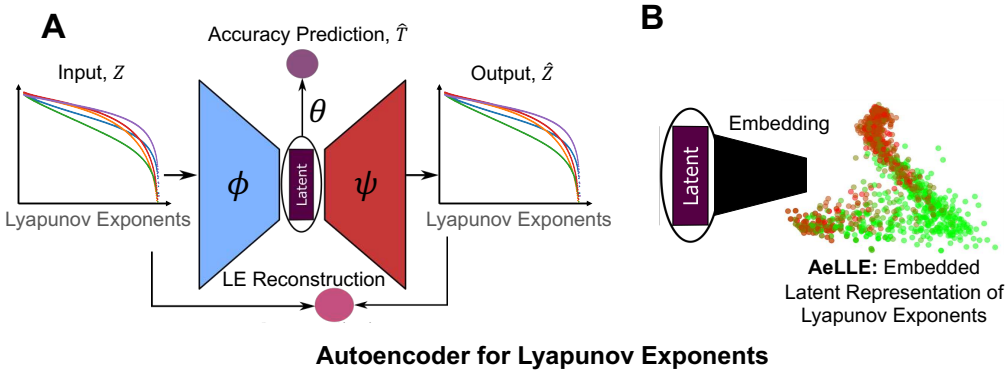

Fig. 3 AeLLE: LE spectrum Autoencoder and Latent Representation Embedding. A) LE Autoencoder is set to reconstruct LE spectrum (LE Reconstruction loss) and correlate it with task accuracy (Accuracy Prediction loss). B) The Latent space of the LE spectrum Autoencoder correlates LE spectrum and accuracy. Embedding of Latent space representation provides a low-dimensional clustering and classification space. We find that Latent space clusters the LE spectra well such that simple embedding (PCA) and simple linear classifiers (hyperplane, hyperellipse or threshold) classify RNN variants according to accuracy.

where $\hat{\boldsymbol{Z}}$, and $\hat{\boldsymbol{T}}$, correspond to the output from the decoder and predicted accuracy, respectively, with loss of $L = Z - \hat{Z}^2 + \alpha \cdot T - \hat{T}_n$. Ae performs the reconstruction task, optimization of the first term of Eq. 7, mean-squared reconstruction error of LE spectrum, as well as prediction of the associated RNN accuracy $\boldsymbol{T}$ (best validation loss), the second term of Eq. 7.

$$\phi, \psi, \theta = \arg\min_{\phi,\psi,\theta}(Z - \hat{Z}^2 + \alpha \cdot T - \hat{T}_n). \quad (7)$$

The parameter n can defined based on the desired behavior. The most common choices are the $n =$ 1-norm and $n = 2$-norm.

During training of Ae, the weight α in the prediction loss is gradually being increased so that Ae emphasizes RNN accuracy prediction once the reconstruction error has converged. We found that this approach allows to capture features of both RNN dynamics and accuracy. A choice of α being too small leads to dominance of the reconstruction loss such that the correlation between LE spectrum and RNN accuracy is not captured. Conversely, when α is initially set to a large value, the reconstruction along with the prediction diverge. The convergence of Ae for different RNN variants, as we demonstrate in Results section, shows that correlative features between LE spectrum and RNN accuracy can be inferred. The dependency of Ae convergence on a delicate balance of the two losses reconfirms that these features are tangled and thus the need for Ae embedding. We describe the settings of α and additional Ae implementation details in Supplementary Materials.

Embedding of Autoencoder Latent Representation

When the loss function of Ae converges, it indicates that the Latent space captures the correlation between LE spectrum and RNN accuracy. However, an additional step is typically required to achieve an organization of the Latent representation based on RNN variants accuracy. For this purpose, a low dimensional embedding, denoted as AeLLE, of the Latent representation needs to be implemented. An effective embedding would indicate the number of dominant features needed for the organization, provide a classification space for the LE spectrum features and connect them with RNN parameters. We propose to apply the Principal Component Analysis (PCA) embedding to the Latent representation [65, 66]. The embedding consists of computing PCA (PC modes) and projecting the representation on the dominant PC modes (e.g. 2 or 3 PC modes). Notably, while other, nonlinear, embeddings are possible, e.g., tSNE or UMAP [67, 68], when PCA results in a few dominant PC modes and a projection of them results in effective organization it indicates that the Latent representation has successfully captured the characterizing features of the correlation. We show in the Results section that, for

all examples of RNN architectures and tasks that we considered, the PCA embedding is sufficient to provide an effective space. In particular, in this space, most accurate RNN variants (green) can be clustered from other variants (red) through a simple clustering procedure, either a hyperplane or a hyperellipse (see Fig. 4).

