Neural Computing and Applications (2024) 36:21211–21226
https://doi.org/10.1007/s00521-024-09824-6

ORIGINAL ARTICLE

Lyapunov-guided representation of recurrent neural network performance

Ryan Vogt[1] · Yang Zheng[2] · Eli Shlizerman[1,2]

Received: 20 January 2024 / Accepted: 12 April 2024 / Published online: 2 July 2024

Abstract

Recurrent neural networks (RNN) are ubiquitous computing systems for sequences and multivariate time-series data. While several robust RNN architectures 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 treating RNN as dynamical systems and correlating hyperparameters with accuracy through Lyapunov spectral analysis, a methodology designed explicitly 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 predicting the accuracy to which RNN would converge when training is complete. We conclude that the representation of RNN through the Lyapunov spectrum, along with AeLLE, provides a novel method for the organization and interpretation of variants of RNN architectures.

Keywords Recurrent neural Networks · Hyperparameter optimization · Networked dynamical systems · Lyapunov exponents · Dynamical systems for deep learning

1 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 applications, including 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 they incorporate, and the learning process they undergo. As a result, it is not straightforward to associate an optimally performing RNN with a particular dataset and task. The difficulty stems from RNN characteristics, making them intricate dynamic systems. In particular, RNN can be classified as (i) nonlinear, (ii) high-dimensional, and (iii) non-autonomous dynamical systems with (iv) varying parameters, which are either global (hyperparameters) or trainable weights (connectivity).

Architectural variants of RNN that are more optimal and robust than unstructured RNN have been introduced, ranging from the long-established gated RNN, such as long short-term memory (LSTM) [12] and gated recurrent units (GRU) [13, 14], to more recent networks, such as anti-symmetric RNN (ASRNN) [15], orthogonal RNN (ORNN) [16], coupled oscillatory RNN (CoRNN) [17], Lipschitz RNN [18], and more. While such specific architectures are robust, they represent singular points in the space of RNN. In addition, the accuracy of these architectures still

✉ Eli Shlizerman
shlizee@uw.edu

Ryan Vogt
ravogt95@uw.edu

Yang Zheng
zheng94@uw.edu

[1] Department of Applied Mathematics, University of Washington, Seattle, WA 98195, USA

[2] Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195, USA

depends on hyperparameters, among which are the initial state of the network, initialization of the network connectivity weights, optimization settings, architectural parameters, and 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, leading to potentially unpredictable behavior.

A powerful dynamical system method for characterization and predictability of dynamical systems is *Lyapunov exponents (LE)* [19, 20]. Recent work identifying RNN as dynamical systems has extended LE calculation and analysis to these systems [21], but the connection between LE and network performance has not been explored extensively. Examples in [22] suggest that features of LE spectrum are correlated with RNN robustness and accuracy. However, standard features such as maximum and mean LE can have weak correlations, making consistent characterization of network quality using a fixed set of LE features unfeasible.

Such inconsistency motivates this 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 (see Fig. 1A). The Latent representation appears to be low dimensional such that even a simple linear embedding of the representation, denoted as AeLLE, corresponds to a classifier for selecting the optimally performing RNN based on the LE spectrum (see Fig. 1B). We show that once AeLLE is trained, it holds for novel inputs, and we also investigate the correlation between AeLLE classification accuracy and the need for RNN training.

The significance of the proposed AeLLE method is that it is a novel LE embedding that effectively facilitates interpretation and classification for a variety of RNN models. For example, we show that AeLLE separates high- and low-accuracy networks across varying weight initialization hyperparameters, network size, and network architecture with no knowledge about the underlying network besides its Lyapunov spectrum. Notably, such separation is not observed when AeLLE is omitted, and the Lyapunov spectrum is directly used in conjunction with standard embeddings. Indeed, our results indicate that AeLLE representation is a necessary step that follows the LE spectrum. While it requires additional computational power and time due to the training of the Autoencoder, it is a necessary computational step since it is able to identify the subtle features of the Lyapunov spectrum. As a result, AeLLE implicitly identifies the network dynamics, which are integral to performance across a wide variety of network hyperparameters. Furthermore, we show that AeLLE can be used to predict final network accuracy early in training. This predictability suggests that during training, AeLLE recovers the Lyapunov spectrum features, which are instrumental for trainability. These features are not evident in standard LE statistics, as discussed in [22–24].

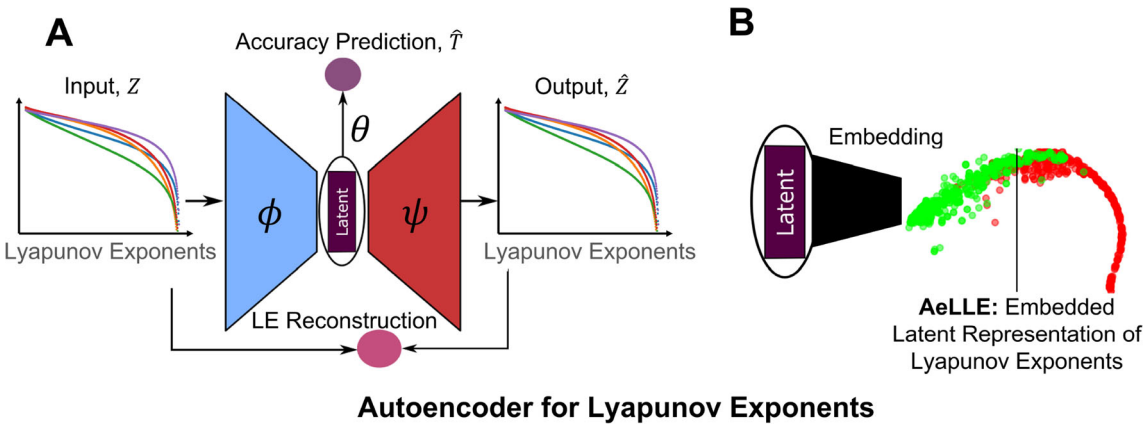

Fig. 1 AeLLE: LE spectrum Autoencoder and Latent representation embedding. (**A**) The Autoencoder takes Lyapunov exponents as input. This input is then embedded into a Latent space (purple) by the encoder (blue). From this Latent space, the Autoencoder predicts the accuracy of the corresponding network, which is compared to the true accuracy of the network to get the prediction loss. Simultaneously, the Lyapunov spectrum is reconstructed from this Latent space by the decoder (red). These reconstructed Lyapunov exponents are then compared to the input Lyapunov exponents to get the reconstruction loss. These sums are added together with normalization factor α to get the total 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, leading to separation between high-accuracy (green) and low-accuracy (red) networks. 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. Shown is an example linear classifier along the first PC dimension for networks trained on the CharRNN task (see Sect. "4" for more details)

2 Related work

2.1 Spectral analysis and model quality

Spectral methods have been used to characterize information propagation properties and, thus, model the quality of RNNs. 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 [25–28]. For architectures used in practice, these approaches appear to have a limited scope [29, 30]. 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, which is the reason for proposing to use Lyapunov exponents in this work for such characterization.

Other techniques for inferring model quality include performing spectral analysis of deep neural network weights and fitting their distribution to truncated power laws to correlate with performance [31]. Another approach uses Koopman operators to linearize RNNs demonstrates that linear representations of RNNs capture the dominant modes of the networks and are able to achieve comparable performance to their nonlinear counterparts [32].

2.2 Dynamical systems approaches to RNNs

Identifying RNN as dynamical systems and 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 [15, 16, 33–37] and additional architectures such as coupled oscillatory RNN [17] and Lipschitz RNN [18]. These approaches set network weights to form dynamical systems that 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 [38]. Furthermore, a universality analysis of fixed points of different RNN, proposed in [39], hints that RNN architectures could be organized in similarity classes such that, despite having different architectural properties, they 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 the development of dynamical systems tools that *characterize and classify RNN variants*.

2.3 Lyapunov spectrum

Lyapunov exponents [19, 20] capture the information generation by a system's dynamics through measurement of the separation rate of infinitesimally close trajectories. The number of Lyapunov exponents of a system is equal to the dimension of that system, and the collection of all LE is called LE spectrum. The maximal LE will determine the linear stability of the system [40]. Further, a system with 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. The dynamics converge toward a fixed point attractor when all exponents are negative. Zero exponents represent limit cycles or quasiperiodic orbits [41, 42]. 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 [43]. The LE variance measures heterogeneity in stability across different directions and can reflect the conditioning of the product of many Jacobians.

