# Takens-inspired neuromorphic processor: A downsizing tool for random recurrent neural networks via feature extraction

Bicky A. Marquez,[1,*] Jose Suarez-Vargas,[2,3] and Bhavin J. Shastri[1]

[1]*Department of Physics, Engineering Physics & Astronomy, Queen's University, Kingston, Ontario, Canada K7L 3N6*

[2]*Elettra-Sincrotrone Trieste, Strada Statale 14-km 163,5, 34149 Basovizza, Trieste, Italy*

[3]*International Centre for Theoretical Physics, Strada Costiera 11, I-34151 Trieste, Italy*

(Received 6 July 2019; published 17 October 2019)

We describe a technique which minimizes the amount of neurons in the hidden layer of a random recurrent neural network (rRNN) for time series prediction. Merging Takens-based attractor reconstruction methods with machine learning, we identify a mechanism for feature extraction that can be leveraged to lower the network size. We obtain criteria specific to the particular prediction task and derive the scaling law of the prediction error. The consequences of our theory are demonstrated by designing a Takens-inspired hybrid processor, which extends a rRNN with virtual nodes. Virtual nodes are defined as time-delayed versions of real network nodes. Our hybrid architecture is therefore designed including both real and virtual nodes. Via this symbiosis, we show performance of the hybrid processor by stabilizing an arrhythmic neural model. Thanks to our obtained design rules, we can reduce the stabilizing neural network's size by a factor of 15 with respect to a standard system.

DOI: 10.1103/PhysRevResearch.1.033030

## I. INTRODUCTION

Artificial neural networks (ANNs) are systems prominently used in computational science as well as investigations of biological neural systems. In biology, of particular interest are recurrent neural networks (RNNs) whose structure can be compared among others with nervous system's networks of advanced biological species [1]. In computation, RNNs have been used to solve highly complex tasks which pose problems to other classical computational approaches [2–6]. Their recurrent architecture allows the generation of internal dynamics, and consequently RNNs can be studied utilizing principles of dynamical systems theory. Therefore, the network's nonlinear dynamical properties are of major importance to its information processing capacity. In fact, optimal computational performances are often achieved in a stable equilibrium network's state, yet near criticality [7,8].

Among the recurrent networks reported in current literature, random recurrent neural networks (rRNNs) are popular models for investigating fundamental principles of information processing. In these models, the synaptic neural links are randomly weighted, typically following a uniform [9] or a Gaussian distribution [10–13]. Recently, there is an increasing interest in some particular types of random recurrent networks with a simplified design, where just the output layers are trained using a supervised learning rule. Such rRNNs are typically referred to as reservoir computers, which include echo state networks [2] and also liquid state machines [14]. Reservoir computing provides state-of-the-art performance for challenging problems like high-quality long-term prediction of chaotic signals [2,15].

Prediction corresponds to estimating the future developments of a system based on knowledge about its past. Chaotic time series prediction is of importance to a large variety of fields, including the forecasting of weather [16], the evolution of some human pathologies [17], population density growth [18], or dynamical control as found in the regulation of chaotic physiological functions [19–21]. In order to build a predictor for chaotic systems, most common techniques can be divided into the following groups [22]: (i) linear and nonlinear regression models such as autoregressive-moving average, multiadaptive regression spline [23], and support vector machine [24]; (ii) state-space-based techniques for prediction of continuous-time chaotic systems, which utilize attractor reconstruction and interactions between internal degrees of freedom to infer the future [22,25–28]. Attractor reconstruction method is based on the embedding of the original state space in a delay-coordinate space [29]. (iii) The connectionist approach, including recurrent and feed-forward [30,31], deep [32], and convolutional ANNs [33]. This approach usually comprehends the design of ANNs using large amounts of neurons to process information [34].

The high dimensionality of the ANNs' hidden layer is commonly translated in a computationally expensive problem when considering the optimization of such networks to solve a task. In the reservoir computing approach such training efforts are reduced due to the training is done on the output layer only. However, the more neurons in the hidden layer which are connected to the output layer, the higher the computational cost of the training step. Introducing a unconventional methodology, we develop a state-space-based concept which

*bama@queensu.ca

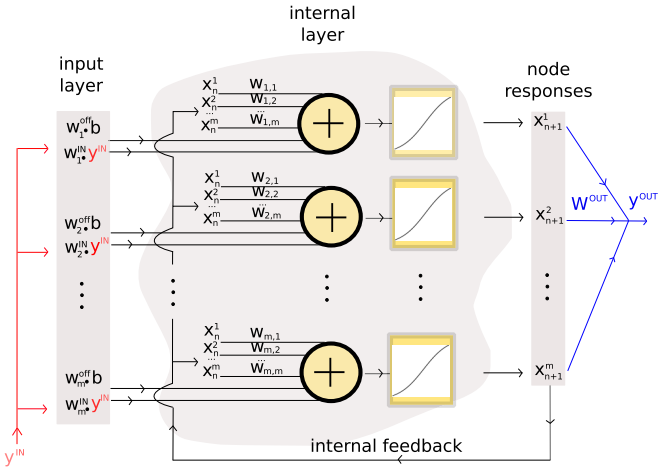


FIG. 1. Explicit illustration of the rRNN diagram. The network is composed by an input layer, where information $y^{\text{in}}$ and $b$ enter to the hidden layer via random input and bias weights vectors $W^{\text{in}}$ and $W^{\text{off}}$, respectively. The internal layer has $m$ neurons whose synaptic weights are defined by the elements of the matrix $W$. The neurons' nonlinear activation functions are hyperbolic tangents. Node responses are internally fed back to the internal layer, yielding to the recurrent architecture of the network. A readout state $y^{\text{out}}$ is created via the readout weight matrix $W^{\text{out}}$.

guides the downsizing of the rRNNs' hidden layer. To achieve this objective, we describe rRNNs and state-space-based models within the same framework. This step allows us to show state-space patterns revealed by spontaneous reconstructions inside the high-dimensional space of our random recurrent network. Furthermore, we introduce a methodology based on the Takens embedding theorem to identify the embedding dimensions of such input system's spontaneous reconstruction, and their relevance to the system's prediction performance.

We immediately exploit our insight and devise a hybrid Takens-inspired ANN concept, in which a network is extended by an *a priori* designed delay external memory. The delay term is used to virtually extend the size of the network by introducing virtual nodes [35] which exist in the delay path. We use this design to first validate our interpretation, and then devise an advanced hybrid rRNN to stabilize a nonperiodic neuronal model which requires 15 times less neurons than a benchmark rRNN [36,37]. As this system is driven by a stochastic signal, we show how our approach can leverage properties of the underlying deterministic system even for the case of a stochastic drive.

## II. RANDOM RECURRENT NETWORKS FOR PREDICTION

A rRNN is illustrated in Fig. 1, indicating the temporal flow of information received by each neuron or node. Nodes are represented by $\bigoplus$. The rRNN consists of a reservoir of $m$ nodes in state $\mathbf{x}_n$ at integer time $n$. Nodes are connected through random, uniformly distributed internal weights defined as coefficients in the matrix $W$ of dimensionality $m\times m$. The resulting randomly connected network is injected with one-dimensional (1D) input data $y^{\text{in}}_{n+1}$ according to input weights defined as random coefficients in the vector $W^{\text{in}}$ of dimensionality $m\times 1$. The time-discrete equation that governs the network is [2]

$$\mathbf{x}_{n+1} = f_{NL}\big(\mu W\cdot\mathbf{x}_n + \alpha W^{\text{in}}\cdot y^{\text{in}}_{n+1} + W^{\text{off}}\cdot b\big), \tag{1}$$

where $\mu$ is the bifurcation parameter that controls network's internal dynamics, $\alpha$ the input gain, $f_{NL}(\cdot)$ is a nonlinear sigmoidlike activation function, and $b$ the constant phase offset injected through offset weights, defined as random coefficients in the vector $W^{\text{off}}$ of dimensionality $m\times 1$. In practice, we construct a network with $m = 1000$, using the MATLAB routine `random`. Connection weights $W_{i,j}$ are distributed around zero. In order to set our recurrent network in its steady state, $W$ is normalized by its largest eigenvalue. The network's connectivity is set to one, hence, it is fully connected.

An output layer creates the solution $y^{\text{out}}$ to the prediction task. In this step the network approximates the underlying deterministic law that rules the evolution of the input system. The output layer provides the computational result according to

$$y^{\text{out}}_{n+1} = W^{\text{out}}\cdot\mathbf{x}_{n+1}. \tag{2}$$

The output weights vector $W^{\text{out}}$ is calculated according to a supervised learning rule, using a reference teacher/target signal $y^T_{n+1}$ [38]. We calculate the optimal output weights vector $W^{\text{out}}_{op}$ by

$$W^{\text{out}}_{op} = \min_{W^{\text{out}}}\left\|W^{\text{out}}\cdot\mathbf{x}_{n+1} - y^T_{n+1}\right\|, \tag{3}$$

via its pseudoinverse (using singular value decomposition) with the MATLABroutine `pinv`. Equation (3) therefore minimizes the error between output $W^{\text{out}}\cdot\mathbf{x}_{n+1}$ and teacher $y^T_{n+1}$. As training error measure we use the normalized mean-squared error (NMSE) between output $y^{\text{out}}_{n+1}$ and target signal $y^T_{n+1}$, normalized by the standard deviation of teacher signal $y^T_{n+1}$.