Results

To investigate application and generality of our proposed method, we consider tasks with various inputs and outputs and various RNN architectures that have been demonstrated as effective models for these tasks. In particular, we choose three tasks: Signal Reconstruction, Character Prediction and Sequential MNIST. All three tasks involve learning temporal relations in the data with different forms of the input and objectives of the task. Specifically, the inputs range from low-dimensional signals to categorical character streams to pixel greyscale values. Nonetheless, across this wide variety of input data and tasks, the AeLLE space and clustering is consistently able to separate variants of hyperparameters according to accuracy (indicated by colors and elliptical regions in Fig. 4) in a way that is more informative than network hyperparameters alone.

More specifically, the Signal Reconstruction task, also known as target learning, a random RNN is being tracked to generate a target output signal from a random input [31, 69]. This task involves intricate time-dependent signals and a generic RNN for which dynamics in the absence of training are chaotic. Character Prediction is a common task which takes as an input a sequence of textual characters and outputs the next character of the sequence. This task is a rather simple task and used to benchmark various RNN variants. In particular, LSTM models have been shown to be successful in this task. Sequential MNIST is a more extensive benchmark for RNN classification accuracy. The input in the task is an image of a handwritten digit unrolled into a sequence of numerical values (pixels greyscale values) and the output is a corresponding label of the digit. We investigate LSTM performance on this task which was demonstrated to achieve high accuracy. We describe the outcomes of AeLLE application and resulting insights per each task below.

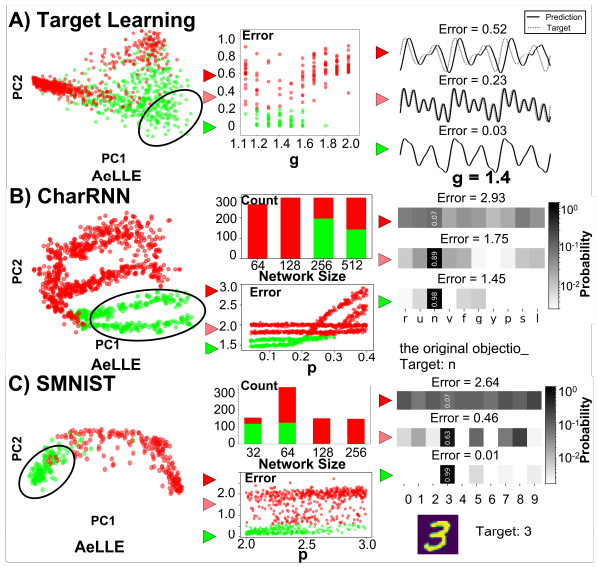

Fig. 4 Application of AeLLE to examples of RNN and tasks. For each example shown left to right: 2D AeLLE space and classifier, error distribution as function of hyperparameters, examples of output for three error values. **A) Target Learning with Rank1-RNN.** While g hyperparameter indicates possibility to train a network with low-error, each g value includes both low and high error variants (middle). AeLLE with 2D PCA embedding clusters the variants such that low error (< 0.2) ones (across g values) are located in the bottom-right of the plane. For such an error, the outputs are visually indistinguishable from the target (right). **B) Character Prediction (CharRNN) with LSTM.** The network size hyperparameter of 256 and 512 appear optimal for accuracy for the CharRNN task. Different initialization can impact the accuracy for each network size (middle). AeLLE 2D PCA (left) clusters variants according to network size branches and in addition organizes variants into groups of high and low error variants (green < 1.75). Low error region can be identified with an ellipse and examples show the probability maps for characters with such error (right). **C) Sequential MNIST classification (SMNIST) with LSTM.** The network size hyperparameter of 32 and 64 appear optimal for accuracy for the SMNIST task (middle) in contrast to CharRNN task. AeLLE 2D PCA (left) clusters variants into an arc shape where low error variants (green < 0.5) are all separated from high error variants (red) in the left segment of the arc. Low error region can be identified with an ellipse and examples show the probability maps for digit recognition with such error (right).

Target Learning with Random RNN

To examine how AeLLE interprets generic RNN with time evolving signals as output and input, we test Rank-1 RNN. Such a model corresponds to training a single rank of the connectivity matrix, the output weights W, on the task of target learning (Eq. 2 only). We set the target signal (output) to be a four-sine wave, a benchmark used in [69]. A key parameter in Rank-1 RNN is the amplification factor of the connectivity, g, which controls the output signal in the absence of training. For $g \leq 1$, the output signal is zero, while for $g \geq 1.8$ the output signal is strongly chaotic. In the interval $1 < g < 1.8$ the output signal is weakly chaotic. Previous work has shown that the network can generate the target when it is in the weakly chaotic regime, i.e., $1 < g < 1.8$ and trained with FORCE optimization algorithm [69, 70].

However, not all samples of the random connectivity correspond to accurate target generation. Even for g values in the weakly chaotic interval, there would be Rank-1 RNN variants that fail to follow the target (see Fig. 4A). Thereby, the target learning task, Rank-1 RNN architecture, and FORCE optimization are ideal candidates to test whether AeLLE can organize the variants of Rank-1 RNN models according to accuracy. The candidate hyperparameters for variation would be in 1) sampling of fixed connectivity weights (from normal distribution) and 2) the parameter g from the weakly chaotic regime. We structure the benchmark set to include 1200 hyperparameter variants and compute LE spectrum for each of them. After training is complete, the set of LE in the validation set are projected onto the Autoencoder's AeLLE space, depicted in Fig. 4A-left.