When LE are computed from the 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) [44, 45]. Efficient approaches and algorithms have been developed for computing LE spectrum [46]. These have been applied to various dynamical systems, including hidden states of RNN and variants such as LSTM and GRU [21]. 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 [47]. It has been demonstrated that some features of LE spectra can have meaningful correlations with the performance of the corresponding networks on a given task. However, the selection of relevant features of it is task-dependent and challenging to determine a prior [22].

3 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.

3.1 Computation of LE [21, 22]

We compute LE by adopting the well-established algorithm [48, 49] and follow the implementation in [21, 22]. 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 $\mathbf{Q}_0$ is initialized as the identity to represent an orthogonal set of nearby initial states whose evolution will be tracked in the sequence of matrices $\mathbf{Q}_t$. The hidden states h_t are initialized as zeros.

To track the expansion and contraction of the vectors of $\mathbf{Q}_t$, the Jacobian of the hidden states at step t, $\mathbf{J}_t$, is calculated and then applied to the vectors of $\mathbf{Q}_t$. 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}

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

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

$$\mathbf{Q}_{t+1}, \mathbf{R}_{t+1} = QR(\mathbf{J}_t \mathbf{Q}_t). \tag{2}$$

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

$$\lambda_k = \frac{1}{T}\sum_{t=1}^{T} \log(r_t^k) \tag{3}$$

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 are calculated over a fixed number of time steps with n 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 to retain the shape of the largest network size. Through this interpolation, we can represent the LE spectra as curves. Spectra curves will have the same number of LE points for small and larger networks.

The expressions for the Jacobians $\mathbf{J}_t$ used in these calculations can be found in the Supplemental Materials.

3.2 Autoencoder for LE spectrum

An Autoencoder consists of two components: an *encoder* network ϕ, which transforms the input into a representation in the Latent layer, and a *decoder* network ψ, which transforms the Latent representation into a reconstruction of the original input.

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 formulating 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 [50, 51].

We propose an adapted Autoencoder (Ae) methodology for correlating LE spectra and RNN task accuracy. In this setup, we consider the LE spectrum as the input $\boldsymbol{Z}$. Our Autoencoder consists of a fully-connected encoder network ϕ, a fully-connected decoder network ψ, and an additional linear prediction network ξ defined by

$$\begin{aligned} \hat{Z} &= (\psi \circ \phi)Z, \\ \hat{T} &= (\xi \circ \phi)Z, \end{aligned} \tag{4}$$

where $\hat{\mathbf{Z}}$ and $\hat{\mathbf{T}}$ correspond to the output from the decoder and predicted accuracy, respectively, with loss of $L = \|Z - \hat{Z}\|^2 + \alpha \cdot \|T - \hat{T}\|_l$. Ae performs the reconstruction task, optimization of the first term of Eq. 5, 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. 5.

$$\phi, \psi, \xi = \arg\min_{\phi,\psi,\xi}(\|Z - \hat{Z}\|^2 + \alpha \cdot \|T - \hat{T}\|_l). \tag{5}$$

The parameter l can defined based on the desired behavior. The most common choice is $l = 1$, indicating the 1-norm, and $l = 2$, indicating the 2-norm.

During the training of Ae, the weight α in the prediction loss gradually increases so that Ae emphasizes RNN error prediction once the reconstruction error has converged. We found that this approach allows Ae to capture features of both RNN dynamics and accuracy. A choice of α being too small leads to a 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, diverges. The convergence of Ae for different RNN variants, as we demonstrate in the 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.

3.3 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 [52, 53]. The embedding consists of performing principal component analysis and projecting the representation on the first few principal component directions (e.g., 2 or 3). While other nonlinear embeddings are possible, e.g., tSNE or UMAP [54, 55], the simple linear projection onto the first two principal components of the Latent space results in an effective organization. This indicates that the Latent representation has successfully captured the characterizing features of performance. In the Results section, we show that the PCA embedding is sufficient to provide an effective space for all examples of RNN architectures and tasks that we considered. In particular, in this space, the more accurate RNN variants (green) can be separated from other variants (red) through a simple clustering procedure.

4 Results

To investigate the applications 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 grayscale values. Nonetheless, across this wide variety of inputs and tasks, AeLLE space and clustering can consistently separate variants of hyperparameters according to accuracy in a more informative way than network hyperparameters alone.

More specifically, we consider *(i) Signal Reconstruction* task, also known as target learning. In this task, a random RNN is being tracked to generate a target output signal from a random input [30, 56]. This task involves intricate time-dependent signals and a generic RNN whose dynamics are chaotic in the absence of training. With this example, we demonstrate that our method is able to distinguish between networks of high and low accuracy across *initialization parameters*.

(ii) Character Prediction is a common task that takes as an input a sequence of textual characters and outputs the next character of the sequence. This task is rather simple and is used to benchmark various RNN variants. With this task, we demonstrate that our method is able to distinguish across *network sizes*, in addition to initialization parameters.

(iii) Sequential MNIST is a more extensive benchmark for RNN classification accuracy. The task input is an image of a handwritten digit unrolled into a sequence of numerical values (pixels' grayscale values), and the output is a corresponding label of the digit. We investigate the accuracy of various RNN variants on row-wise SMNIST, demonstrating that our method distinguishes according to performance across *network architectures*. We describe the outcomes of AeLLE application and the resulting insights per each task below.

For each task, we present the low-dimensional projection of AeLLE using the first two principal components. Furthermore, we reduce the projection to a single dimension by showing the distribution of the AeLLE in the first principal component as stacked histograms of the high- and low-accuracy networks or the different hyperparameters. For these stacked histograms, we represent each distribution separately and stack the histograms on top of each other, such that the bar heights are discrete and not cumulative. Therefore, the ordering of the colors will always be the same, and the height of each bar indicates the number of networks in that bin, regardless of whether it is above or below the other colors.

4.1 Signal reconstruction via 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 target learning task. We set the target signal (output) to be a four-sine wave, a benchmark used in [56]. A key parameter in Rank-1 RNN is the amplification factor of the connectivity g, which controls the output signal in the absence of

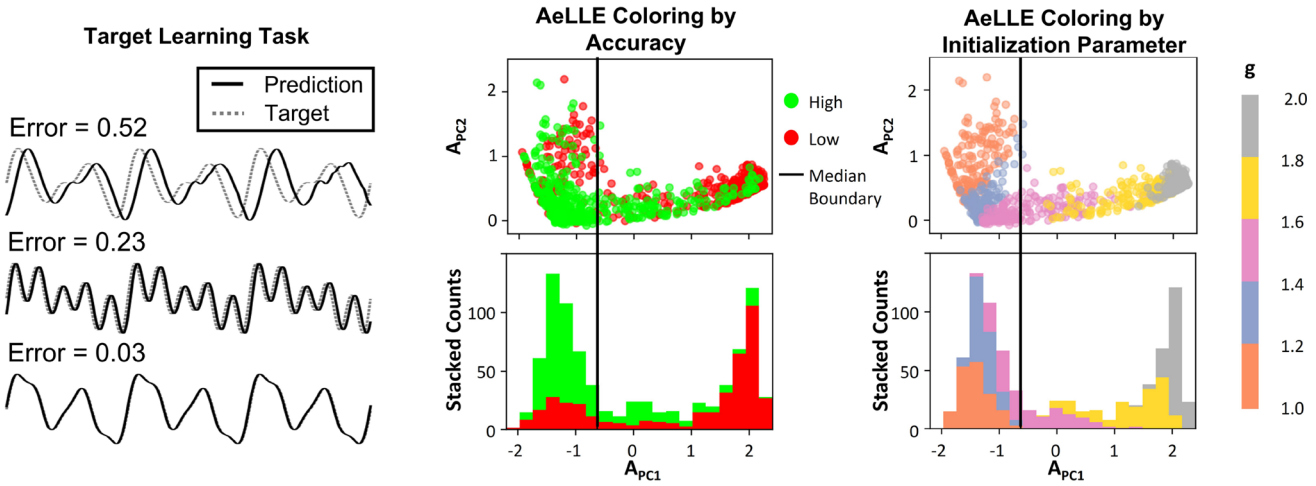

Fig. 2 Clustering by performance across initialization parameter: AeLLE for networks trained on signal reconstruction task. *Left:* The signal reconstruction task involves recreating a target signal. Higher losses indicate a greater pointwise difference between the target and predicted signals. *Center:* In the AeLLE representation, most of the high-accuracy networks (green) cluster to the left of most of the low-accuracy networks (red). The black vertical line indicates the location of the median classifier along the first principal component of the AeLLE space. Moreover, the greatest concentration of the high-accuracy networks is in the bottom-left of the space shown, which is consistent with the stacked histogram indicating that there are relatively few high-accuracy networks to the right of the median compared to low accuracy, and many more to the left. *Right:* In comparison, the cluster to the bottom-left of the space shown contains a mixture of networks with initialization parameters ranging from 1.0 to 1.6. In general, networks with larger initialization parameter g (particularly once $g > 1.6$) tend to cluster further to the right in this space. This is consistent with the fact that networks with $g > 1.6$ tend to have lower accuracy on this task, but networks with $1 < g < 1.6$ tend to have similarly high accuracy (see Supplemental Materials)