When a rRNN is used for time-series prediction of a chaotic oscillator such as the Mackey-Glass (MG) time-delayed system [17], it can achieve good long-term prediction performances with 1000 neurons. Here, long-term predictions are defined as predictions far beyond one step in the future. The task is to predict future steps of the chaotic MG system in its discrete-time version:

$$y_{n+1} = y_n + \delta\left(\frac{\vartheta y_{\tau_m}}{1+(y_{\tau_m})^{\nu}} - \psi y_n\right), \tag{4}$$

where $y_{\tau_m} = y_{(n-\tau_m/\delta)}$, $\tau_m = 17$ as the time delay, and $\delta = \frac{1}{10}$ is the step size indicating that the time series is subsampled by 10. The other parameters are set to $\vartheta = 0.2$, $\nu = 10$, $\psi = 0.1$. For any prediction task in this paper, we consider 20 network models via different initializations of $\{W, W^{\text{in}}, W^{\text{off}}\}$. The prediction horizon is estimated to be 300 time steps, defined by the inverse of the largest Lyapunov exponent of the MG system ($\lambda^{\text{MG}}_{\max}\simeq 0.0036$). For such prediction horizon, and for predicting the value of 20 different MG sequences, we obtain the average of all NMSEs, resulting in $0.091\pm 0.013$. This performance was obtained for $b = 0.2$, $\alpha = 0.8$, and bifurcation parameter $\mu = 1.1$, which was found to offer the best prediction performance in a range of $\mu\in[0.1, 1.3]$. Moreover, the network was trained with 3000 values of the

MG system, with a teacher signal given by $y^T_{n+1} = y^{\text{in}}_{n+1}$. We subtracted the average of the MG time series before injection into the rRNN, which is a common practice [38]. Then, we discarded the first 1000 points of the network's response to avoid initial transients. Right after training, where $W^{\text{out}}$ was determined, we connected $y^{\text{out}}_{n+1}$ to $y^{\text{in}}_{n+1}$ and left the network running freely 300 time steps, indicated by the prediction horizon.

Given that good long-term prediction performances are obtained, we wonder what is the underlying process carried out by the network when processing such information. To tackle this interrogation, we take inspiration from qualitative investigations that are common practices in image classification tasks, where the extracted features are often identified and illustrated together with the hidden layers that have generated them [39]. In the following, we introduce a technique that allows us to identify feature representations of the input information in the rRNN's high-dimensional space, which are linked to good prediction performances.

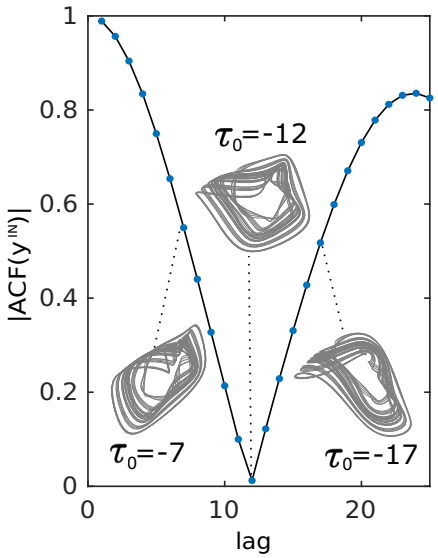


FIG. 2. Absolute value of the autocorrelation function for the MG system together with three examples of 2D delay reconstruction for embedding lags $\{-7, -12, -17\}$.

## III. A METHOD FOR FEATURE EXTRACTION IN RANDOM RECURRENT NETWORKS

As shown previously, our rRNN is able to predict the future values of 1D input chaotic data $y^{\text{in}}$. Such kind of data come from a continuous-time chaotic system, i.e., the MG system. As it is known in chaos theory, a minimum of three dimensions are required in a continuous-time nonlinear system to generate chaotic solutions. Typically, continuous-time chaotic solutions come from models consisting of a system of at least three nonlinear ordinary differential equations (ODEs). However, there are other ways to obtain such chaotic dynamics. One of those ways includes the introduction of an explicit temporal variable to an ODE, such as a time delay with respect to the main temporal variable. We define these models as delay differential equations (DDEs), where a DDE is in fact equivalent to an infinite-dimensional system of differential equations [40]. The number of solutions to a DDE is in theory infinite due to the infinite amount of initial conditions in the continuous rank required to solve the equation. Each initial condition initializes an ODE from the infinite-dimensional system of equations. Thus, the introduction of a time delay in an ODE, resulting in a DDE, provides sufficient dimensionality to allow for the existence of chaotic solutions. For instance, this is how the MG system can develop chaotic solutions. In such case, one just has access to a time series from a single accessible dimension, represented by the variable $y_{n+1}$ in Eq. (4), while all others remain hidden. Nevertheless, hidden variables are participating in the development of the global dynamics as well as the accessible variables.

In order to approximate a full-dimensional representation of these oscillators, we could embed the 1D sequence $y_{n+1}$ into a high- dimensional space, consequently reconstructing its state space. A state space is defined as the geometric space created by all dimensions of the original dynamical system, where the evolution of the system's state trajectory is represented. Among the most practiced methods to embed 1D information, we highlight state-space reconstruction techniques such as delay reconstruction (Whitney and Takens embedding theorems [41,42]) or through the Hilbert transform [43].

Delay reconstruction is a widely used method to complete missing state-space information. According to the Takens embedding theorem, the time-delayed version of a time series suffices to reveal the structure of the state-space trajectory. Let us represent the data in a $M$-dimensional space by the vectors $\mathbf{y}_n = [y_n,\ y_{(n+\tau_0)},\ \ldots,\ y_{(n+(M-1)\tau_0)}]^\dagger$, where $[\cdot]^\dagger$ is a transpose matrix and $y_n$ is the original time series. The essential parameters for state-space reconstruction are $M$ and time delay $\tau_0$. Embedding delay $\tau_0$ is often estimated by applying autocorrelation analysis or time-delayed mutual information to the signal $y_n$. The temporal position of the first zero [25] of either method maximizes the possibility to extract additional information which contains independent observations from the original signal, and hence obtain trajectories or dynamic motion along potentially orthogonal state-space dimensions. If successful, information of such maximized linear independence can enable the inference of the missing degrees of freedom [42,44,45]. On the other hand, we estimate the minimum amount of required embedding dimensions $M$ by using the method of *false nearest neighbors* [25]. Crucially, the following concepts are not restricted to a particular method of determining $\tau_0$ or $M$.

The autocorrelation function (ACF) is often employed to identify the temporal position $\tau_0$ used to reconstruct missing coordinates that define any continuous-time dynamical system. In order to show how Takens-based attractor reconstruction performs, we use a data set of $1\times10^4$ values provided by Eq. (4). The outcome of the autocorrelation analysis reveals that the ACF has its first zero at lag $\tau_0 = -12$. In Fig. 2, we show the absolute value of the ACF together with three examples of two-dimensional (2D) delay reconstructions for three different values of the embedding lag $\tau_0$. As it can be seen, 2D reconstructions based on lags $\tau_0 = -7$ and $-17$ also unfold the geometrical object within a state space. However,

aiming at a maximally orthogonal embedding, we base our analysis on the attractors reconstructed by exactly the first zero of the ACF, $\tau_0 = -12$. According to the false nearest-neighbor analysis, the *minimum* dimensions $M$ required to reconstruct the MG attractor is 4. The Takens scheme therefore provides a set of coordinates $\mathbf{y}_n = \{y_n, y_{n-12}, y_{n-24}, y_{n-36}\}$ which reconstructs the state-space object.

As a matter of fact, Takens embedding technique can be interpreted as a classical method to extract feature representations $\mathbf{y}_n$ from the original sequence $y_n$. Accordingly, comparable attractor reconstruction inside our rRNN's state space could be identified. In the following, a method to extract similar features is introduced.

### A. A Takens-inspired feature extraction technique

Our analysis begins by specifying the network's state space, which is defined by the set of random orthogonal vectors in $W$. In order to identify possible attractor reconstruction inside a rRNN's state space, we analyze the injected signal representation $y^{\text{in}}_{n+1}$ via network's node responses. As input information $y^{\text{in}}_{n+1}$ is being randomly injected into the network's high-dimensional space, we search for possible spatial representations of the 1D input, where network nodes serve as embedding dimensions.

With that goal in mind, we proceed with an analysis comparable to the ACF used in delay embedding, based on the estimation of the maximum absolute value of the cross-correlation function $|\text{CC}(x^i, y^{\text{in}})|_{\max}$ between all node responses $\{x^i\}$ and the input data $y^{\text{in}}$. As our aim is to identify nodes providing observations approximately orthogonal to $y^{\text{in}}$, we record for every network node such cross-correlation maximum and the temporal position, or lag, of this maximum. Figure 3(a) shows the cross-correlation analysis (CCA) for $\mu = 1.1$, where node correlation lags and maxima correspond to the abscissa and ordinate, respectively. In this case, a distribution of node lags, where an extension of the range to $\{l_{\min}, l_{\max}\} = \{-49, 45\}$, is shown. Thus, the rRNN's nodes reveal a strong cross correlation at time lags covering all Takens embedding delays $\{0, \tau_0, 2\tau_0, 3\tau_0\}$. Nodes lagged around $\{-36, -24, -12, 0\}$ should well approximate the Takens embedded attractor, and a state-space representation of the input sequence is presented within the rRNNs' space.

