Convolutional autoencoder and conditional random fields hybrid for predicting spatial-temporal chaos

Cite as: Chaos **29**, 123116 (2019); https://doi.org/10.1063/1.5124926
Submitted: 18 August 2019 . Accepted: 14 November 2019 . Published Online: 12 December 2019

S. Herzog, F. Wörgötter, and U. Parlitz

Convolutional autoencoder and conditional random fields hybrid for predicting spatial-temporal chaos

Cite as: Chaos **29**, 123116 (2019); doi: 10.1063/1.5124926
Submitted: 18 August 2019 · Accepted: 14 November 2019 ·
Published Online: 12 December 2019

S. Herzog,[1,2,3] F. Wörgötter,[2] and U. Parlitz[1,3,4,a)]

AFFILIATIONS

[1]Max Planck Institute for Dynamics and Self-Organization, Am Fassberg 17, 37077 Göttingen, Germany
[2]Third Institute of Physics and Bernstein Center for Computational Neuroscience, University of Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany
[3]DZHK (German Centre for Cardiovascular Research), partner site Göttingen, Robert-Koch-Str. 42a, 37075 Göttingen, Germany
[4]Institute for the Dynamics of Complex Systems, University of Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany

Note: This paper is part of the Focus Issue, "When Machine Learning Meets Complex Systems: Networks, Chaos and Nonlinear Dynamics."
[a)]**Electronic mail:** ulrich.parlitz@ds.mpg.de

ABSTRACT

We present an approach for data-driven prediction of high-dimensional chaotic time series generated by spatially-extended systems. The algorithm employs a convolutional autoencoder for dimension reduction and feature extraction combined with a probabilistic prediction scheme operating in the feature space, which consists of a conditional random field. The future evolution of the spatially-extended system is predicted using a feedback loop and iterated predictions. The excellent performance of this method is illustrated and evaluated using Lorenz-96 systems and Kuramoto-Sivashinsky equations of different size generating time series of different dimensionality and complexity.

With the advent of novel measurement devices like dense sensor networks or high speed high resolution cameras, large datasets are available to describe complex dynamics in spatially-extended systems. Using such spatiotemporal time series to forecast high-dimensional chaotic processes remains a challenge because methods from nonlinear time series analysis for reconstructing the dynamics using delay coordinates cannot be applied straightforwardly due to the extremely high dimension of the required delay embedding space. This difficulty can be overcome by nonlinear dimension reduction and a suitable factorization of the remaining multivariable conditional probability distributions providing forecasts based on historic data. This hybrid concept has been implemented using a convolutional autoencoder combined with an extended conditional random field modeling the time evolution. When applied to different very hyperchaotic spatiotemporal benchmark time series from the literature, this prediction method turns out to be very effective.

I. INTRODUCTION

Development of explicit mathematical models for complex physical phenomena, while desirable, often remains difficult or unfeasible. This is, in particular, true for high-dimensional, nonlinear dynamical systems for which many times adequate models based on first principles do not exist. As a result, it remains hard or impossible to predict the temporal development of such systems in any reliable way. This has led to attempts that use implicit data-driven descriptions of such systems, for example, employing artificial neural networks. For example, in the field of fluid dynamics, artificial neural networks[46] were used early on to introduce reduced-order models.[48] Over time, more and more approaches have been proposed for generating models from (training) data including autoregressive models[3] or adaptive fuzzy rule-based models.[2] The authors in Ref. 28 introduced a tree-structured, recurrent switching linear dynamical system. This probabilistically models the multiscale property of the observed data through a hierarchy of locally linear dynamic elements

that jointly approximate the global nonlinear dynamics of the system. Another current approach introduced in Ref. 15 treats time series prediction by using local state reconstructions,[30] dimension reduction, and nearest neighbor methods for local modeling. Furthermore, there are approaches that use echo state networks (ESNs) with very good results.[31] In Ref. 33, possible advantages of deep learning approaches over reservoir computing are shown, where a time-delay reservoir is outperformed by a deep learning network by at least an order of magnitude. Also, investigations were made with long short-term memory (LSTM) networks, which are prominent recurrent neural networks in the field of deep learning, where the LSTM uses the short-term history of the reduced-order variable to predict the state derivative and uses it for one-step prediction.[47] The approach presented below employs a convolutional autoencoder[4] for dimension reduction and feature extraction combined with a probabilistic prediction scheme to learn the dynamics of the considered systems only using recorded data.[11] Our goal is to show that, after learning, this model will describe data in different systems precisely and with a high level of predictability. This will be demonstrated using the Lorenz-96 system[25] as well as the Kuramoto-Sivashinsky equations,[21,40,41] where we can show that our approach substantially increases the prediction horizon compared to the state of the art.[31]

II. METHODS

In data-driven modeling, mathematical models are not based on first principles (e.g., Newton's laws, Maxwell's equations, etc.) but are directly derived from experimental data. In the following, we present a method[11] which combines a convolutional autoencoder (AE)[4] for feature extraction and dimension reduction with an extended version of conditional random fields (CRFs)[22] in order to model the properties of temporal sequences.