training. For $g \leq 1$, the output signal is zero, while for $g \geq 1.8$, the output signal is significantly chaotic. The output signal is weakly chaotic in the interval $1 < g < 1.8$. Previous work has shown that the network can generate the target signal when it is in the weakly chaotic regime, i.e., $1 < g < 1.8$, and trained with FORCE optimization algorithm. Moreover, it has been shown that this training setup most consistently optimally converges when the amplification factor g is in the interval $1.2 < g < 1.5$ [56, 57].

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. Thereby, the target learning task, Rank-1 RNN architecture, and FORCE optimization are excellent candidates to test whether AeLLE can organize the variants of Rank-1 RNN models according to accuracy.

The candidate hyperparameters for variation would be of (1) samples of fixed connectivity weights (from a normal distribution) and (2) the parameter g within 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, LE in the validation set are projected onto the Autoencoder's AeLLE space, depicted in Fig. 2, where each sample is a dot in the $A_{PC_1} - A_{PC_2}$ plane.

Our results show that AeLLE organizes the variants in a 2D space of $A_{PC_1} - A_{PC_2}$ according to accuracy. The variants with smaller error values (high accuracy) (< 0.57) are colored in *green*, and variants with larger error values (low accuracy) (> 0.57) are colored in *red*. We demonstrate the disparity in the signals that different error values correspond to in Fig. 2—left. AeLLE space succeeds in correlating the LE spectrum with accuracy such that most low-error networks are clustered in the bottom-left of the two-dimensional projection (see Fig. 2—center), whereas large error networks are concentrated primarily in the right and top of the region shown. This clustering of high-accuracy networks allows for identifying multiple candidates as top-performing variants in this space. Comparison of AeLLE clustering with a direct clustering according to g values, Fig. 2—right, shows that while most networks with $g < 1.7$ 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 of the Rank-1 RNN variant being accurate.

Notably, AeLLE contributes to this problem beyond the validation of AeLLE itself. The representation selects variants that are "non-trivial" as well, i.e., it is interesting to examine the particular configurations of models that do not belong to $1.2 < g < 1.5$, but AeLLE still rightfully reports them being successful at target learning or vice versa models in $1.2 < g < 1.5$ but do not reconstruct the target. AeLLE indeed identifies models that are on the tail of the distribution in terms of the parameter g.

For comparison, we calculated the F1 score of the classifier, which uses the median first principal component value as the decision boundary for classifiers that use simple statistics of the Lyapunov spectra. When we use the median value of the Lyapunov spectrum means and maximum Lyapunov exponent as the decision boundaries, the resulting F1 scores are 0.705 and 0.504, respectively, meaning that the mean LE is much more indicative of performance than the maximum LE for this task. Additionally, we define another classifier by projecting the raw Lyapunov exponents onto their first two principal components and using the median of the first principal component as the decision boundary to get an LE PC classifier. For this task, the LE PC classifier performs similarly to the LE mean, with an F1 score of 0.703. Meanwhile, the AeLLE classifier achieves an F1 score of 0.724, indicating a significant improvement over the max LE classifier and a modest improvement over the mean LE and LE PC classifiers (see Supplementary Materials for more details).

4.2 Character prediction with LSTM RNNs of different sizes

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 the English translation of Leo Tolstoy's *War and Peace*, similar to the setup described in [58]. In this setup, each unique character is assigned an index (the number of unique characters in this text is 82). 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 impact the accuracy the most. We create 1200 variants of these parameters, varying the number of hidden units from 64 to 512 (by factors of 2) 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 an Autoencoder training set (80%) and a validation set (20%).

Similar to the target learning task, we project the LE of the variant networks onto the first two PC dimensions of the Latent space of the trained Autoencoder and mark them according to accuracy. LSTM networks with loss below the median among these networks (loss <1.75) are considered high accuracy (*green*), while those with loss above the median are considered low accuracy (*red*).

We find that AeLLE in 2D space separates the spectra of the variants according to accuracy across network sizes. Performing principal component analysis of the AeLLE illustrates that the low- and high-accuracy networks are separated along the PC1 dimension, with higher-accuracy networks being further to the left in the space and lower-accuracy ones clustering to the right (Fig. 3). For

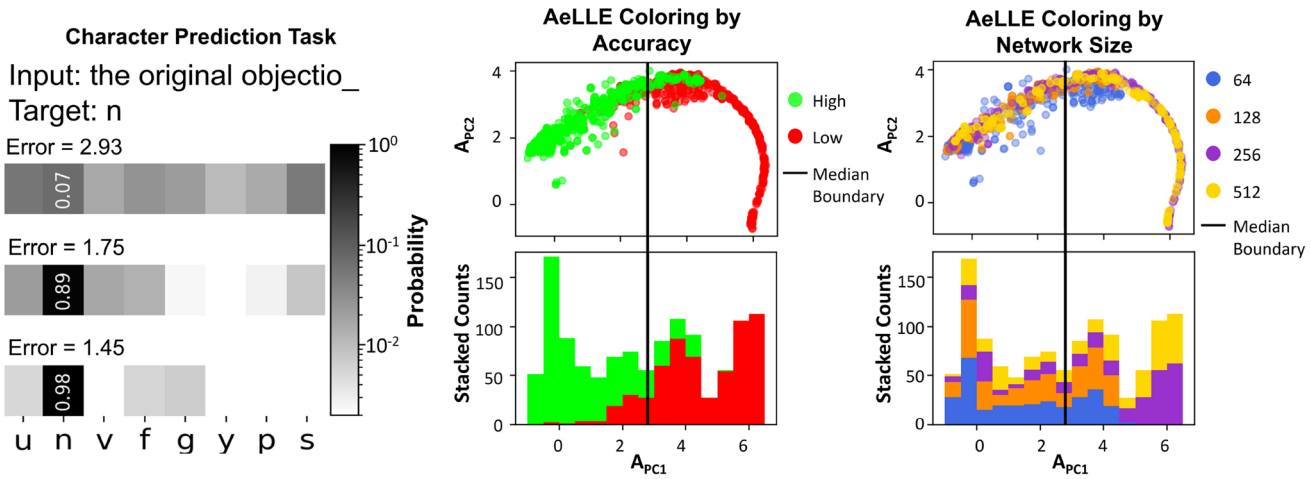

Fig. 3 Clustering by performance across network size: AeLLE for LSTM RNNs trained on CharRNN task. *Left:* The CharRNN task involves predicting the next character in a sequence. Larger losses correspond to less confidence in predicting the next character in a sequence. *Center:* In the AeLLE representation, the higher-accuracy networks, regardless of size, tend to cluster by performance, with the low-accuracy networks to the left in this representation. *Right:* By contrast, LSTM of different sizes is often mixed together in this representation, with larger networks covering a wider range within the AeLLE space, but still overlapping with smaller networks throughout the space

comparison, we show the median value of the first principal component across all networks (black line), showing that the vast majority of high-accuracy networks are to the left of this line. In contrast, the distribution of the network sizes (Fig. 3—right) is more evenly distributed in this space. This demonstrates that this method is able to learn properties from the LE spectrum that correlates with performance across network sizes, which are more informative than network size alone.