In Fig. 3(b), we illustrate some of the numerous possible extracted features from the originally injected attractor embedded in the rRNN's space for $\mu = 1.1$. The first column of the figure shows simple 2D projections of the delay-reconstructed attractor by using the set of lags $\{-24, -18, -12, -6\}$. The attractors reconstructed by network nodes lagged at $\{-24, -18, -12, -6\}$ are shown in the three next columns, where each 2D projection was reconstructed with network nodes enlisted in Table I. This set of projections shows some random hidden feature representations of the input data in the rRNN's high-dimensional space. To build such projections, network nodes chosen from Fig. 3(a) represent the ordinate, and the input data $y^{\text{in}}$ represent the abscissa. Attractors reconstructed by network nodes with maximum cross-correlation values at lags $\{-18, -6\}$ do not belong to the set of original Takens delay coordinates. We included these additional delay dimensions to better illustrate

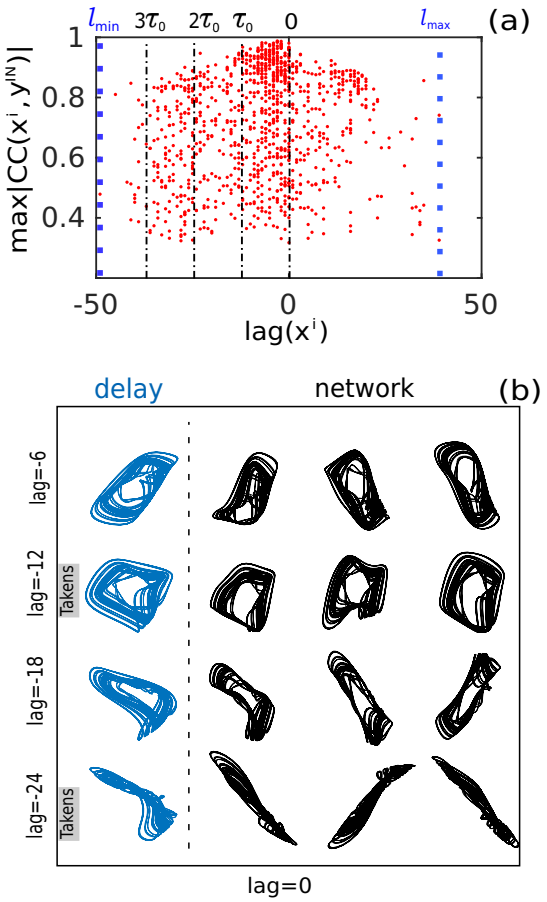


FIG. 3. (a) Maximum absolute value of the cross-correlation function between node responses $x^i$ and input signal $y^{\text{in}}$ for $\mu = 1.1$, in which each of the 1000 reservoir nodes is considered with a red dot. Labels $\{l_{\min}, l_{\max}\}$ remark the lags' limits for the distributions of nodes, and $\tau_0 = -12$. (b) The first column shows the 2D projection of different high-dimensional attractors of the MG system, using diverse time lags. The following three columns show the embedding found inside the rRNN's space for $\mu = 1.1$.

the breadth of features provided by the rRNN. Hence, all extracted features can be qualitatively compared with input delay-reconstructed 2D projections, shown in the first column of Fig. 3(b). At this point, we wonder what is the relevance of such identified features to the network's prediction performance. Then, in the next subsection we add a quantitative comparison between those features and the original input attractor.

Our analysis begins highlighting the fact that the CCA also finds node lags and correlations that do not agree with the

TABLE I. Lags and network nodes list that build 2D projections shown by Fig. 3(b) when plotted against $y^{\text{in}}$.

| Lag | Network nodes | | |
|---|---|---|---|
| −6 | 13 | 91 | 302 |
| −12 | 5 | 119 | 648 |
| −18 | 396 | 771 | 848 |
| −24 | 158 | 24 | 700 |

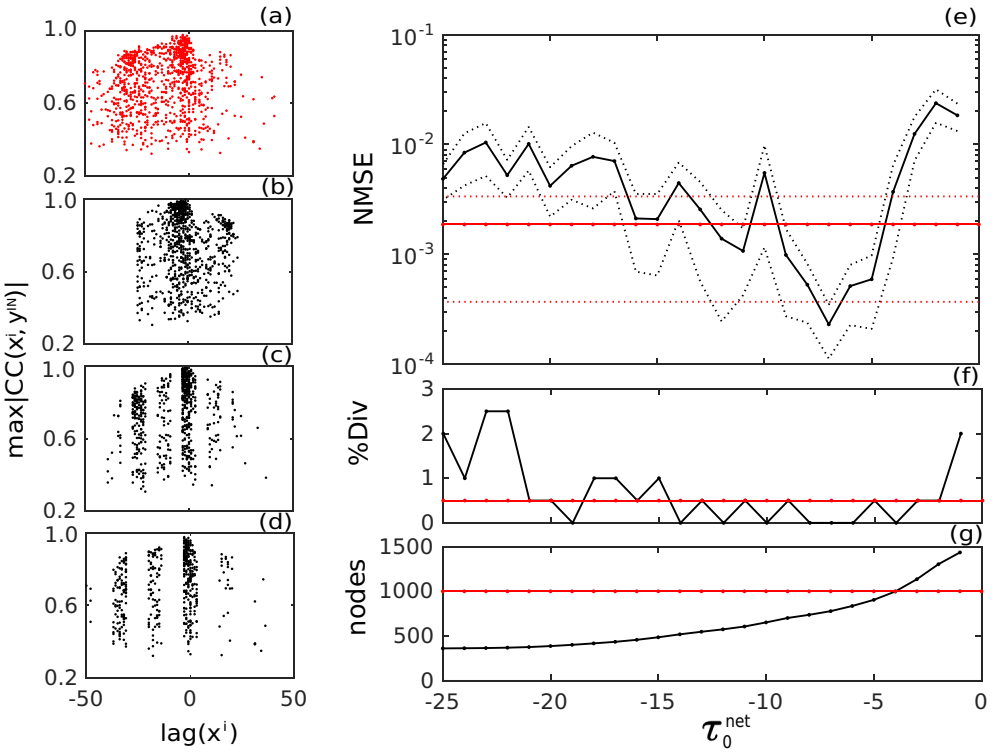


FIG. 4. Maximum absolute value of the cross-correlation function between node responses and input signal for $\mu = 1.1$, considering the rRNN network with (a) all nodes; and nodes lagged at $\Gamma(\tau_0^{\rm net}) = \{(M-1)\tau_0^{\rm net} \pm \delta\tau_0^{\rm net}\}_M$, where $M = 1, 2, 3, 4$ and $\delta\tau_0^{\rm net} = 3$, for (b) $\tau_0^{\rm net} = -7$, (c) $\tau_0^{\rm net} = -12$, and (d) $\tau_0^{\rm net} = -17$. (e) Measure of prediction error via average of NMSEs for 20-network model at $\mu = 1.1$, over 20 MG different time series. The red constant lines shows prediction performances for the networks including all nodes. The black curves show prediction performances only considering nodes contained in $\Gamma(\tau_0^{\rm net})$. (f) Percentage of divergence showing the average of realizations for which NMSE $> 1$, and (g) average of nodes contained in $\Gamma(\tau_0^{\rm net})$.

Takens framework. Non-Takens coordinates could negatively impact network prediction performances. Therefore, we introduce a methodology to exclude such additional delay dimensions from our predictor. As a starting point, we suppress nodes with specific CCA-lag positions during the training step of the output layer, such that they will not be available to the readout matrix but still take part in the rRNN's state evolution. To that end, we select nodes for which their CCA-lag positions are within windows of width $\delta\tau_0^{\rm net}$, centered at integer multiples of $\tau_0^{\rm net}$, where $\tau_0^{\rm net}$ represents the time lag used for delay reconstructions in the Takens' scheme. The windows width $\delta\tau_0^{\rm net}$ defines a time-lag uncertainty associated to the identified rRNNs' delay coordinates. All nodes with CCA-lag positions not inside the set of $(n\tau_0^{\rm net} \pm \delta\tau_0^{\rm net})$, $n \in \mathbb{Z}$, will not be available to the readout layer.

To illustrate our method and its effect, we show the non-filtered CCA of the rRNN when driven by the MG signal and with bifurcation parameter $\mu = 1.1$ in Fig. 4(a). Using a constant CCA-windows width of $\delta\tau_0^{\rm net} = 3$, as the minimum uncertainty found associated with good performances, we now scan the position of these CCA windows by changing $\tau_0^{\rm net}$. We restrict the number of windows to $n \in \{-M, -M+1, \ldots, M\}$. For $\tau_0^{\rm net} \in \{-17, -12, -7\}$, in Figs. 4(b)–4(d), we show examples of filtered CCAs where just rRNN nodes available for the readout layer are present.