Our results show that AeLLE organizes the variants in a 2D space according to accuracy. The variants with low error values (< 0.2) are colored in *green* and variants with larger error values (> 0.2) are colored in *red*. With this threshold, only 22% of networks are classified as low-error and 78% as large error. We demonstrate the disparity in the signals that different error values correspond to in Fig. 4A-right. AeLLE space succeeds to correlate LE spectrum with accuracy such that most low-error networks are clustered in the bottom-right of the two-dimensional projection (see Fig. 4A), whereas large-error networks are concentrated in the left and top of the region shown. The coloring demonstrates that the red and the green regions can be easily separated using a hyperplane or a hyperellipse and multiple candidates as top performing variants can be identified with such a simple classification. Comparison of AeLLE clustering with a direct clustering according to values of g, Fig. 4A-left vs Fig. 4A-middle, shows that while most networks with $g < 1.6$ include variants with low-error, there are also variants with high-error for each value of g. This is not the case for all variants in the low-error hyperellipse of the AeLLE space. These variants have different g and connectivity values and sampling from the hyperellipse provides a higher probability Rank-1 RNN variant to be accurate.

Character Prediction with LSTM

Multiple RNN tasks are concerned with non-time dependent signals, such as sequences of characters in a written text. Therefore, we test AeLLE on LSTM networks that perform the character prediction task (CharRNN) in which for a given sequence of characters (input) the network predicts the character (output) that follows. In particular, we train LSTM networks on English translation of Leo Tolstoy's *War and Peace*, similar to the setup described in [71]. In this setup, each unique character is assigned an index (number of unique characters in this text is 82), and the text is split into disjoint sequences of a fixed length $l = 101$, where the first $l - 1 = 100$ characters represent the input, and the final character represents the output. The loss is computed as the cross entropy loss between the expected character index and the output one.

The hyperparameters of network size (number of hidden states) and initialization of weight parameters appear to have most impact on the accuracy. We create 1200 variants of these parameters, varying the number of hidden units from 64 to 512 and sample initial weights from a symmetric uniform distribution with the parameter p denoting the half-width of the uniform distribution from which the initial weights are sampled in the range of $[0.04, 0.4]$. We split the variants into Autoencoder training set (80%) and validation set (20%). The validation is depicted in Fig. 4B-left.

Similarly to the target learning task, we mark the variant networks according to accuracy. LSTM

networks with loss < 1.75 are considered as low-error (*green*) which includes 28.3% of networks, while the rest of the networks with larger loss are considered as large-error (*red*). We depict examples of error values and the impact on character prediction in Fig. 4B-right. Similarly to the target learning example, nor the network size, nor the initialization parameter p alone provide clear separation between low-error and large-error variants and need to be set to optimal values together, Fig. 4B-middle. Previous investigations indicated that larger network sizes would be preferable for achieving better accuracy. This is indeed reflected in the variants that we have examined. Smaller networks (size 64 and 128) do not reach low-error accuracy. Larger networks can attain low-error, however, we observe that much larger networks (size 512) are not necessarily more accurate than mid size networks (size 256) across variants that we compared.

AeLLE analysis, based on LE spectrum, clearly reflects these observations. We find that AeLLE in 2D space separates the spectra of the variants into four branches which correspond to the four network sizes that were varied. The branches corresponding to smaller networks are located in the top of the AeLLE plane, whereas larger networks are in either the lower half or the far left of the AeLLE plane. Among larger networks, the set of low-error variants is stretched to the right of the bottom region (marked by ellipse), meanwhile, the higher-error networks, even of larger size, are all located in the top and left regions of the AeLLE plane, see Fig. 4B-left. Effectively, AeLLE disentangles the dependence on the two parameters and succeeds to cluster high-accuracy networks into an easily identifiable cluster.

Sequential MNIST Classification with LSTM

LSTM networks have been shown to be applicable in a variety of sequence related tasks. A common benchmark for LSTM is the sequential MNIST task (SMNIST) applied on MNIST dataset [72]. In this task, the input is a sequence of pixel greyscale values unrolled from an image of handwritten digits from $0-9$. The output is a prediction of the corresponding label (digit) written in the image. We follow the SMNIST task setup in [73], where each image is treated as sequential data and each row is the input at one time, and number of time steps is equal to the number of columns in the image. The loss corresponds to the cross entropy between the predicted and the expected one-hot encoding of the digit. As in CharRNN, we vary two hyperparameters, network size chosen from values of $32-256$, and initialization parameter p sampled from the range of $[2.0, 3.0]$.