Comparing the separation in the AeLLE space with a classifier based on direct LE statistics, we find that using the median value of the mean Lyapunov exponent or max Lyapunov exponent as the decision boundary gives classifiers with F1 scores of 0.834 and 0.859, respectively, suggesting that both statistics are strongly indicative of performance on this task. The LE PC classifier with a decision boundary defined by the median value of the first principal components of the raw Lyapunov exponents has a similar F1 score of 0.860. Meanwhile, the AeLLE classifier achieves an F1 score of 0.877, indicating an improvement in both direct statistics (see Supplementary Materials for more details). This shows that, while the metrics used are indicative of performance, the AeLLE method is able to achieve a slightly better discrimination of network performance.

4.3 Sequential MNIST classification with different network types

A standard benchmark for sequential models is the sequential MNIST task (SMNIST) [59]. In this task, the input is a sequence of pixel grayscale 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 [60], where each image is treated as sequential data, each row is the input at one time, and the 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.

We train a larger number of RNN variants on this task to demonstrate how the AeLLE properties translate across *network architectures*. The architectures trained on this task were: LSTM, gated recurrent unit (GRU), (vanilla) RNN, anti-symmetric RNN (ASRNN) [15], coupled oscillatory RNN (coRNN) [17], Lipschitz RNN [18], Noisy RNN [38], and long expressive memory network (LEM) [61]. For each network type, we train 200 variants of hidden size 64. Every network was trained for 10 epochs, and LE of post-trained networks were collected. This constitutes a set of 1600 variants, where we use 70% for Autoencoder training, 10% for validation, and 20% for testing. For more details on the training of this sequential MNIST task, see Appendix A.3. Similarly to previously described tests, we color code the variants according to accuracy. Networks with loss $<2.2\times 10^{-3}$ are considered as high accuracy (*green*), which includes 50% of networks, while the rest of the networks with higher loss are considered as low accuracy (*red*).

As in the previous tests, AeLLE analysis is able to unravel variants and their accuracy according to LE spectrum. With a median value of PC1 in the AeLLE plane (black line in Fig. 4), the AeLLE plane could be divided

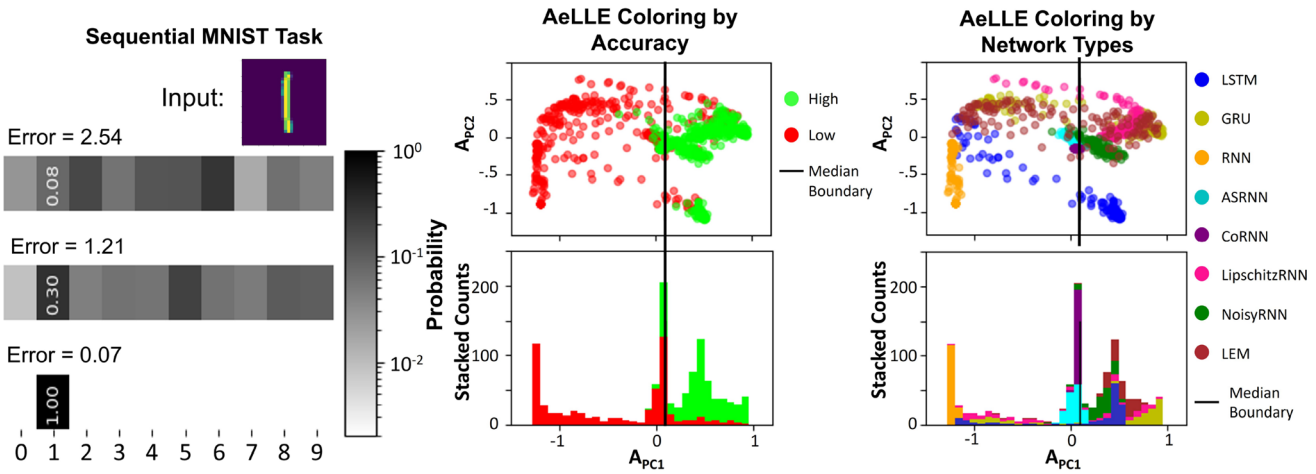

Fig. 4 Clustering across network architecture: AeLLE for networks trained on SMNIST task. *Left:* In this task, the network predicts the digit shown in a given image. Higher losses correspond to lower confidence in the correct digit label. *Center:* AeLLE representation shows that networks of similar error are clustered together. When classifying these networks according to high accuracy (green) and low accuracy (red), the high-accuracy networks, regardless of network architecture, are consistently located further to the right (more positive PC1) than low-accuracy networks. *Right:* Networks of the same type are often clustered together in AeLLE representation, with some overlap between similar architectures (such as coRNN and ASRNN). For each network architecture (except vanilla RNN), there are variants with high and low accuracy separated across the first PC dimension

into two clusters where low accuracy is in the left part of the plane, and high accuracy is in the right part. Through the distribution of network architectures in this representation (see Fig. 4—right), we find clusters of mixed architectures with similar accuracy. Moreover, we find that GRU, Lipschitz RNN, LEM, and, to a lesser extent, Noisy RNN all occupy a similar part of this space, with their best-performing variants generally being located to the top-right of the space shown, and then moving to the left as the performance of the variants deteriorates. While no vanilla RNN variants achieve high accuracy, even networks that have variants with high accuracy, such as LSTM and LEM, have low-accuracy variants that are projected onto the same space as the cluster of vanilla RNNs. Meanwhile, ASRNN and coRNN, both constrained to have dynamics that preserve information, are projected very close to each other in this space into relatively small clusters near the median boundary.

The LE mean and LE max classifier on this dataset achieve an F1 score of 0.609 and 0.566, respectively, suggesting that both statistics are non-optimal predictors of performance on their own. The LE PC classifier has an F1 score of 0.608, representing a minor improvement in accuracy. However, the AeLLE classifier using the median value of the first principal component achieves an F1 score of 0.859 (see Supplementary Materials for more details). This score is a significant improvement on the direct LE statistics. It demonstrates that AeLLE is particularly advantageous for this task, suggesting that characteristics of the dynamics shared across architectures that determine network performance are non-trivial.

4.4 Pre-trained AeLLE for accuracy prediction across training epoch

In the three tests described above, we find that the same general approach of AeLLE allows the 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, the 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., those that underwent 0–50% of training. This test is expected to show how robust the inferred features within AeLLE correlating hyperparameters and LE spectrum are subject to the 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, GRU, RNN, ASRNN, CoRNN, Lipschitz RNN, Noisy RNN, and LEM models. Each model has a hidden states size of 64, and the initialization parameter p is the same as the previous experiment (200 variants for each model). We then compute LE spectrum for the first five epochs of the training (out of all 10 epochs) for all variants. Note, LE spectrum before training is also computed. Therefore, 6000 LE spectra are considered. We then select 20% variants into a training set, and they span over all epochs. We define such AeLLE as a pre-trained AeLLE and investigate its performance.

We select another 20% of data as the validation set (1800 variants) and apply the same loss threshold of 2.2×10^{-3} as above. We use this validation set to define a simple threshold that classifies low- and high-accuracy variants according to AeLLE (Fig. 5 green shaded region of $PC1 > -0.03$).

We then apply the same pre-trained AeLLE and accuracy threshold to variants in the test set (3600 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) does not change significantly over the course of training. Specifically, before training, we find that the recall, i.e., the number of the low-error networks that fall within the low-error threshold region, is 93.8%. The precision, i.e., how many networks in the region are low error, is 77.0%. Then, after a single epoch, 10% of training, the recall becomes 91.5%, and the precision improves to 83.6%, indicating that more of the networks in the region are correctly identified as high accuracy. The recall and precision do not change too much over training. This indicates that the LE spectrum captures properties of network dynamics that emerge with network initialization and remain throughout training.