A. Artificial neural networks

The structure of artificial neural networks is divided into three parts. First, the input data are recorded in an input layer. Then, the recorded data are processed by an arbitrary number of hidden layers and in the third step this processed data is passed to an output layer, which maps the data to a descriptive space (e.g., classification[10,20]) or a generative space (e.g., super resolution[7]). Hereby, the hidden layers can have very different functions; for the algorithm presented in the following, the relevant parts are

1. Convolutional layers: Convolution of the input by a kernel sliding over the input. The numbers of rows and columns of the kernel are hyperparameters; in this work, they are set to be (3×3).
2. Batch normalization layer: Normalization of the activations of the previous layer during training and for each batch. Batch normalization allows the use of higher learning rates, being computationally more efficient, and also acts as a regularizer.[14]
3. Leaky ReLU[26] layer: Leaky version of a rectified linear unit (ReLU),[9] such that

$$\nu(x) = \begin{cases} \alpha x \text{ for } x < 0, \\ x \text{ for } x \geq 0. \end{cases}$$

4. Max pooling layer: Sample-based operation for discretization based on a kernel that slides over the input like the convolutional operator but only the maximum value of the kernel is passed to the next layer. Width and height of the kernel are hyperparameters (in this work 2×2). In contrast to the convolutional layer, a pooling layer is not trainable.
5. Dropout layer: Regularization method to prevent overfitting where during training some weights are set randomly to zero.[43] In this work, the probability of setting the weights to zero is 0.4.

A network is called feed forward network (FFN) if only the inputs from previous layers are used in a present layer. A FFN with convolutional layers is called a convolutional neural network (CNN) introduced in Ref. 23.

Let $X \in \mathcal{X} \subset \mathbb{R}^d$ and $Y \in \mathcal{Y} \subset \mathbb{R}$ be two random variables and $Y = f(X)$ for some unknown function f. Given samples $\{(x^{(r)}, y^{(r)})\}_{r=1,\dots,R}$ drawn from the joint distribution of X and Y, the goal is to find the mapping $\hat{f} : \mathcal{X} \mapsto \mathcal{Y}$ which minimizes the expected loss by a given loss function $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, leading to:

$$\hat{f} = \underset{f \in \mathcal{F}}{\arg\min} \mathbb{E}[\mathcal{L}(Y, f(X))], \tag{1}$$

where $\mathcal{F}$ is a restricted function space and $\mathbb{E}$ the expected value. CNNs are supposed to find solutions for Eq. (1).

A CNN with depth L consists of convolutional layers given by

$$x^{(l)} = h^{(l)}(x) = \nu\left(W^{(l)} * h^{(l-1)}(x) - b^{(l)}\right), \quad l = 1, \dots, L, \tag{2}$$

where $x^{(l)}$ is the output of the hidden layer, at layer l, $W^{(l)}$ is a convolutional tensor, ν an activation functions (in this work the leaky ReLU[26]) which acts componentwise, and $b^{(l)}$ is a vector of bias values for layer l. For $h^{(0)}(x) = x$ holds. $W^{(l)} * h^{(l-1)}(x)$ is the discrete convolution between W, a tensor of weights, and $h^{(l-1)}(x)$, a tensor of outputs from the $l-1$ layer.

A classical autoencoder[13] (AE) is a FFN, a convolutional autoencoder[4] a CNN, both try to replicate the input data as good as possible, while processing the data through a hourglass similar architecture, where in the middle of the network the data space is reduced drastically. This reduced space is called *feature space*. The compressing part of the network, often called encoding part, tries to reduce the complexity of the data as representatively as possible in the feature space. The second part, often called the decoding part, tries to reconstruct the data from the features such that the difference between input and output is minimal. AEs are self-supervised learning methods such that no annotated data are needed, changing Eq. (1) to

$$\hat{f} = \underset{f \in \mathcal{F}}{\arg\min} \mathbb{E}[\mathcal{L}(X, f(X))]. \tag{3}$$

The loss function $\mathcal{L} : \mathbb{R}^{|\theta|} \mapsto \mathbb{R}$,

$$\mathcal{L}(\theta) = \frac{1}{R} \sum_{r=1}^{R} |x^{(r)} - x_{(L)}^{(r)}(\theta)|, \tag{4}$$

quantifies the difference between input $x^{(r)}$ and output $x_L^{(r)}$ of the AE by means of a suitable metric (in this case, the mean absolute error), where L denotes the output layer and $x^{(r)}$ are the different inputs. The loss function $\mathcal{L}(\theta)$ can also be used to train the weights θ of

the encoder and the decoder by a gradient descent method[18] to minimize $\mathscr{L}(\theta)$. In our prediction algorithm, the features generated in the bottleneck of the AE architecture will be the input of the CRF, which predicts their temporal development.