Similarly to previously described tests, we color code the variants according to accuracy. LSTM networks with loss < 0.5 are considered as low-error (*green*) which includes 28% of networks, while the rest of the networks with higher loss are considered as large-error (*red*). We illustrate different outputs and error values in Fig. 4C-right. SMNIST accuracy appears to be favorable for smaller network sizes, conversely to CharRNN task. LSTM variants of larger size (128 and 256) do not achieve low-errors while LSTMs of 32 units appear to have the smallest number of high-error variants. Furthermore, the distribution of errors appears to be in the full range of the parameter p indicating the importance of selecting both hyperparameters for optimal accuracy.

As in previous tests, AeLLE analysis is able to unravel variants and their accuracy according to LE spectrum. AeLLE plane clusters the spectra of variants with low-error into a group in a form of a dense arc in the mid-left part of the plane, see Fig. 4C-left. Such separation clearly distinguishes between the two groups of accuracies and allows easy detection of the high-accuracy group by fitting an ellipse which contains the arc.

Pre-Trained AeLLE for Accuracy Prediction

In the three tests described above, we find that the same general approach of AeLLE allows selection of variants of hyperparameters of RNN associated with accuracy. LE spectrum is computed for fully trained models to set apart the sole role of hyperparameter variation. Namely, all variants in these benchmarks have been trained prior to computing LE spectrum. Over the course of training, connectivity weight parameters vary and as a result LE spectrum undergoes deformations. However, it appears that the general properties of LE spectrum such as the overall shape emerge early in training.

From these findings and the success of AeLLE, a natural question arises: how early in training can AeLLE identify networks that will perform well upon completion of training? To investigate this question we use a pre-trained AeLLE classifier, i.e., trained on a subset of variants that were fully trained for the task. We then propose to test how such pre-trained fixed AeLLE represents variants that are only partially trained, e.g., underwent $0\% - 50\%$ of training. This test is expected to show how robust are the inferred features within AeLLE correlating hyperparameters and LE spectrum subject to optimization of connectivity weights. Also it would provide insight into how long it is necessary to train the network to predict the accuracy of a hyperparameter variant.

We select the SMNIST task with LSTM models with 64 hidden states size and initialization parameter p between 2.0 and 2.5 (200 variants with similar number of low- and large-accuracy models) as the test for the study. We then compute LE spectrum for first ten epochs of the training (out of all 20 epoch) for all variants. We then select 64 variants into a training set and train AeLLE on LE spectrum curves for all epochs of the training set. We define such AeLLE as a Pre-Trained AeLLE and investigate its performance.

We select the validation set (16 variants), and apply the same loss threshold of 0.5 as above. Therefore, 56.3% of these variants are low-error networks and 43.7% are large-error networks. We use this set to define a simple threshold that classifies low- and large-error variants according to AeLLE (Fig. 5 green shaded region of $PC1 > 5$). We then apply the same Pre-Trained AeLLE and accuracy threshold to variants in the test set (120 variants) at different epochs (formulated as % of Training) illustrated in Fig. 5 with training progressing from left to right and Table 1. We observe that projection of variants LE spectrum onto AeLLE (points in AeLLE 2D embedding) changes over the course of training. Before training, the embedding is dense with none of the embedded LE spectra is positioned in the low-error region. In the course of training, the embedded points separate from each other such that points corresponding to low-error move to the right and points of large-error remain to the left and outside of the threshold region. Specifically, after just a single epoch, 5% of training, we find that the

Training	**AeLLE** vs. [Loss]			
%	Recall		Precision	F1
0%	0%	[0%]	–	–
5%	**35.9%**	[21.4%]	**95.8%** [100%]	**0.52** [0.35]
20%	**67.2%**	[49.0%]	**91.5%** [100%]	**0.78** [0.66]
35%	**82.0%**	[62.6%]	**79.2%** [100%]	**0.81** [0.77]
50%	**89.1%**	[70.4%]	**75.0%** [99%]	**0.81** [0.82]
100%	**96.9%**		**62.0%**	**0.76**

Table 1 Precision, Recall, and F1 Score of pre-trained AeLLE classifier for RNN final accuracy evaluated at different stages of training compared with a direct classifier based on loss values.

Recall, i.e., the number of the low-error networks which fall within the low-error threshold region, is 35.9%. The Precision, i.e., how many networks in the region are low-error, is 95.8%, reflecting that 23 of 24 variants are correctly identified as low-error. After 4 epochs of training (20%), the Recall becomes 67.2% such that the region contains more low-error networks, and Precision of 91.5%. As training proceeds to 35% and 50%, we observe that the Recall rate is further enhanced to 82% and 89.1% respectively. This can be visually observed with almost all green points being included in the green threshold region in Fig. 5-(50% Task Training). With improvement of the Recall, we also observe that the Precision slightly decreases to 79% and 75% respectively, indicating that several large-error LE spectrum (red points) are erroneously classified as low-error. As a result, we observe that F1 score settles at approximately 0.81 value.We contrast these results by computing the Recall and the Precision rates at the completion training (100%). We find that in such a case the Recall would increase to approximately 96.9%, while Precision would drop to 62% with F1 score of 0.76. Such F1 score is lower than the score for 35% and 50% which indicates that in terms of optimizing the F1 score an optimum would be reached much earlier in training. Notably, the classifier that we have chosen is the simplest classifier (threshold of the dominant PC mode) to establish a low bound on the the effectiveness of AeLLE during training. We expect improved classification rates for more precise classifiers.