In summary, we find that pre-trained AeLLE is an effective classifier that predicts the accuracy of the given RNN when fully trained, even before it has undergone 50% training. 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 that variants with low loss early in training correspond to variants that will be classified as low

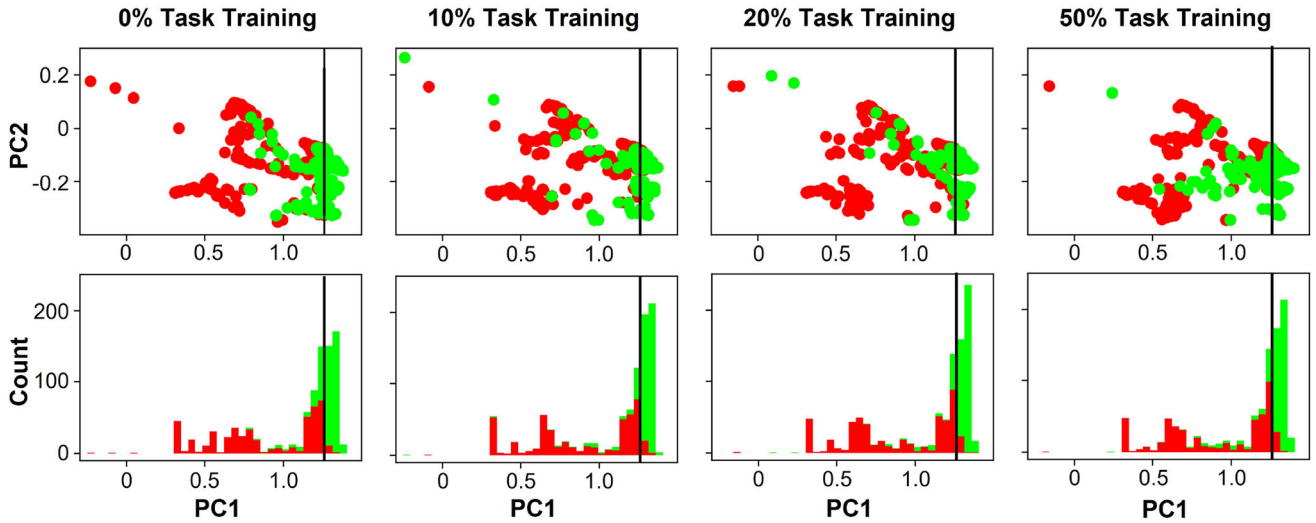

Fig. 5 Pre-trained AeLLE during RNN training on SMNIST task. Prediction of task accuracy of inputs in a test set by pre-trained AeLLE threshold classifier ($PC1 < -0.03$) of SMNIST variants is shown at $(0, 10, 20, 50\%)$ of training of networks on SMNIST. Throughout the training, the high-accuracy networks (green) tend to cluster to the left of the pre-trained AeLLE median classifier (black line). Furthermore, the distribution of networks in this space changes little over training, indicating that the dynamical properties that facilitate networks' successful learning of a task emerge early in training

Table 1 Precision, recall, and F1 score of pre-trained AeLLE classifier for RNN final accuracy evaluated at different stages of training indicated by bold numbers

Training	AeLLE vs. [Loss]			
%	Recall		Precision	F1
0%	93.8%	[–]	77.0% [–]	0.85 [–]
10%	**91.5%**	[16.9%]	**86.6%** [95.1%]	**0.89** [0.35]
20%	**91.8%**	[45.8%]	**86.5%** [97.7%]	**0.90** [0.66]
50%	**88.6%**	[77.7%]	**83.7%** [97.3%]	**0.86** [0.86]

This classification is compared with a classification based on loss value at the corresponding epoch, indicated by [·]

error, indicated by an almost perfect precision rate of 99–100%. However, many variants of high accuracy do not converge quickly. Indeed, the recall rate for a classifier based on loss values is 17–77% for 10–50% of training, while in contrast, AeLLE recall rate is consistently above 88% across all training epochs.

For further comparison, we construct classifiers using the LE mean, LE max, and the first PC of the raw LEs, as training progresses. For each classifier, we select the value of the statistic that yields the best F1 score on the validation set as the decision boundary, and then test the accuracy of such a classifier in determining whether a network's performance at the end of training will be of high or low accuracy on the test set. The distribution of LE statistics and the first PC of the AeLLE classifier over training is shown in Fig. 6. We see that the distribution of the mean LE is heavily concentrated near its maximum value of 0 and does not change significantly over training. Max LE distribution exhibits more changes, but there is a similar number of high- and low-accuracy networks to either side of the boundary throughout training. The first PC of AeLLE has the greatest concentration of high-accuracy networks to one side of the decision boundary, suggesting that this is a more useful classifier. In Table 2, we present the recall, precision, and F1 score of each of these classifiers. This comparison shows that the AeLLE classifier has significantly higher F1 scores than the other classifiers across all epochs.

4.4.1 AeLLE contribution to classification

To delineate the necessity of AeLLE representation, we extend the classifier from a simple classifier (PC1), which we used in earlier sections, to a linear regression classifier. We then perform a comparative study to test whether AeLLE representation is responsible for the effective classification, we observe or whether a more advanced classifier applied directly to LE could achieve a similar outcome. To test this, we use a linear regression classifier in conjunction with AeLLE and compare the classification accuracy with a scenario of using the classifier directly with LE (both full spectrum (Raw) and low dimensions (PCA)). Table 3 shows the results in terms of F1 score and demonstrates that AeLLE is a required step in classification. In particular, in the two scenarios, where AeLLE representation is used, classification in conjunction with PC1 or in conjunction with linear regression achieves

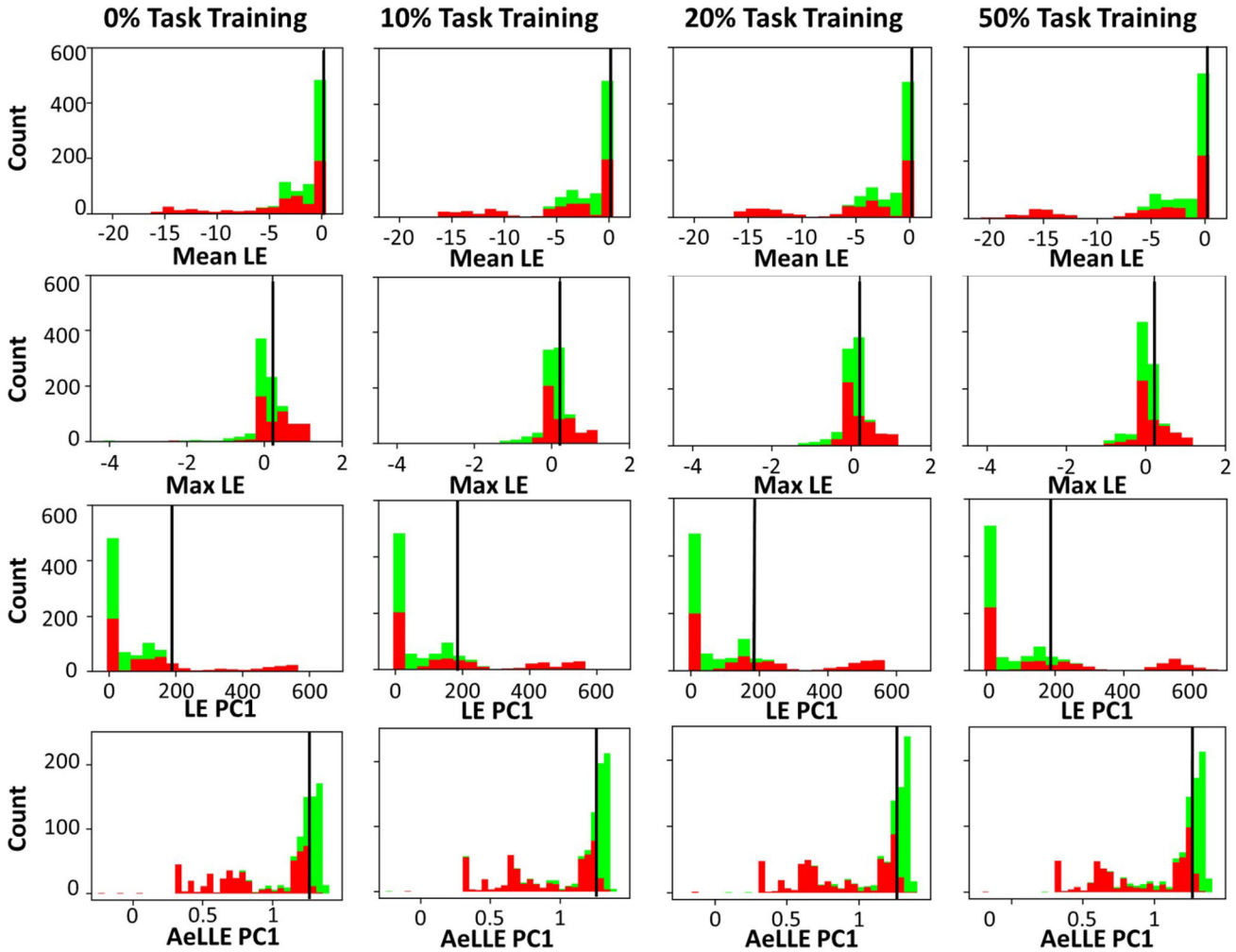

Fig. 6 Comparison of LE mean, LE max, and AeLLE classifiers and distributions throughout network training. The distribution of the mean LE (first row), max LE (second row), and first principal component of the raw Lyapunov exponents (third row), and first principal component of the AeLLE (bottom) are shown from 0% to 50% training on the SMNIST task. The decision boundary value of each metric is shown as a vertical black bar, which is used as a classifier in Table 2