Based on such movable CCA filters, we can estimate the relevance of different CCA lags on the rRNN's capacity to predict a particular temporal sequence. We define 20 network models via different initializations of $\{W, W^{\rm in}, W^{\rm off}\}$, and for each model we obtain the NMSE for predicting the value of 20 different MG sequences at 300 time steps into the future. In Fig. 4(e), we show the resulting NMSE, averaging over all system combinations and for $-25 \leqslant \tau_0^{\rm net} \leqslant -1$. Here, the average NMSE is given by the solid line, and the standard deviation (stdev) interval by the dashed black lines. In Fig. 4(f) we show the percentage of rRNN's for which prediction diverged from the target, i.e., NMSE $> 1$. Figure 4(g) shows how many nodes are available to the network's output layer. The constant red curves present in all panels show averaged performances obtained for the nonfiltered rRNN again with the stdev interval given by the dashed red lines.

Restricting the system's output based on the CCA-filter windows has a strong and systematic impact onto the rRNN's prediction performance. Performance is optimized for very characteristic filter positions, i.e., for $\tau_0^{\rm net} \in [-16, -5]$. For $\tau_0^{\rm net} \in [-12, -11]$ the embedding available to the network's output closely corresponds with the lags of the original Takens attractor embedding. The performance achieved by setting $\tau_0^{\rm net} \simeq \tau_0$ in our approach slightly reduces the NMSE by $\sim$0.09, even though the system has in average significantly less network nodes available to the output layer ($\sim$500) [see Fig. 4(g)]. For $\tau_0^{\rm net} = -7$, the performance is higher by one order of magnitude, accessed by using approximately the same initial set of nodes available to the output ($\sim$1000)

[see Fig. 4(g)]. Thus, we are therefore able to identify node families that show attractor embedding features in the network's space based on their CCA lag. For filters based on $\tau_0^{\text{net}} \in [-16, -5]$ the prediction performances are either coincident or better than the nonfiltered CCA case. This result is in agreement with Fig. 2, where we show that lags in such interval seem to still unfold the object in the state space.

Finally, some further aspects are seen in our data. For $\tau_0^{\text{net}} > -4$, the CCA-filter windows overlap and nodes with a lag inside such positions of overlap are assigned to multiple windows. This artificially increases the number of nodes available to the system's output beyond $m = 1000$. The resulting NMSE strongly increases beyond the one for the original rRNN. We attribute this characteristic to overfitting during learning, where there are just repetitions of the same few delay coordinates. In summary, it is noticeable that a link between identified attractorlike features and prediction performance can be established. In the following, a quantitative comparison between those features and the original input attractor is presented.

### B. Characteristics of the extracted features

Our previous analysis identifies and harnesses meaningful features related to good prediction performances. As this was realized by randomly connected networks, attractor reconstruction was achieved by randomly mapping the originally 1D input signal onto the high-dimensional rRNN's space. Such random mapping is treated by the framework of random projections theory (RPT) [46–50]. An active area of study within this field treats the question if original input data are randomly mapped onto the dimensions of the projection space, the structural damage to the original object is minimized.

To determine the degree of potential structural distortions to the original input attractor after random mapping onto the network's high-dimensional space according to $\mathbf{y}_n^{\text{in}} \to \varphi(\mathbf{y}_n^{\text{in}})$, we measure distances between consecutive states (interstate distances) of the original $||\mathbf{y}_{n+1}^{\text{in}} - \mathbf{y}_n^{\text{in}}||$ and the projected $||\varphi(\mathbf{y}_{n+1}^{\text{in}}) - \varphi(\mathbf{y}_n^{\text{in}})||$ objects. Following these steps, we take inspiration from RPT extended to nonlinear mapping [51], and develop a similar study which allows us to compare such original and projected objects. Under such mapping, we find that the interstate distances of the original attractor $\|\mathbf{y}_{n+1}^{\text{in}} - \mathbf{y}_n^{\text{in}}\|$ and of the projected attractors $\|\varphi(\mathbf{y}_{n+1}^{\text{in}}) - \varphi(\mathbf{y}_n^{\text{in}})\|$ in the rRNN's space are bound to the range $[(1-\epsilon_1), (1+\epsilon_2)]$ according to

$$(1-\epsilon_1)\left\|\mathbf{y}_{n+1}^{\text{in}} - \mathbf{y}_n^{\text{in}}\right\| \leqslant \left\|\varphi\left(\mathbf{y}_{n+1}^{\text{in}}\right) - \varphi\left(\mathbf{y}_n^{\text{in}}\right)\right\| \leqslant (1+\epsilon_2)\left\|\mathbf{y}_{n+1}^{\text{in}} - \mathbf{y}_n^{\text{in}}\right\|, \tag{5}$$

where $\{\varphi(\mathbf{y}_n^{\text{in}}), \varphi(\mathbf{y}_{n+1}^{\text{in}})\}$ are states built by rRNN node responses, where those node responses are assigned to an embedding dimension using the CCA. Details in the estimation of $\{\epsilon_1, \epsilon_2\}$ are added in Appendix.

As $\tau_0^{\text{net}} = -12$ was found to be associated to good prediction performances using approximately half of the initial set of nodes, we show in Fig. 5(a) the estimation of $\{\epsilon_1(\mu), \epsilon_2(\mu)\}$ for the rRNN for which nodes' CCA-lag positions that are within windows of width $\delta\tau_0^{\text{net}} = 3$, centered at integer multiples of $\tau_0^{\text{net}} = -12$. Here, we present the average of the statistical distribution that includes 20 network models and the prediction of 20 different MG time series at 300 time steps each into the future. In Fig. 5(b), we schematically illustrate the relevant geometrical properties of the attractors mapped onto the rRNN's space. Such study considered $\tau_0 = -12$ with which we obtain the set of coordinates $\mathbf{y}_n = \{y_n, y_{n-12}, y_{n-24}, y_{n-36}\}$ that we use to unfold the state-space object, where $\mathbf{y}_n^{\text{in}} = \mathbf{y}_n$.

The consequence of an increasing $\mu$ in Fig. 5(a) can be explained with the graphic representation of the limits shown by Fig. 5(b). Here, we illustrate the three general cases (I, II, III) connected to their corresponding ranges in $\mu$ in Fig. 5(a). The evolution of the original attractor's trajectory is illustrated along three black curves sampled at the positions of the big black dots. Such curves represent random portions of the attractor's trajectory in a 2D space $[\mathbf{y}_n^{\text{in}}, \mathbf{y}_{n+\tau_0}^{\text{in}}]$, and the set $\{\mathbf{y}_n^{\text{in}}, \mathbf{y}_{n+1}^{\text{in}}, \mathbf{y}_{n+2}^{\text{in}}\}$ contains three attractor states from time step $n$ to $n+2$. Gray dots are network's neighbor states to the $\mathbf{y}_{n+1}^{\text{in}}$, for instance. The first case (I) corresponds to neighbors which form a dense cloud of samples, highlighted in brown color, that are arranged closely around the original sample since $\epsilon_1 \simeq 1$ and $\epsilon_2 < 1$. The neighbor samples insufficiently enhance diversity in feature extraction and the network cannot predict the system's future evolution. This is confirmed by Figs. 5(c) and 5(d), which show bad prediction performance and unity divergence: the rRNN cannot predict the system.

Case (II) includes the values of $\mu$ where $\epsilon_1 \leqslant 1$ and $\epsilon_2 \geqslant 1$. Within this parameter range, the system's prediction performance strongly increases until reaching the lowest prediction error. Our analysis reveals the following mechanism behind this improvement: according to $\epsilon_2$, the maximum interstate distance possible inside the rRNN's space is twice the interstate distance of the original trajectory. As a consequence, the rRNN samples neighbors to state $\mathbf{y}_{n+1}^{\text{in}}$. Hence, as the state neighborhood is broader, there now is a sufficient random scanning of the attractor's vicinity, such that the network can use the different features to solve prediction. The network can therefore use the projected objects to predict, which is confirmed by good performance according to Figs. 5(c) and 5(d).

The last case (III) appears for $\mu > 1.3$, where all approximated distances of the embedded attractor are much larger than the original distance. The rRNN's autonomous dynamics therefore enlarge the sampling distance such that no dense nearest neighbors are anymore available for prediction. The result is the distortion caused by the autonomous network's dynamics typically found to be chaotic for $\mu \gtrsim 1.4$, already identified in our previous work [9]. Consequently, the network's folding property distorts the projected features. Therefore, $\epsilon_1$ becomes undefined, meaning that information about the structure of the embedded trajectory is lost. As a consequence, prediction performance strongly reduces [see Figs. 5(c) and 5(d)].