In summary, we find that Pre-Trained AeLLE converges early in training to an effective classifier that predicts the accuracy of the given RNN when fully trained. To further quantify the effectiveness of AeLLE classifier prediction, we compare it with a direct feature of training, the loss value at each stage of training, see Table 1. We find

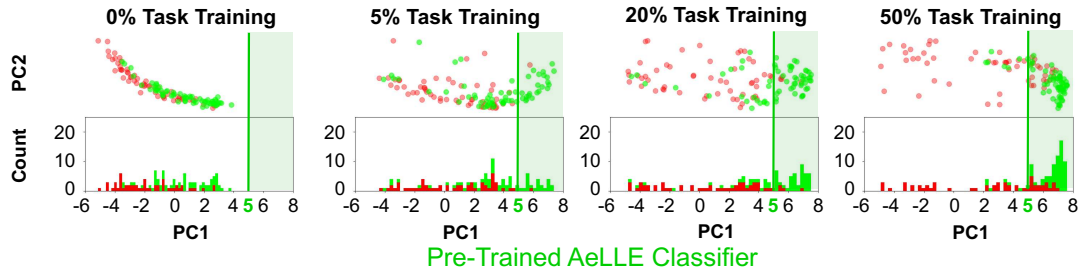

Fig. 5 Pre-Trained AeLLE during RNN training. Prediction of task accuracy of inputs in a test set by pre-trained AeLLE threshold classifier ($PC1 > 5$) of SMNIST LSTM variants is shown during $(0\%, 5\%, 20\%, 50\%)$ RNN training. Before the RNN is trained there are no low error variants (green; task accuracy of < 0.5) crossing the threshold. As training progresses, low error variants cross the threshold, such that at 50% of training almost all low error variants have crossed the threshold (high Recall). At 20% of training many low error variants crossed the threshold and almost all large error variants (red) remained outside of the threshold (high Precision).

that variants with low loss early in training do correspond to variants that will be classified as low-error, indicated by almost perfect Precision rate of $99\%-100\%$. However, it appears that many variants of high accuracy do not converge quickly. Indeed, the Recall rate for a classifier based on loss values is $21\%-70\%$ for $5\%-50\%$ of training, while in contrast, AeLLE Recall rate is $35\% - 89.1\%$ during the same training procedure. This indicates that classification based only on loss could miss multiple variants that may turn out to be accurate.

Discussion

While Recurrent Neural Networks (RNN) are ubiquitous in their applicability to multi-dimensional input sequences and associated tasks, the relation of RNN configurations to accuracy is still lacking. Indeed, RNN can be viewed as nonlinear high-dimensional non-autonomous dynamical systems with varying parameters, and as such, are complex systems that require advanced appropriate methods for analysis. In this work, we studied whether the methodology of Lyapunov exponents (LE), designed for nonlinear complex systems, can assist in inference of the relation between RNN hyperparameters and accuracy. While the direct approach of Lyapunov analysis and its standard features cannot be correlated with accuracy, we show that there exists a learnable relationship through an auxiliary Autoencoder for LE spectrum. The proposed Autoencoder infers the features of LE spectrum that are most related to accuracy and hence serves as a link between RNN architectural configurations and accuracy on a given task.

LE methodology is an effective toolset to study nonlinear dynamical systems since LE measure the divergence of nearby trajectories, thus indicate the degree of stability and chaos in that system. Indeed, LE has been applied to various dynamical systems and applications, and there exist theoretical underpinning for characterization of these systems by LE spectrum. However, RNN differ from typical systems by being non-autonomous and possessing multiple hyperparameters that provide a configuration for network architecture. Furthermore, RNN undergo training, in which optimization of loss varies the connectivity parameters. These aspects make the information contained in LE tangled such that it is unknown whether under such alterations LE spectrum remains descriptive of the underlying RNN. Our results demonstrate that the information that LE contain regarding the dynamics of a network could be related to network accuracy on various tasks. In particular, we show that AeLLE Latent representation can generalize across inputs such that AeLLE trained on LE spectra of RNN with inputs belonging to the validation set, can generalize and provide an effective representation for a separate set of inputs, i.e., the test set. Furthermore, we also show that AeLLE can generalize for connectivity weights training such that pre-trained AeLLE can capture the unique features that correspond to accuracy early-on in training.