similar F1 scores (the last two rows in Table 3). These scores are significantly higher than the application of linear regression directly to LE values (row 1) or LE values projected to lower dimensional space with PCA (rows 2, 3). We find that for all tested training durations (from 0 to 50%), classifiers applied to AeLLE achieve an F1 score higher by *0.12* on average than the score computed directly on LE or its PCA embedding. In addition, the study suggests that an extension of the classifier, such as extending the classifier from PC1 to linear regression, does not replace AeLLE but rather could provide additional means in conjunction with AeLLE to enhance classification.

4.4.2 Higher dimensions of AeLLE

AeLLE space is not restricted to two dimensions. In general, the more dimensions are used, the more accurate the representation is expected to be. To test this hypothesis, we extend the classification to higher dimensions. Specifically, we use the first d PCs where the median of each PC divides the space into 2^d subspaces. In each subspace, we check the number of optimal and non-optimal network samples. Then, we take the union of all optimal subspaces (those that contain more optimal networks than sub-optimal networks) to be the overall optimal region for the classifier. We then test the classifier on the test dataset for each epoch. Our results are reported in Table 4, showing the F1 score for 0% and 50% training, with d being the number of PCs set to $d = 1, 2, 4, 10$.

The results show that as additional PCs are included, the F1 score increases at the initial phase before training and during training. Hence, the full AeLLE space includes in higher dimensions additional features of LE and the corresponding networks. The results also show that the first-

Table 2 Precision, recall, and F1 score of LE mean, LE max, and AeLLE classifiers throughout training

Training	Classifier	Recall	Precision	F1
0%	LE mean	99.1%	55.6%	0.71
	LE max	65.3%	**90.3%**	0.72
	LE PCA	**100%**	55.9%	0.76
	AeLLE	93.8%	77.0%	**0.85**
10%	LE mean	**97.7%**	58.3%	0.73
	LE max	60.7%	**92.6%**	0.74
	LE PCA	96.8%	60.0%	0.74
	AeLLE	91.5%	86.6%	**0.89**
20%	LE mean	**97.6%**	58.5%	0.73
	LE max	59.5%	**94.3%**	0.73
	LE PCA	96.6%	60.3%	0.74
	AeLLE	91.8%	86.5%	**0.90**
50%	LE mean	**96.4%**	58.9%	0.73
	LE max	58.8%	**94.3%**	0.72
	LE PCA	94.9%	60.4%	0.74
	AeLLE	88.6%	83.7%	**0.86**

The best of each score (precision, recall, and F1) is indicated by bold numbers

order PC classifier is able to capture a significant portion of high- and low-accuracy networks.

4.5 LE features visualization in AeLLE space

Lyapunov spectrum for different architectures can have highly variable maximum values, slopes, variances, means, and more. Specific properties of these spectra would intuitively correlate with performance, but the relation between each feature and accuracy is unclear. This is evident from the comparison of Lyapunov spectra of LSTM, RNN, GRU, and LEM to coRNN, ASRNN, Lipschitz RNN, and Noisy RNN trained on SMNIST in Fig. 7(left). Whereas coRNN, ASRNN, Lipschitz RNN, and Noisy RNN all have exponents close to zero for all indices, the spectra of LSTM, GRU, RNN, and LEM, all dip well below zero, with RNN decreasing much faster than the others.

Table 4 F1 score of pre-trained AeLLE classifier on networks at various levels of training using median from increasing numbers of principal components (PCs)

	#PCs			
Training %	1	2	4	10
0%	0.81	0.81	0.84	0.89
50%	0.84	0.84	0.87	0.91

The total dimension of the AeLLE space is 32

Meanwhile, GRU maintains a far greater number of exponents close to zero, similar to coRNN, for about half of the indices. It would then be natural to assume the performance of all networks with all LEs close to zero would be most similar and that LSTM, GRU, and GRU would all be most similar to RNN in performance.

The representation in AeLLE space (Fig. 7—center) shows two clusters, with one tight cluster representing RNN, LSTM, and LEM and another looser cluster representing all other tested networks. Within this looser cluster, we find both the coRNN network, which has a similar spectrum to ASRNN and Lipschitz RNN, and the GRU, which is visually much closer to LSTM and LEM. However, we find that, within this cluster, GRU is located closest to Lipschitz RNN and Noisy RNN, all of which are found to the right of the median boundary along with ASRNN. Meanwhile, coRNN is located just to the left of the boundary, in the direction of the RNN, LSTM, and LEM. While visual inspection of the spectra does not immediately indicate the reason for this ordering, it becomes more apparent when we observe the loss of the networks (Fig. 7—right). Since ASRNN, Lipschitz RNN, Noisy RNN, and GRU obtain the optimal accuracy (indicated by green color), they are mapped to the right of coRNN and the other networks point in AeLLE plane. Furthermore, while LSTM, RNN, and LEM are different networks in architectures and dynamics, these appear to be mapped to the same cluster in AeLLE. The reason for such non-intuitive mapping could be explained by accuracy again, since on this task, RNN, LSTM, and LEM all exhibit low accuracy (indicated by red color).

Table 3 Comparison of F1 score of PC1 and linear regression classifiers in conjunction with LE-related representations (Raw—row 1, PCA—rows 2, 3, and AeLLE—rows 4, 5) throughout training duration (from 0% training to 50%)

Repr Space and Clf	0%	10%	20%	30%	40%	50%	Testing set
LE (Raw) - Lin. Reg.	0.62	0.61	0.61	0.62	0.64	0.61	0.62
LE (PCA) - PC1	0.72	0.74	0.74	0.75	0.75	0.74	0.74
LE (PCA) - Lin. Reg	0.73	0.77	0.78	0.78	0.77	0.75	0.76
AeLLE (PCA) - PC1	0.85	**0.89**	**0.89**	**0.90**	**0.89**	0.86	**0.88**
AeLLE (PCA) - Lin. Reg	**0.86**	**0.89**	**0.89**	**0.90**	**0.89**	**0.87**	**0.88**

The testing set consists of all samples from 0% training to 50% training. The highest score for each training duration is marked in bold, and classifiers in conjunction with AeLLE are marked in italic

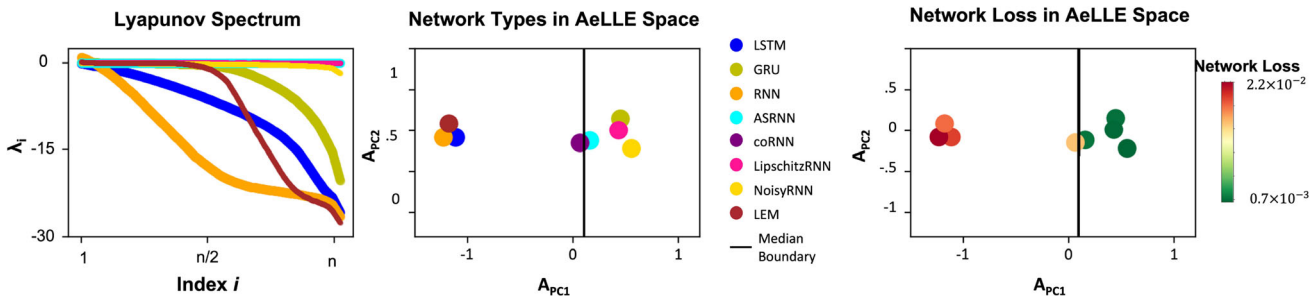

Fig. 7 Comparison of Lyapunov spectrum curves, AeLLE, and loss for different network architectures on SMNIST task. Lyapunov spectrum curves (left) of five different architectures show that the ASRNN, coRNN, Lipschitz RNN, and Noisy RNN have very similar spectra with many exponents close to zero. Meanwhile, the spectra of LSTM, GRU, RNN, and LEM dip well below zero, but at different rates. These networks in AeLLE space (center) are thus grouped such that RNN, LSTM, and LEM are closer to each other than to other network types, but that the GRU is grouped with the networks with spectra close to zero. Such mapping appears to be correlated with accuracy (right; colors indicate loss from low-green to high-red). Despite the similar spectra of coRNN, ASRNN, Lipschitz RNN, and Noisy RNN, the loss of coRNN is larger than that of the others by an order of magnitude, causing it to be separated from some of the other networks in this space. While the LE curves of LSTM, RNN and LEM are distinct, their loss values are much higher (red), and they are clustered in the same vicinity farther from more optimal networks

Such experiments indicate that easily observable LE features, such as the number of exponents near zero or overall spectrum shape, do not vary uniformly with performance. Instead, a more complex, nonlinear combination of LE features extracted by AeLLE would be required to determine this relation.