The process described in this section allowed us to identify and use relevant features that benefit good long-term prediction performances with a downsized rRNN. Additionally, we found that the neighborhood generated by the interstate distances of such features has an meaningful impact on the network's ability to predict at all. At this point, we show how

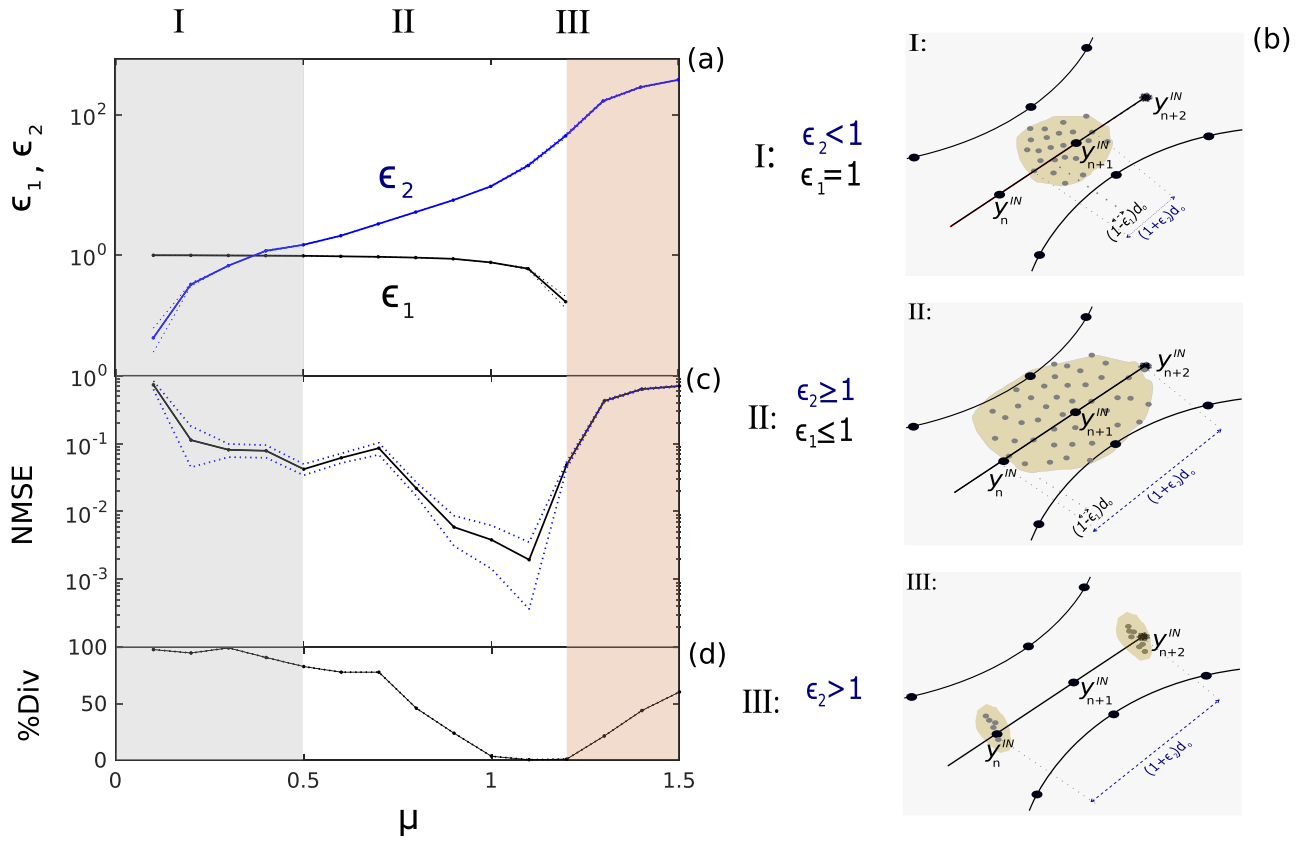


FIG. 5. (a) Maximum and minimum average boundaries $\{\epsilon_1, \epsilon_2\}$ as function of $\mu$, where $\tau_0^{\text{net}} = -12$ and $\delta\tau_0^{\text{net}} = 3$. (b) Illustrative scheme showing the evolution of $\{\epsilon_1, \epsilon_2\}$ for three different cases connected to their corresponding ranges in $\mu$. (c) Measure of prediction error via average of NMSEs for 20-network model, over 20 MG different time series; and (d) percentage of divergence showing the average of realizations for which NMSE $> 1$ as functions of $\mu$.

to simplify even more our process and package all the above-described steps that allowed us to utilize features related to good prediction performances.

## IV. HYBRID TAKENS-INSPIRED RANDOM RECURRENT NETWORKS

In this section, we directly exploit our newly gained insight and introduce a modified version of the classical random neural network for time-series prediction. Here, we design a system which aims to only take into account such Takens dimensions that we found to be relevant for prediction. As it was previously described, actions provided by the nodes of the rRNN can be interpreted in the light of delay embedding. We consequently modify the classical rRNN by including a Takens-inspired external memory:

$$\mathbf{x}_{n+1} = f_{NL}\big(\mu W \cdot \mathbf{x}_n + \alpha W^{\text{in}} \cdot y^{\text{in}}_{n+1} + W^{\text{off}} \cdot b\big), \tag{6}$$

$$y^{\text{out}}_{n+1} = W^{\text{out}} \cdot (\mathbf{x}_{n+1}, \mathbf{x}_{n+1+\tau_T}), \tag{7}$$

where $\mathbf{x}_{n+1+\tau_T}$ is a delayed term added to the output layer [see Fig. 6(a)]. All elements of the reservoir layer have been copied and then time shifted by a delay term $\tau_T$ that could be the Takens embedding delay $\tau_0$. This process allows us to add virtual nodes to our network, which are distributed in the delay lines. This Takens rRNN (TrRNN) combines nonvolatile external memory (virtual nodes) with a neural network (real nodes) and therefore shares functional features of the recently introduced hybrid computing concept [52]. Yet, our concept makes any additional costly optimization unnecessary.

We start our analysis by identifying the embedding delay related to the best prediction performance. Here, we fix $\mu = 0.1$, and modify then the delay term $\tau_T \in [-20, -1]$. For $\mu = 0.1$, the delay coordinates found in the network's space only span approximately two Takens embedding dimensions of the MG system with delays $\{2\tau_0, 0\}$, when $\tau_0 = -12$ [see Fig. 6(b)]. Furthermore, most node responses are distributed along the columns centered in lags $\{2\tau_0, 0\}$. Consequently, as shown in Figs. 5(c) and 5(d), the prediction performance is almost the lowest possible due to insufficient dimensionality to get attractorlike features. Additionally, we set the number of nodes to 350, which is 70% of the nodes that the best CCA-windows filtered rRNN had to disposal (see Sec. III A); and it uses 35% of the nodes used by the nonfiltered classical rRNN.

According to Figs. 7(a) and 7(b), the best averaged prediction performance, for 20 network models over 20 MG different time series at 300 time steps into the future, is found for $\tau_T = -12$, belonging to an interval $\tau_T \in [-13, -10]$ with the lowest NMSE values and divergent rates. The delay $\tau_T$ therefore agrees with the one identified for Takens attractor embedding $\tau_0$ and is almost identical to one of the lags $\tau_0^{\text{net}}$

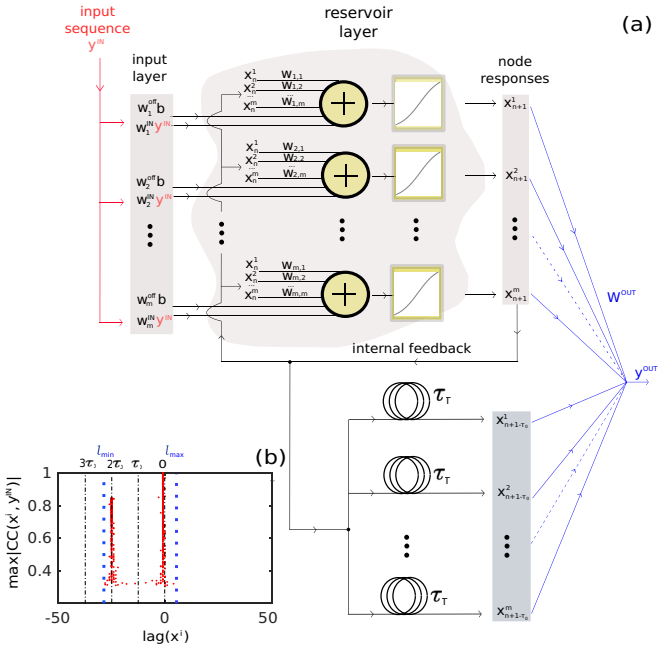


FIG. 6. (a) Schematic illustration of a TrRNN. Information enters the system via the input, a recurrently connected network forms a neural network. Based on our theory, we propose a simplistic extension to the system via an external delay memory $\tau_T$. (b) Maximum absolute value of the cross-correlation function between node responses $x^i$ and input signal $y^{\text{in}}$ for $\mu = 0.1$, in which each of the 1000 reservoir nodes is considered with a red dot. Labels $\{l_{\min}, l_{\max}\}$ remark the lags' limits for the distributions of nodes, and $\tau_0 = -12$.