Due to these generalization properties, we conclude that AeLLE with an appropriate classifier can serve as a predictive and diagnostic tool

for hyperparameter selection early in training. Specifically, the projection of the LE spectrum of a network onto Pre-Trained AeLLE can predict whether a network will eventually achieve low or large error before completing training. With a simple classifier, being a threshold in the dominant PC mode, we observe that prediction is effective as early as 35% into training. With a more precise classifier, we expect this prediction to happen even sooner. From practical standpoint, this property allows a quick selection of sets of hyperparameters for the network during hyperparameter optimization. Notably, AeLLE is unable to predict the performance of networks which have not been subject to training at all. This is expected as the statistics of the input and the task are not yet captured in the parameters of the RNN and, therefore, in the LE.

While our work focuses on selection of RNN hyperparameters with high-accuracy, the AeLLE methodology is general and could be utilized to other systems and extensions. For example, the Autoencoder and its Latent representation can be utilized to organize and classify architectural variants of RNN for particular tasks and could be targeted toward inference of features that define each architecture. Furthermore, extension of AeLLE methodology can be used to search and unravel novel architectures. AeLLE approach can be also adopted to analyze other complex dynamical systems. For example, long-term forecasting of temporal signals from dynamical systems is a challenging problem that has been addressed with a similar data-driven approach using autoencoders or spectral methods [74, 75]. Application of AeLLE could unify such approaches for dynamical systems representing various physical systems. The key building blocks in AeLLE that would need to be established for each of these extensions is efficient computation of LE exponents and sufficient sampling of data to train the Autoencoder to form an informative Latent representation.

Code and Data Availability

The datasets analysed during the current study as well as the code used to generated results are available in the LyapunovAutoEncode repository, https://github.com/shlizee/LyapunovAutoEncode.

References

[1] Pang, B., Zha, K., Cao, H., Shi, C., Lu, C.: Deep rnn framework for visual sequential applications. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

[2] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association (2010)

[3] Das, S., Olurotimi, O.: Noisy recurrent neural networks: the continuous-time case. IEEE Transactions on Neural Networks **9**(5), 913–936 (1998). https://doi.org/10.1109/72.712164

[4] Tino, P., Schittenkopf, C., Dorffner, G.: Financial volatility trading using recurrent neural networks. IEEE transactions on neural networks **12**(4), 865–874 (2001)

[5] Su, K., Liu, X., Shlizerman, E.: PREDICT & CLUSTER: Unsupervised Skeleton Based Action Recognition (2019)

[6] Pennington, J., Schoenholz, S., Ganguli, S.: Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In: Advances in Neural Information Processing Systems, pp. 4785–4795 (2017)

[7] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)

[8] Gregor, K., Danihelka, I., Graves, A., Rezende, D., Wierstra, D.: Draw: A recurrent neural network for image generation. In: International Conference on Machine Learning, pp. 1462–1471 (2015). PMLR

[9] Choi, K., Fazekas, G., Sandler, M.: Text-based LSTM networks for automatic music composition (2016)

[10] Mao, H.H., Shin, T., Cottrell, G.: Deepj:

Style-specific music generation. In: 2018 IEEE 12th International Conference on Semantic Computing (ICSC), pp. 377–382 (2018). IEEE

[11] Guo, D., Zhou, W., Li, H., Wang, M.: Hierarchical lstm for sign language translation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)

[12] Elman, J.L.: Finding structure in time. Cognitive science **14**(2), 179–211 (1990)

[13] Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep Learning vol. 1. MIT press Cambridge, ??? (2016)

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

[15] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[16] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR **abs/1406.1078** (2014) https://arxiv.org/abs/1406.1078

[17] Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR **abs/1412.3555** (2014) https://arxiv.org/abs/1412.3555

[18] Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging (2015)

[19] Santhanam, S.: Context based text-generation using lstm networks (2020)

[20] Liu, Q., Zhou, F., Hang, R., Yuan, X.: Bidirectional-convolutional lstm based spectral-spatial feature learning for hyperspectral image classification. Remote Sensing **9**(12), 1330 (2017)

[21] Ghosh, S., Vinyals, O., Strope, B., Roy, S., Dean, T., Heck, L.: Contextual lstm (clstm) models for large scale nlp tasks (2016)

[22] Akilan, T., Wu, Q.J., Safaei, A., Huo, J., Yang, Y.: A 3d cnn-lstm-based image-to-image foreground segmentation. IEEE Transactions on Intelligent Transportation Systems **21**(3), 959–971 (2019)

[23] Wang, C., Yang, H., Bartz, C., Meinel, C.: Image captioning with deep bidirectional lstms. In: Proceedings of the 24th ACM International Conference on Multimedia, pp. 988–997 (2016)

[24] Byeon, W., Breuel, T.M., Raue, F., Liwicki, M.: Scene labeling with lstm recurrent neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3547–3555 (2015)

[25] Zhou, F., Hang, R., Liu, Q., Yuan, X.: Hyperspectral image classification using spectral-spatial lstms. Neurocomputing **328**, 39–47 (2019)

[26] Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp. 1310–1318 (2013)