5 Discussion

LE methodology is an effective toolset to study nonlinear dynamical systems since LE measure the divergence of nearby trajectories and indicate the degree of stability and chaos in that system. Indeed, LE has been applied to various dynamical systems and applications, and there exists theoretical underpinning for the characterization of these systems by LE spectrum. However, it is unclear how to relate LE spectrum to system characterization. Our results demonstrate that the information that LE contain regarding the dynamics of a network can be related to network accuracy on various tasks through an Autoencoder embedding, called AeLLE. In particular, we show that AeLLE representation encodes information about the dynamics of recurrent networks (represented by their Lyapunov exponents) along with the performance. We demonstrated that this relation to performance can be learned across choice of weight initialization, network size, network architecture, and even training epoch. Effectively, AeLLE representation discovers the implicit parameters of the network.

Such a representation is expected to be invaluable in uniting research that looks to assess and predict model quality on particular tasks and those that emphasize and constrain model dynamics to encourage particular solutions. Our approach allows mapping the dynamics of a network to accuracy through the Latent space representation of the LE Autoencoder. This mapping captures multiple characteristics of the networks, with some direct, such as network type, accuracy, and number of units, and some implicit. All these are contained in AeLLE representation and effectively provide salient features/parameters for the networks being considered.

Specifically, the significance of our results is that they show that AeLLE representation is able to distinguish networks according to accuracy across a choice of network architectures with greater accuracy than using simple spectrum statistics alone. Such findings suggest that the features of the dynamics that correspond to optimal performance on given tasks are consistent across network architectures, whether they are RNN, gated architectures, or dynamics-constrained architectures, even if they are not immediately apparent upon visual inspection or through the first-order statistics of the spectrum. AeLLE is able to capture these.

While the accuracy of such a classifier is enhanced over those using direct statistics, this comes at the cost of the training time of the Autoencoder and the interpretability of extracted features. Analysis of the individual components of this Autoencoder methodology (4) could provide more insight into the interpretation of these AeLLE features. Namely, the representations of the Lyapunov exponents that AeLLE produces ($\phi(Z)$) could be directly analyzed statistically or otherwise. The contribution of each dimension of the Latent space of the Autoencoder to the predicted loss value could be extracted from the linear prediction layer (ξ). Furthermore, the corresponding LE spectrum features for these Latent dimensions could be analyzed using the decoder (ψ). Additional experiments exploring the AeLLE representation beyond those outlined here could also be carried out to provide further insights. These studies are left for future work to be applied with optimization and analysis of

particular machine learning tasks, based on generic methodology presented in this work.

Multiple recent works have demonstrated the effectiveness of transformers in processing sequential data [62]. While there is significant overlap between the datasets and problems to which RNN and transformers are often applied, the mechanisms by which they accomplish these tasks are not directly comparable. Notably, transformers process entire sequences at once and use attention mechanisms to emphasize the key points without employing the explicit time-stepping of RNNs. Since the computation of the LE method developed for RNN measures the expansion and contraction over time of the hidden states, a direct comparison between the LE of RNNs and transformers using AeLLE would require identifying analogous dynamic variables for transformers. LE could be computed for activation variables transitioning between layers; however, these variables are not indicative of sequential stepping through the sequence, and their relevance to sequence processing is unclear. The identification of suitable analogous variables for LE spectrum across network types and analysis and comparison of their characteristics through methods such as AeLLE represented in this work offer exciting directions for future research.

While applying AeLLE for the search of optimal networks given a task is outside of the scope of this work, exploration into AeLLE representation to find predicted optimal dynamics for a task would be a natural extension of the results reported here. Furthermore, an extension of AeLLE methodology is to search and unravel novel architectures with desired dynamics defined by a particular LE spectrum. The AeLLE approach can also be adapted to analyze other complex dynamical systems. For example, long-term forecasting of temporal signals from dynamical systems is a challenging problem addressed with a similar data-driven approach using Autoencoders and spectral methods along with linearization or physics-informed constraints [63–67]. 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 are efficient computation of Lyapunov exponents, sufficient sampling of data to train the Autoencoder to form an informative Latent representation, and stable back-propagation of gradients across many iterations of QR decomposition in the LE calculation.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00521-024-09824-6.

Acknowledgements The authors thank Dr. Guillaume Lajoie and Dr. Maximilian Puelma Touzel for stimulating discussions regarding the role of Lyapunov Exponents features for RNNs, Dr. Eric Shea-Brown for helpful feedback, and Dr. Panos Stinis for providing the inspiration for this work. The authors acknowledge the partial support of HDR Institute: Accelerated AI Algorithms for Data-Driven Discovery (A3D3) National Science Foundation grant PHY-2117997 (RV, YZ, ES). National Science Foundation grant IIS-2036255 (ES), and Washington Research Fund (ES). Authors also acknowledge the partial support by the Departments of Electrical Computer Engineering (YZ, ES), Applied Mathematics (RV,ES). Authors are thankful to the Center of Computational Neuroscience and the eScience Center at the University of Washington.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

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

References

1. Pang B, Zha K, Cao H, Shi C, Lu C (2019) Deep RNN framework for visual sequential applications. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)
2. Mikolov T, Karafiát M, Burget L, Černockỳ J, Khudanpur S (2010) Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association
3. Das S, Olurotimi O (1998) Noisy recurrent neural networks: the continuous-time case. IEEE Trans Neural Netw 9(5):913–936. https://doi.org/10.1109/72.712164
4. Tino P, Schittenkopf C, Dorffner G (2001) Financial volatility trading using recurrent neural networks. IEEE Trans Neural Netw 12(4):865–874
5. Su K, Liu X, Shlizerman E(2020) Predict & cluster: unsupervised skeleton based action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 9631– 9640
6. Pennington J, Schoenholz S, Ganguli S (2017) Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In: Advances in Neural Information Processing Systems, pp 4785– 4795
7. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp 3104– 3112