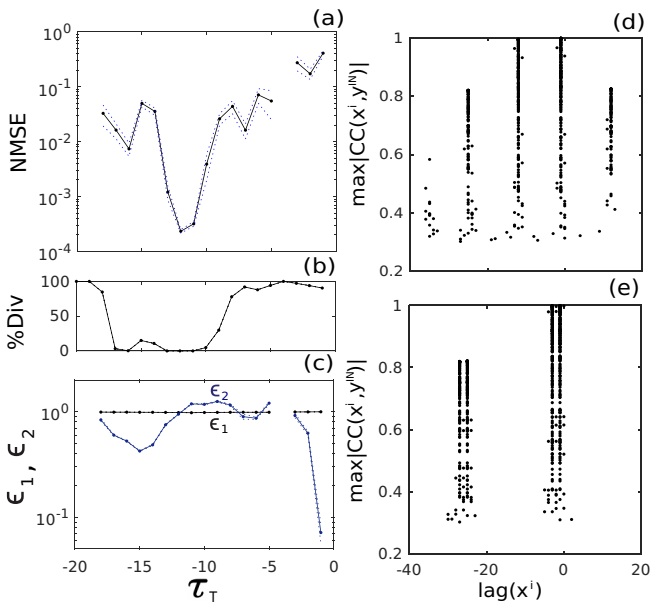


FIG. 7. (a) Average of prediction performances NMSEs for 20-network model, over 20 MG different time series using a TrRNN with 350 nodes at $\mu = 0.1$ and the delay term $\tau_T \in [-20, 0]$. (b) Percentage of divergence showing the average of realizations for which NMSE $> 1$. (c) Maximum and minimum average boundaries $\{\epsilon_1, \epsilon_2\}$ as function of $\tau_T$. Maximum absolute value of the cross-correlation function between node responses and input signal for (d) $\tau_T = -12$ and (e) $\tau_T = -3$.

found optimal in the CCA-window filtering. Furthermore, here the system only has as many CCA windows at its disposal as dimensions required to embed the MG attractor. This removes the disambiguity present in the CCA-window filtering analysis, where a time-lag uncertainty was required to exclude possible scattered delay coordinates. Consequently, the optimum performance is found only for a TrRNN embedding exactly along lags according to Takens embedding. In comparison to the classical rRNN, our TrRNN achieves the same performance, simultaneously reducing the amount of nodes in the network layer from 1000 to 350 in the output layer. Compared to the pristine rRNN, we obtain one order of magnitude better performance with a network three times smaller.

Figure 7(c) shows the estimation of $\{\epsilon_1, \epsilon_2\}$ with the variation of $\tau_T$. As it can be seen, $\epsilon_2 \geqslant 1$ is associated to good prediction performances, found for $\tau_T \in [-13, -10]$. This result agrees with the results provided by the classical random network in Sec. III B. The CCA for $\tau_T = -12$ is shown by Fig. 7(d), where we can find the set of nodes with all delay coordinates required to fully reconstruct the MG attractor. In the cases where prediction was not possible, the CCA identifies the nonadequacy of the rRNN delay embedding as the reason [see Fig. 7(e) for $\tau_T = -3$].

### A. Application: Control an arrhythmic neuronal model

We directly utilize our TrRNN as a part of an efficient feedback control mechanism in an arrhythmic excitable system. We task the TrRNN to aid stabilizing a system which models the firing behavior of a noise-driven neuron. It consists in the FitzHugh-Nagumo (FHN) neuronal model [53,54]

$$\epsilon \frac{dv(t)}{dt} = v(t)[v(t) - g][1 - v(t)] - w + I + \xi(t), \quad (8)$$

$$\frac{dw(t)}{dt} = v(t) - Dw(t) - H, \quad (9)$$

where $v(t)$ and $w(t)$ are voltage and recovery variables. $I = 0.3$ is an activation signal, $\xi$ is Gaussian white noise with zero mean and standard deviation $\sim 0.02$, $\epsilon = 0.005$, $g = 0.5$, $D = 1.0$, and $H = 0.15$. These equations have been solved by the Euler-Maruyama algorithm for stochastic differential equation's integration. In its resting state, the neuron's membrane potential is slightly negative. Once the membrane voltage $v(t)$ is sufficiently depolarized through an external stimuli, the neuron spikes due to the rise of the action potential [55,56]. The time between consecutive spikes is defined as *interspike intervals* (ISIs). In Fig. 8(a), random ISI's evolution is shown from step 0 to 7464, where such trajectory exhibits the nonregular neural spiking of the FHN neuronal model.

We aim to control this random spiking behavior of the FHN neuronal model by proportional perturbation feedback (PPF) method [57] and by using either a TrRNN or a rRNN. The PPF method consists in the application of perturbations to locate the system's unstable fixed point onto a stable trajectory [57,58]. This method is used to fit instabilities in the FHN neuronal model through the design of a control equation. In our case, the goal of using the PPF method is to build a control subsystem, which applies an external stimuli to trigger spiking and reduce the degree of chaos. Once the control is activated,

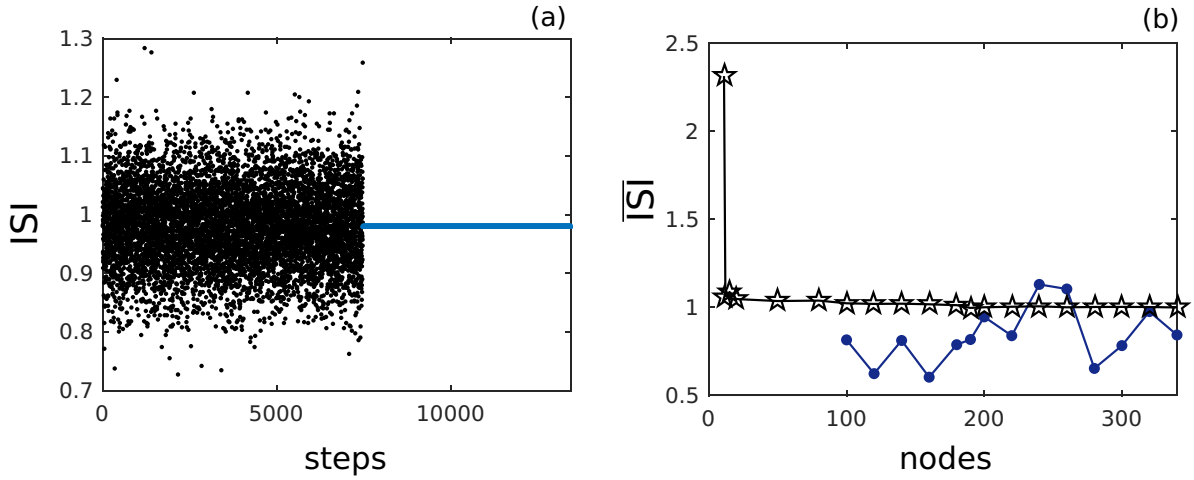


FIG. 8. (a) Interspike intervals (ISIs) of an arrhythmic excitable system comparable to a heart. Stabilization of the system based on TrRNN with only 12 network nodes. (b) Comparison between the stabilized mean of the TrRNN (black curve with stars) and a classical rRNN (blue curve with dots).

the resulting control signal is injected via $I$ through a train of pulses which take discrete values.

In our approach, the past information provided by voltage $v(t)$ in the FHN model is used to determine two things: (i) the parameters to design the control equation, and (ii) training parameters for rRNN ($\mu = 1.1$) and TrRNN [$\mu = 0.1$, and $\tau_T = -166$ obtained via the ACF minimum of $v(t)$ as in the Takens' scheme] to predict future values of $v(t)$. The predicted $v(t)$ is used to calculate the full control signal with which we stabilize the neuron's spiking activity. Such stabilization will cause the cease of random ISIs and then the beginning of regular spiking, where each recorded ISI should follow a constant evolution. Our methodology allows us to replace the quantity under control $v(t)$ by a predicted signal generated by either a rRNN or a TrRNN. This replacement is a typical practice in control theory as it is related with the replacement of *sensors* in an exemplary control system.

To train the network, we inject $1\times10^5$ values and we let the network run freely for other $4\times10^6$ steps, allowing us to stabilize 5619 ISI points. We then evaluate the quality of the stabilization for networks ranging from 11 to 340 nodes. In Fig. 8(a), we show how random ISIs are evolving from step 0 to 7464. Then, once the control is activated at step 7465, the set of blue dots along a constant line shows how the TrRNN's output, by means of the control equation, can stabilize the ISI activity. As it can be seen, the network can control the random ISI starting from step 7465. The excellent stabilization was achieved with a TrRNN containing only 12 nodes.

Figure 8(b) shows the full comparison between rRNN and TrRNN. The mean value of ISI is calculated for the different sizes of rRNN and TrRNN and then normalized by the mean value of the random ISI. The TrRNN starts inferring the inner dynamics of the FHN system for an extremely small network containing just 12 nodes, from which point on it is always capable to correctly stabilize the ISI. In contrast, the classical rRNN does not predict at all until its architecture has at least 80 nodes, but performance remains poor in comparison with TrRNN. For 200 nodes the rRNN starts predicting the dynamic of the FHN system more or less correctly, allowing the control signal to fully stabilize the ISI. Yet, for more than 200 nodes the good performance still can fluctuate, even significantly dropping again. This indicates that in general the stabilization via a classical rRNN is not robust. Furthermore, with the TrRNN one can reduce the number of nodes to 15 times less than the classical rRNN. This stark difference in performance highlights (i) the difficulty of the task, and (ii) the excellent efficiency that the addition of a simple, linear delay term adjusted to the Takens embedding delay brings to the system. Our TrRNN, therefore, is not only an interesting ANN concept for the prediction of complex systems, it also helps with the downsize of random recurrent networks' hidden layers while preserving good prediction performances.