[27] Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., Ganguli, S.: Exponential expressivity in deep neural networks through transient chaos. In: Advances in Neural Information Processing Systems, pp. 3360–3368 (2016)

[28] Wang, B., Hoai, M.: Predicting body movement and recognizing actions: an integrated framework for mutual benefits. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), pp. 341–348 (2018). IEEE

[29] Chen, M., Pennington, J., Samuel: Gating Enables Signal Propagation in Recurrent Neural Networks. In: ICML (2018)

[30] Yang, G.: Scaling limits of wide neural networks with weight sharing: Gaussian process

behavior, gradient independence, and neural tangent kernel derivation (2019)

[31] Zheng, Y., Shlizerman, E.: R-FORCE: Robust Learning for Random Recurrent Neural Networks (2020)

[32] Wisdom, S., Powers, T., Hershey, J., Le Roux, J., Atlas, L.: Full-capacity unitary recurrent neural networks. Advances in neural information processing systems **29**, 4880–4888 (2016)

[33] Jing, L., Shen, Y., Dubcek, T., Peurifoy, J., Skirlo, S., LeCun, Y., Tegmark, M., Soljačić, M.: Tunable efficient unitary neural networks (eunn) and their application to rnns. In: International Conference on Machine Learning, pp. 1733–1741 (2017). PMLR

[34] Mhammedi, Z., Hellicar, A., Rahman, A., Bailey, J.: Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In: International Conference on Machine Learning, pp. 2401–2409 (2017). PMLR

[35] Vorontsov, E., Trabelsi, C., Kadoury, S., Pal, C.: On orthogonality and learning recurrent networks with long term dependencies. In: International Conference on Machine Learning, pp. 3570–3578 (2017). PMLR

[36] Azencot, O., Erichson, N.B., Ben-Chen, M., Mahoney, M.W.: A Differential Geometry Perspective on Orthogonal Recurrent Models (2021)

[37] Chang, B., Chen, M., Haber, E., Chi, E.H.: AntisymmetricRNN: a dynamical system view on recurrent neural networks. In: International Conference on Learning Representations (2019). https://openreview.net/forum?id=ryxepo0cFX

[38] Kerg, G., Goyette, K., Puelma Touzel, M., Gidel, G., Vorontsov, E., Bengio, Y., Lajoie, G.: Non-normal recurrent neural network (nnrnn): learning long time dependencies while improving expressivity with transient dynamics. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 13613–13623. Curran Associates, Inc., ??? (2019). http://papers.nips.cc/paper/9513-non-normal-recurrent-neural-network-nnrnn-learn pdf

[39] Rusch, T.K., Mishra, S.: Coupled oscillatory recurrent neural network (cornn): An accurate and (gradient) stable architecture for learning long time dependencies. In: International Conference on Learning Representations (2020)

[40] Erichson, N.B., Azencot, O., Queiruga, A., Hodgkinson, L., Mahoney, M.W.: Lipschitz recurrent neural networks (2020)

[41] Lim, S.H., Erichson, N.B., Hodgkinson, L., Mahoney, M.W.: Noisy recurrent neural networks (2021)

[42] Maheswaranathan, N., Williams, A., Golub, M., Ganguli, S., Sussillo, D.: Universality and individuality in neural dynamics across large populations of recurrent networks. In: Wallach, H., Larochelle, H., Beygelzimer, A., dAlche-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 15629–15641. Curran Associates, Inc., ??? (2019). http://papers.nips.cc/paper/9694-universality-and-individuality-in-neural-dynamics-pdf

[43] Ruelle, D.: Ergodic theory of differentiable dynamical systems. Publications Mathématiques de l'Institut des Hautes Études Scientifiques **50**(1), 27–58 (1979)

[44] Oseledets, V.: Oseledets theorem. Scholarpedia **3**(1), 1846 (2008). https://doi.org/10.4249/scholarpedia.1846. revision #142085

[45] Saitô, N., Ichimura, A.: Ergodic components in the stochastic region in a hamiltonian system. In: Casati, G., Ford, J. (eds.) Stochastic Behavior in Classical and Quantum Hamiltonian Systems, pp. 137–144. Springer, Berlin, Heidelberg (1979)

[46] Ochs, G.: Stability of oseledets spaces is

equivalent to stability of lyapunov exponents. Dynamics and Stability of Systems **14**(2), 183–201 (1999)

[47] Geist, K., Parlitz, U., Lauterborn, W.: Comparison of Different Methods for Computing Lyapunov Exponents. Progress of Theoretical Physics **83**(5), 875–893 (1990) https://arxiv.org/abs/https://academic.oup.com/ptp/article-pdf/83/5/875/5302061/83-5-875.pdf. https://doi.org/10.1143/PTP.83.875

[48] Engelken, R., Wolf, F., Abbott, L.: Lyapunov spectra of chaotic recurrent neural networks (2020)

[49] platform, O.: The ordinal numbers of systems of linear differential equations. mathematical journal **31**(1), 748–766 (1930)