8. Gregor K, Danihelka I, Graves A, Rezende D, Wierstra D (2015) Draw: a recurrent neural network for image generation. In: International Conference on Machine Learning, pp 1462– 1471 . PMLR
9. Choi K, Fazekas G, Sandler MB (2016) Text-based lstm networks for automatic music composition. CoRR arXiv: abs/1604.05358
10. Mao HH, Shin T, Cottrell G (2018) Deepj: Style-specific music generation. In: 2018 IEEE 12th International Conference on Semantic Computing (ICSC), pp 377– 382 . IEEE
11. Guo D, Zhou W, Li H, Wang M (2018) Hierarchical LSTM for sign language translation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 32
12. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
13. Cho K, Merrienboer B, Gülçehre Ç, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR **abs/1406.1078**arXiv:1406.1078
14. Chung J, Gülçehre Ç, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR abs/1412.3555 arXiv: 1412.3555
15. Chang B, Chen M, Haber E, Chi EH (2019) AntisymmetricRNN: a dynamical system view on recurrent neural networks. In: International Conference on Learning Representations . https://openreview.net/forum?id=ryxepo0cFX
16. Vorontsov E, Trabelsi C, Kadoury S, Pal C (2017) On orthogonality and learning recurrent networks with long term dependencies. In: International Conference on Machine Learning, pp 3570– 3578 . PMLR
17. Rusch TK, Mishra S (2021) Coupled oscillatory recurrent neural network (coRNN): an accurate and (gradient) stable architecture for learning long time dependencies. In: International Conference on Learning Representations
18. Erichson NB, Azencot O, Queiruga A, Hodgkinson L, Mahoney MW (2020) Lipschitz recurrent neural networks. In: International Conference on Learning Representations
19. Ruelle D (1979) Ergodic theory of differentiable dynamical systems. Publications Mathématiques de l'Institut des Hautes Études Scientifiques 50(1):27–58
20. Oseledets V (2008) Oseledets theorem. Scholarpedia 3(1):1846. https://doi.org/10.4249/scholarpedia.1846. (**revision #142085**)
21. Engelken R, Wolf F, Abbott LF (2020) Lyapunov spectra of chaotic recurrent neural networks. arXiv preprint arXiv:2006.02427
22. Vogt R, Puelma Touzel M, Shlizerman E, Lajoie G (2022) On Lyapunov exponents for RNNs: understanding information propagation using dynamical systems tools. Front Appl Math Stat. https://doi.org/10.3389/fams.2022.818799
23. Mikhaeil J, Monfared Z, Durstewitz D (2022) On the difficulty of learning chaotic dynamics with rnns. Adv Neural Inform Process Syst 35:11297–11312
24. Herrmann L, Granz M, Landgraf T (2022) Chaotic dynamics are intrinsic to neural network training with sgd. Adv Neural Inform Process Syst 35:5219–5229
25. Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp 1310– 1318
26. Poole B, Lahiri S, Raghu M, Sohl-Dickstein J, Ganguli S (2016) Exponential expressivity in deep neural networks through transient chaos. In: Advances in Neural Information Processing Systems, pp 3360– 3368
27. Wang B, Hoai M (2018) 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 . IEEE
28. Chen M, Pennington J, Schoenholz S (2018) Dynamical isometry and a mean field theory of RNNs: gating enables signal propagation in recurrent neural networks. In: International Conference on Machine Learning, pp 873– 882 . PMLR
29. Yang G (2019) Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. arXiv preprint arXiv:1902.04760
30. Zheng Y, Shlizerman E (2020) R-FORCE: Robust learning for random recurrent neural networks. arXiv preprint arXiv:2003.11660
31. Martin C, Peng T, Mahoney M (2021) Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. Nat Commun 12:4122. https://doi.org/10.1038/s41467-021-24025-8
32. Naiman I, Azencot O (2021) A Koopman approach to understanding sequence neural models. arXiv . https://doi.org/10.48550/ARXIV.2102.07824
33. Wisdom S, Powers T, Hershey J, Le Roux J, Atlas L (2016) Full-capacity unitary recurrent neural networks. Adv Neural Inform Process Syst 29:4880–4888
34. Jing L, Shen Y, Dubcek T, Peurifoy J, Skirlo S, LeCun Y, Tegmark M, Soljačić M (2017) Tunable efficient unitary neural networks (eUNN) and their application to RNNs. In: International Conference on Machine Learning, pp 1733– 1741 . PMLR
35. Mhammedi Z, Hellicar A, Rahman A, Bailey J (2017) Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In: International Conference on Machine Learning, pp 2401– 2409 . PMLR
36. Azencot O, Erichson NB, Ben-Chen M, Mahoney MW (2021) A differential geometry perspective on orthogonal recurrent models. arXiv preprint arXiv:2102.09589
37. Kerg G, Goyette K, Puelma Touzel M, Gidel G, Vorontsov E, Bengio Y, Lajoie G (2019) Non-normal recurrent neural network (nnRNN): learning long time dependencies while improving expressivity with transient dynamics. Adv Neural Inform Process Syst 32
38. Lim SH, Erichson NB, Hodgkinson L, Mahoney MW (2021) Noisy recurrent neural networks. Adv Neural Inform Process Syst 34:5124–5137
39. Maheswaranathan N, Williams A, Golub M, Ganguli S, Sussillo D (2019) Universality and individuality in neural dynamics across large populations of recurrent networks. Adv Neural Inform Process Syst 32
40. platform O (1930) The ordinal numbers of systems of linear differential equations. Math J 31(1):748–766
41. Dawson S, Grebogi C, Sauer T, Yorke JA (1994) Obstructions to shadowing when a Lyapunov exponent fluctuates about zero. Phys Rev Lett 73:1927–1930. https://doi.org/10.1103/PhysRevLett.73.1927
42. Abarbanel HDI, Brown R, Kennel MB (1991) Variation of Lyapunov exponents on a strange attractor. J Nonlinear Sci 1(2):175–199. https://doi.org/10.1007/BF01209065
43. Shibata H (2001) Ks entropy and mean Lyapunov exponent for coupled map lattices. Physica A 292(1):182–192. https://doi.org/10.1016/S0378-4371(00)00591-4
44. Saitô N, Ichimura A (1979) Ergodic components in the stochastic region in a Hamiltonian system. In: Casati G, Ford J (eds) Stochastic Behavior in Classical and Quantum Hamiltonian Systems. Springer, Berlin, Heidelberg, pp 137–144
45. Ochs G (1999) Stability of Oseledets spaces is equivalent to stability of Lyapunov exponents. Dyn Stab Syst 14(2):183–201
46. Geist K, Parlitz U, Lauterborn W (1990) Comparison of Different Methods for Computing Lyapunov Exponents. Progress Theoret Phys 83(5):875–893. https://doi.org/10.1143/PTP.83.875 (https://

academic.oup.com/ptp/article-pdf/83/5/875/5302061/83-5-875.pdf)
47. Arnold L (1995) Random dynamical systems. Dyn Syst 1–43
48. Benettin G, Galgani L, Giorgilli A, Strelcyn J.-M (1980) 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
49. Dieci L, Van Vleck ES (1995) Computation of a few Lyapunov exponents for continuous and discrete dynamical systems. Appl Num Math 17(3):275–291
50. Goodfellow IJ, Bengio Y, Courville A (2016) Deep Learning. MIT Press, Cambridge, MA, USA . http://www.deeplearningbook.org
51. Chollet F (2021) Deep Learning with Python. Simon and Schuster, ???
52. Liii Pearson K (1901) on lines and planes of closest fit to systems of points in space. The London. Edinburgh Dublin Philos Mag J Sci 2(11):559–572
53. Su K, Shlizerman E (2020) Clustering and recognition of spatiotemporal features through interpretable embedding of sequence to sequence recurrent neural networks. Front Artif Intell 3:70
54. Maaten L, Hinton G(2008) Visualizing data using t-sne. J Mach Learn Res 9(11)
55. McInnes L, Healy J, Saul N, Großberger L (2018) Umap: uniform manifold approximation and projection. J Open Sour Softw 3(29):861
56. Sussillo D, Abbott LF (2009) Generating coherent patterns of activity from chaotic neural networks. Neuron 63(4):544–557
57. DePasquale B, Cueva CJ, Rajan K, Escola GS, Abbott L (2018) full-FORCE: a target-based method for training recurrent networks. PLoS ONE 13(2):0191527
58. Karpathy A, Johnson J, Fei-Fei L (2015) Visualizing and understanding recurrent networks. arXiv e-prints, 1506
59. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
60. Bai S, Zico Kolter J, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv e-prints, 1803
61. Rusch TK, Mishra S, Erichson NB, Mahoney MW (2022) Long expressive memory for sequence modeling
62. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30
63. Lusch B, Kutz JN, Brunton SL (2018) Deep learning for universal linear embeddings of nonlinear dynamics. Nat Commun 9(1):4950. https://doi.org/10.1038/s41467-018-07210-0
64. Lange H, Brunton SL, Kutz JN (2021) From Fourier to Koopman: spectral methods for long-term time series prediction. J Mach Learn Res 22(41):1–38
65. Morton J, Jameson A, Kochenderfer MJ, Witherden F (2018) Deep dynamical modeling and control of unsteady fluid flows. Adv Neural Inform Process Syst 31
66. Erichson NB, Muehlebach M, Mahoney MW (2019) Physics-informed Autoencoders for Lyapunov-stable Fluid Flow Prediction. arXiv . https://doi.org/10.48550/ARXIV.1905.10866
67. Azencot O, Erichson NB, Lin V, Mahoney M(2020) Forecasting sequential data using consistent Koopman autoencoders. In: International Conference on Machine Learning, pp 475– 485 . PMLR