## V. COMPARISON WITH CLASSICAL STATE-SPACE PREDICTION

Up to now, we have been describing a method for downsizing the hidden layer of rRNNs which can be summarized by the same steps that should be followed if we attempt to solve continuous-time chaotic signal prediction in the state-space framework. This framework can be divided according to three fundamental aspects [22,25,26]: (i) insufficient information to represent the complete state-space trajectory of the chaotic system: this problem originates from the fact that in many cases one does not have access to all state-space dimensions. In this case, a reconstruction of the dynamics along the chaotic system's missing degrees of freedom is required. The knowledge of all dimensions allows us to design predictors based on full state-space trajectories. (ii) The second problem is related to the sampling resolution: all information that is acquired, be it from simulations or from experiments, comes with a particular resolution. To minimize the divergence between a prediction and the correct value, the sampling resolution has to be maximized. This is of particular importance for prediction of chaotic systems as these by definition show exponential divergence. (iii) For deterministic chaotic systems, future states of a given trajectory can in principle be approximated from the exact knowledge of the present state. Therefore, the

final step toward prediction is approximating the underlying deterministic law ruling the dynamical system's evolution.

By the same token, step (i) is fulfilled by the random mapping which takes place in the high-dimensional space of the network. Here, RPT supports the fact that the original input data are randomly mapped onto the dimensions of the projection space, and then the structural damage to the original object is minimized. Step (ii) is fulfilled by the analysis made in Sec. III B, where the sampling resolution is maximized to cover the region between the states $\mathbf{y}^{\text{in}}_n$ and $\mathbf{y}^{\text{in}}_{n+1}$. Finally, step (iii) relates to the training itself of the rRNN, where $W^{\text{out}}$ has to be determined via regression.

## VI. CONCLUSION

We have introduced a unconventional method of rRNNs analysis which demonstrates how prediction is potentially achieved in high-dimensional nonlinear dynamical systems. Random recurrent networks and prediction of a specific signal can consequently be described via a common methodology. Quantifying measures such as the memory related cross-correlation analysis and the feature extraction are quantitatively interpretable. We therefore significantly extend the toolkit previously available for random neural network analysis. Tools developed in the paper might be comparable to the utilization of the t-SNE [59] technique for analyzing ANNs during a classification task.

Our scheme has numerous practical implications. The most direct is motivating the development and analysis of new learning strategies. Furthermore, we already designed a hybrid processor which includes both virtual and real nodes that efficiently predicts via *a priori* defined external memory access rules. This approach allows us to improve the design of our neural network in order to reduce the number of nodes and connections required to solve prediction.

## ACKNOWLEDGMENT

Funding for B.A.M. and B.J.S. was provided by the 2019 Queen's postdoctoral fellowship fund, the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant and the Queen's Research Initiation Grant (RIG).

## APPENDIX: ESTIMATION OF $\{\epsilon_1, \epsilon_2\}$

Each state in Takens space is described by $M$ delay coordinates

$$\mathbf{y}^{\text{in}}_n = \left(y^{\text{in}}_n, y^{\text{in}}_{n+\tau_0}, \ldots, y^{\text{in}}_{n+(M-1)\tau_0}\right), \tag{A1}$$

$$\mathbf{y}^{\text{in}}_{n+1} = \left(y^{\text{in}}_{n+1}, y^{\text{in}}_{(n+1)+\tau_0}, \ldots, y^{\text{in}}_{(n+1)+(M-1)\tau_0}\right). \tag{A2}$$

The second step is to define the corresponding two arbitrary consecutive states $\{\varphi(\mathbf{y}^{\text{in}}_n),\ \varphi(\mathbf{y}^{\text{in}}_{n+1})\} \in \mathbb{R}^h$, where $h$ depends on $\mu$. The value of $h$ is determined from the CCA, where we approximately assign the mapped objects dimensionality to the number of elements found in the interval $[l_{\min}, l_{\max}]$ for each $\mu$ [see Figs. 2(b) and 2(c)]. In order to construct those projected states, we use all the delay coordinates provided by the network, i.e., the full range $[l_{\min}, l_{\max}]$ for each value of $\mu$, as follows:

$$\varphi\left(\mathbf{y}^{\text{in}}_n\right) = \left[\varphi_{l_1}\left(\mathbf{y}^{\text{in}}_n\right), \varphi_{l_2}\left(\mathbf{y}^{\text{in}}_n\right), \ldots, \varphi_{l_h}\left(\mathbf{y}^{\text{in}}_n\right)\right], \tag{A3}$$

$$\varphi\left(\mathbf{y}^{\text{in}}_{n+1}\right) = \left[\varphi_{l_1}\left(\mathbf{y}^{\text{in}}_{n+1}\right), \varphi_{l_2}\left(\mathbf{y}^{\text{in}}_{n+1}\right), \ldots, \varphi_{l_h}\left(\mathbf{y}^{\text{in}}_{n+1}\right)\right], \tag{A4}$$

where $\{\varphi_{l_1}(\mathbf{y}^{\text{in}}_n), \varphi_{l_2}(\mathbf{y}^{\text{in}}_n), \ldots\}$ are node responses lagged at $[l_{\min}, l_{\max}]$. The size of the interval $[l_{\min}, l_{\max}]$ depends on the value of $\mu$, as it was shown by Figs. 2(b) and 2(c), where we find a broader distribution of delay coordinates for higher values of $\mu$.

The *interstate distances* $\|\mathbf{y}^{\text{in}}_{n+1} - \mathbf{y}^{\text{in}}_n\|$ and $\|\varphi(\mathbf{y}^{\text{in}}_{n+1}) - \varphi(\mathbf{y}^{\text{in}}_n)\|$ have to be bounded in the interval $[(1-\epsilon_1), (1+\epsilon_2)]$ according to

$$\frac{\left\|\varphi\left(\mathbf{y}^{\text{in}}_{n+1}\right) - \varphi\left(\mathbf{y}^{\text{in}}_n\right)\right\|}{\left\|\mathbf{y}^{\text{in}}_{n+1} - \mathbf{y}^{\text{in}}_n\right\|} \in [(1-\epsilon_1), (1+\epsilon_2)]. \tag{A5}$$

Under these conditions, we can claim that the transformation by the rRNN agrees with a nonlinear random projections. Estimating limits $\{\epsilon_1, \epsilon_2\}$ requires to find the inferior $\epsilon_{\min}$, and superior $\epsilon_{\max}$ interstate distance limits:

$$\frac{\left\|\varphi\left(\mathbf{y}^{\text{in}}_{n+1}\right) - \varphi\left(\mathbf{y}^{\text{in}}_n\right)\right\|_{\min}}{\left\|\mathbf{y}^{\text{in}}_{n+1} - \mathbf{y}^{\text{in}}_n\right\|} = \epsilon_{\min}; \tag{A6}$$

$$\frac{\left\|\varphi\left(\mathbf{y}^{\text{in}}_{n+1}\right) - \varphi\left(\mathbf{y}^{\text{in}}_n\right)\right\|_{\max}}{\left\|\mathbf{y}^{\text{in}}_{n+1} - \mathbf{y}^{\text{in}}_n\right\|} = \epsilon_{\max}, \tag{A7}$$

where $\epsilon_1$ and $\epsilon_2$ are calculated by isolating these constants from $\epsilon_{\min} = (1-\epsilon_1)$ and $\epsilon_{\max} = (1+\epsilon_2)$. These limits contain information about the minimum and maximum distortions that we can find in order to get the best neighbors in the rRNN. $\|\varphi(\mathbf{y}^{\text{in}}_{n+1}) - \varphi(\mathbf{y}^{\text{in}}_n)\|_{\min}$ and $\|\varphi(\mathbf{y}^{\text{in}}_{n+1}) - \varphi(\mathbf{y}^{\text{in}}_n)\|_{\max}$ are calculated by using Euclidean distance under minimum and maximum norms

$$\left\|\varphi\left(\mathbf{y}^{\text{in}}_{n+1}\right) - \varphi\left(\mathbf{y}^{\text{in}}_n\right)\right\|_{\min} = \left(\sum_{l_g=l_{\min}}^{l_{\max}} \left[\varphi_{l_g}\left(\mathbf{y}^{\text{in}}_{n+1}\right) - \varphi_{l_g}\left(\mathbf{y}^{\text{in}}_n\right)\right]^2_{\min}\right)^{1/2}, \tag{A8}$$

$$\left\|\varphi\left(\mathbf{y}^{\text{in}}_{n+1}\right) - \varphi\left(\mathbf{y}^{\text{in}}_n\right)\right\|_{\max} = \left(\sum_{l_g=l_{\min}}^{l_{\max}} \left[\varphi_{l_g}\left(\mathbf{y}^{\text{in}}_{n+1}\right) - \varphi_{l_g}\left(\mathbf{y}^{\text{in}}_n\right)\right]^2_{\max}\right)^{1/2}, \tag{A9}$$

where $\varphi_{l_g}(\mathbf{y}^{\text{in}}_n)$ are node responses lagged at $l_g \in [l_{\min}, l_{\max}]$, $\forall\ g = 1, 2, \ldots, h$.