[50] Dawson, S., Grebogi, C., Sauer, T., Yorke, J.A.: Obstructions to shadowing when a lyapunov exponent fluctuates about zero. Phys. Rev. Lett. **73**, 1927–1930 (1994). https://doi.org/10.1103/PhysRevLett.73.1927

[51] Abarbanel, H.D.I., Brown, R., Kennel, M.B.: Variation of lyapunov exponents on a strange attractor. Journal of Nonlinear Science **1**(2), 175–199 (1991). https://doi.org/10.1007/BF01209065

[52] Shibata, H.: Ks entropy and mean lyapunov exponent for coupled map lattices. Physica A: Statistical Mechanics and its Applications **292**(1), 182–192 (2001). https://doi.org/10.1016/S0378-4371(00)00591-4

[53] Brandstäter, A., Swift, J., Swinney, H.L., Wolf, A., Farmer, J.D., Jen, E., Crutchfield, P.J.: Low-dimensional chaos in a hydrodynamic system. Phys. Rev. Lett. **51**, 1442–1445 (1983). https://doi.org/10.1103/PhysRevLett.51.1442

[54] Yamada, M., Ohkitani, K.: The Inertial Subrange and Non-Positive Lyapunov Exponents in Fully-Developed Turbulence. Progress of Theoretical Physics **79**(6), 1265–1268 (1988) https://arxiv.org/abs/http://oup.prod.sis.lan/ptp/article-pdf/79/6/1265/5295968/79-6-1265.pdf. https://doi.org/10.1143/PTP.79.1265

[55] Vogt, R., Puelma Touzel, M., Shlizerman, E., Lajoie, G.: On lyapunov exponents for rnns: Understanding information propagation using dynamical systems tools. Frontiers in Applied Mathematics and Statistics **8** (2022). https://doi.org/10.3389/fams.2022.818799

[56] Arnold, L.: Random Dynamical Systems. Springer, ??? (1991). https://doi.org/10.1007/978-3-662-12878-7. http://arxiv.org/abs/math/0608162

[57] Benettin, G., Galgani, L., Giorgilli, A., Strelcyn, J.-M.: Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 1: Theory. Meccanica **15**(1), 9–20 (1980)

[58] Dieci, L., Van Vleck, E.S.: Computation of a few Lyapunov exponents for continuous and discrete dynamical systems. Applied Numerical Mathematics **17**(3), 275–291 (1995)

[59] Meng, Q., Catchpoole, D., Skillicom, D., Kennedy, P.J.: Relational autoencoder for feature extraction. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 364–371 (2017). IEEE

[60] Zabalza, J., Ren, J., Zheng, J., Zhao, H., Qing, C., Yang, Z., Du, P., Marshall, S.: Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. Neurocomputing **185**, 1–10 (2016)

[61] Hou, X., Shen, L., Sun, K., Qiu, G.: Deep feature consistent variational autoencoder. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1133–1141 (2017). IEEE

[62] Lv, N., Chen, C., Qiu, T., Sangaiah, A.K.: Deep learning and superpixel feature extraction based on contractive autoencoder for change detection in sar images. IEEE transactions on industrial informatics **14**(12), 5530–5538 (2018)

[63] Liu, H., Taniguchi, T.: Feature extraction and pattern recognition for human motion by a deep sparse autoencoder. In: 2014 IEEE International Conference on Computer and Information Technology, pp. 173–181 (2014). IEEE

[64] Chollet, F.: Deep Learning with Python. Simon and Schuster, ??? (2017)

[65] Pearson, K.: Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin philosophical magazine and journal of science **2**(11), 559–572 (1901)

[66] Su, K., Shlizerman, E.: Clustering and recognition of spatiotemporal features through interpretable embedding of sequence to sequence recurrent neural networks. Frontiers in artificial intelligence **3**, 70 (2020)

[67] Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)

[68] McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction (2018)

[69] Sussillo, D., Abbott, L.F.: Generating coherent patterns of activity from chaotic neural networks. Neuron **63**(4), 544–557 (2009)

[70] DePasquale, B., Cueva, C.J., Rajan, K., Escola, G.S., Abbott, L.: full-force: A target-based method for training recurrent networks. PloS one **13**(2), 0191527 (2018)

[71] Karpathy, A., Johnson, J., Fei-Fei, L.: Visualizing and understanding recurrent networks (2015)

[72] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[73] Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling (2018)

[74] Lusch, B., Kutz, J.N., Brunton, S.L.: Deep learning for universal linear embeddings of nonlinear dynamics. Nature Communications **9**(1), 4950 (2018). https://doi.org/10.1038/s41467-018-07210-0

[75] Lange, H., Brunton, S.L., Kutz, J.N.: From fourier to koopman: Spectral methods for long-term time series prediction. J. Mach. Learn. Res. **22**(41), 1–38 (2021)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- AELLELyaponuvGuidedEmbeddingSM.pdf