Here, we therefore identify the smallest and largest distances $[\varphi_{l_g}(\mathbf{y}^{\text{in}}_{n+1}) - \varphi_{l_g}(\mathbf{y}^{\text{in}}_n)]_{\min,\max}$ along each delay coordinate. Then, it is true that these smallest and largest distances bound the Euclidean distance of these. Finally, we determine $\|\mathbf{y}^{\text{in}}_{n+1} - \mathbf{y}^{\text{in}}_n\|$ via

$$\left\|\mathbf{y}^{\text{in}}_{n+1} - \mathbf{y}^{\text{in}}_n\right\| = \sqrt{\left(y^{\text{in}}_{n+1} - y^{\text{in}}_n\right)^2 + \cdots + \left(y^{\text{in}}_{(n+1)+(M-1)\tau_0} - y^{\text{in}}_{n+(M-1)\tau_0}\right)^2}. \tag{A10}$$

[1] W. Maass, Searching for principles of brain computation, Curr. Opin. Behav. Sci. **11**, 81 (2016).

[2] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).

[3] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. **31**, 855 (2009).

[4] A. Graves, A. R. Mohamed, and G. Hinton, Speech recognition with deep recurrent neural networks, in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (IEEE, Piscataway, NJ, 2013), p. 6645.

[5] H. Sak, A. Senior, and F. Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling, in *Proceedings of the 14th Annual Conference of the International Speech Communication Association, Interspeech 2013* (ICSA, Baixas, France, 2014), p. 6645.

[6] X. Li and X. Wu, Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition, in *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, Piscataway, NJ, 2015), p. 4520.

[7] C. G. Langton, Computation at the edge of chaos: Phase transitions and emergent computation, Phys. D (Amsterdam) **42**, 12 (1990).

[8] T. Natschlager, N. Bertschinger, and R. Legenstein, At the edge of chaos: Realtime computations and self-organized criticality in recurrent neural networks, in *Advances in Neural Information Processing Systems* (NIPS, San Diego, 2005).

[9] B. A. Marquez, L. Larger, M. Jacquot, Y. K. Chembo, and D. Brunner, Dynamical complexity and computation in recurrent neural networks beyond their fixed point, Sci. Rep. **8**, 3319 (2018).

[10] A. M. Bruckstein, D. L. Donoho, and M. Elad, From sparse solutions of systems of equations to sparse modeling of signals and images, SIAM Rev. **51**, 34 (2009).

[11] S. Ganguli and H. Sompolinsky, Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis, Annu. Rev. Neurosci. **35**, 485 (2012).

[12] B. Babadi and H. Sompolinsky, Sparseness and expansion in sensory representations, Neuron **83**, 1213 (2014).

[13] M. Schottdorf, W. Keil, D. Coppola, L. E. White, and F. Wolf, Random wiring, ganglion cell mosaics, and the functional architecture of the visual cortex, PLoS Comput. Biol. **11**, e1004602 (2015).

[14] W. Maass, T. Natschlaeger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Comput. **14**, 2531 (2002).

[15] P. Antonik, M. Haelterman, and S. Massar, Brain-Inspired Photonic Signal Processor for Generating Periodic Patterns and Emulating Chaotic Systems, Phys. Rev. Appl. **7**, 054014 (2017).

[16] E. N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. **20**, 130 (1963).

[17] M. C. Mackey and L. Glass, Oscillation and chaos in physiological control systems, Science **197**, 287 (1977).

[18] E. Ott, *Chaos in Dynamical Systems* (Cambridge University Press, Cambridge, 1993).

[19] R. FitzHugh, Mathematical models of threshold phenomena in the nerve membrane, Bull. Math. Biophys. **17**, 257 (1955).

[20] R. FitzHugh, Impulses and physiological states in theoretical models of nerve membrane, Biophys. J. **1**, 445 (1961).

[21] J. Nagumo, S. Arimoto, and S. Yoshizawa, An active pulse transmission line simulating nerve axon, Proc. IRE **50**, 2061 (1962).

[22] A. S. Weigend and N. A. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past* (Westview Press, Boulder, CO, 1993).

[23] M. Zarandi, M. Zarinbal, N. Ghanbari, and I. Turksen, A new fuzzy functions model tuned by hybridizing imperialist competitive algorithm and simulated annealing. application: Stock price prediction, Inf. Sci. **222**, 213 (2013).

[24] A. Celikyilmaz and I. B. Turksen, Fuzzy functions with support vector machines, Inf. Sci. **177**, 5163 (2007).

[25] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis* (Cambridge University Press, Cambridge, 1997).

[26] J. D. Farmer and J. J. Sidorowich, Predicting Chaotic Time Series, Phys. Rev. Lett. **59**, 845 (1987).

[27] A. K. Alparslan, M. Sayar, and A. R. Atilgan, State-space prediction model for chaotic time series, Phys. Rev. E **58**, 2640 (1998).

[28] D. Kugiumtzis, O. C. Lingjærde, and N. Christophersen, Regularized local linear prediction of chaotic time series, Phys. D (Amsterdam) **112**, 344 (1998).

[29] E. Ott, C. Grebogi, and J. A. Yorke, Controlling Chaos, Phys. Rev. Lett. **64**, 1196 (1990).

[30] R. Rojas, *Neural Networks: A Systematic Introduction* (Springer, Berlin, 1996).

[31] K. Gurney, *An Introduction to Neural Networks* (CRC Press, Boca Raton, FL, 1997).

[32] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE **86**, 2278 (1998).

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (The MIT Press, Cambridge, MA, 2016).

[35] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Information processing using a single dynamical node as complex system, Nat. Commun. **2**, 468 (2011).

[36] B. A. Marquez, Complex signal embedding and photonic reservoir Computing in time series prediction. Neural and Evolutionary Computing [cs.NE]. Université Bourgogne Franche-Comté, 2018. English. NNT: 2018UBFCD042.

[37] B. A. Marquez, J. Suarez-Vargas, L. Larger, M. Jacquot, Y. K. Chembo, and D. Brunner, Embedding in neural networks: A-priori design of hybrid computers for prediction, in *Proceedings of the IEEE International Conference on Rebooting Computing (ICRC), Washington, DC* (IEEE, Piscataway, NJ, 2017), pp. 1–4.

[38] H. Jaeger, The echo state approach to analyzing and training recurrent neural networks, Fraunhofer Institute for Autonomous Intelligent Systems, Technical Report No. 148, 2001.

[39] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, Deepface: Closing the gap to human-level performance in face verification, in *Proceedings of the Conference on Computer*

*Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2014), p. 1701.

[40] G. V. Demidenko, V. A. Likhoshvai, and A. V. Mudrov, On the relationship between solutions of delay differential equations and infinite-dimensional systems of differential equations, Diff. Equ. **45**, 33 (2009).

[41] H. Whitney, Differentiable manifolds, Ann. Math. **37**, 645 (1936).

[42] F. Takens, Detecting strange attractors in turbulence, *Dynamical Systems and Turbulence, Lecture Notes in Mathematics* (Springer, Berlin, 1981).

[43] A. Pikovsky, M. Rosenblum, and J. Kurths, *Synchronization* (Cambridge University Press, Cambridge, 2001).

[44] J. P. Eckmann and D. Ruelle, Ergodic theory of chaos and strange attractors, Rev. Mod. Phys. **57**, 617 (1985).

[45] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, Geometry from a Time Series, Phys. Rev. Lett. **45**, 712 (1980).

[46] W. B. Johnson and J. Lindenstrauss, *Conference in Modern Analysis and Probability* (Contemporary Mathematics) (American Mathematical Society, Providence, RI, 1984).

[47] P. Indyk and R. Motwani, *Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality* (ACM Press, New York, 1998).

[48] P. Frankl and H. Maehara, The johnson-lindenstrauss lemma and the sphericity of some graphs, J. Combin. Theory B **44**, 355 (1988).

[49] S. Dasgupta and A. Gupta, An elementary proof of a theorem of Johnson and Lindenstrauss, Random Struct. Alg. **22**, 60 (2002).

[50] D. Sivakumar, Algorithmic derandomization using complexity theory, in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing, Canada* (ACM Press, New York, 2002).

[51] J. B. Tenenbaum, V. de Silva, and J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science **290**, 2319 (2000).

[52] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwinska, S. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. Badia, K. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, Hybrid computing using a neural network with dynamic external memory, Nature (London) **538**, 471 (2016).

[53] A. Longtin, Stochastic resonance in neuron models, J. Stat. Phys. **70**, 309 (1993).

[54] D. J. Christini and J. J. Collins, Controlling Nonchaotic Neuronal Noise using Chaos Control Techniques, Phys. Rev. Lett. **75**, 2782 (1995).

[55] R. M. Enoka, *Neuromechanics of Human Movement* (Human Kinetics, Champaign, IL, 2015).

[56] B. Tirozzi, D. Bianchi, and E. Ferraro, *Introduction to Computational Neurobiology and Clustering* (World Scientific, Singapore, 2007).

[57] A. Garfinkel, M. L. Spano, W. L. Ditto, and J. N. Weiss, Controlling cardiac chaos, Science **257**, 1230 (1992).

[58] S J. Schiff, K. Jerger, D H. Duong, T. Chang, M. L. Spano, and W. L. Ditto, Controlling chaos in the brain, Nature (London) **370**, 615 (1994).

[59] L. J. P. van der Maaten and G. E. Hinton, Visualizing high-dimensional data using t-sne, J. Mach. Learn. Res. **9**, 2579 (2008).