B. Extended conditional random fields

Before introducing the extended conditional random field (eCRF) used for prediction, we shall revisit the general concept of CRFs. For this purpose, we follow the introduction to CRFs by Sutton and McCallum[44] and adopt their notation, as far as possible. The task of time series prediction for a temporal sequence of frames (fields, images) can be formulated in a probabilistic framework referring to a set of random variables $X \cup Y$, where X is a set of (known) input variables (here: features of the current and previous frames) and Y is a set of (unknown) output variables to be predicted (here: the feature representation of the next future frame). The relation between input and output variables $\mathbf{x} \in X$ and $\mathbf{y} \in Y$, respectively, can be expressed in terms of a conditional probability distribution $p(\mathbf{y}|\mathbf{x})$. Maximizing this conditional probability provides the desired predicted value

$$\mathbf{y}_p = \underset{\mathbf{y}}{\operatorname{argmax}}(p(\mathbf{y}|\mathbf{x})). \tag{5}$$

Despite the dimension reduction provided by the AE, the feature representation of spatiotemporal time series still consists of many variables (in the following examples 32×64, see Fig. 2). To cope with this kind of high-dimensional (conditional) probability distributions, we use here conditional random fields (CRFs)[22] which belong to the class of graphical models (GMs).[19] GMs are a framework for dealing with multivariate probability distributions. They exploit the fact that a distribution over many variables can often be expressed as a product of so-called *factors* (also called *local functions* or *compatibility functions*) that each depend on a (small) subset of variables exploiting statistical independence between (sets of) variables. With CRFs, this decomposition is applied to the conditional probability distribution,

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^{A} \psi_a(\mathbf{y}_a, \mathbf{x}_a), \tag{6}$$

where $F = \{\psi_a\}$ is the set of factors $\psi_a \geq 0$ with a total number of A elements, such that $X_a \subset \mathscr{X}$, $Y_a \subset \mathscr{Y}$, with $\mathbf{x}_a \in X_a$ and $\mathbf{y}_a \in Y_a$. $Z(\mathbf{x})$ is a normalization term, defined later. Such factorizations of probability distributions are called GMs because their structure can efficiently be represented by means of an undirected bipartite graph $G(V, F, E)$ where the first set of nodes V consists of the random variables, F stands for the set of nodes representing the factors (or local functions), and E is the set of edges defining which variables enter which factor ψ_a, see Fig. 1.

(X, Y) is a *conditional random field* if for any $\mathbf{x} \in X$ the conditional probability distribution $p(\mathbf{y}|\mathbf{x})$ factorizes according to the graph G. Up to this point, it was assumed that the $\mathbf{x}, \mathbf{y}$ are a pair of variables occurring at the same time or a temporal state. Assuming a stationary input, the factorization of the probability distribution and the structure of the graph G will not change in time. As will be shown in more detail below, the factors ψ_a are composed of a linear superposition of *feature functions* with corresponding *weights* (representing

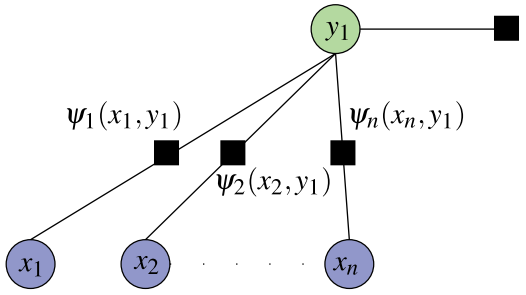

FIG. 1. Example of a graph G describing the distribution $p(y_1|\mathbf{x})$, with local functions $\psi_1(x_1, y_1)$, $\psi_2(x_2, y_1)$,..., $\psi_n(x_n, y_1)$. The circles are variable nodes, where the blue ones are the input and the green one the output, and the shaded boxes are factor nodes.

parameters to be learned). Another step for reducing the complexity of the factorization is the common use of feature functions and weights by several factors. This results in a partitioning of the graph G into a set $\mathscr{C} = \{C_1, C_2, \ldots, C_q\}$ of cliques C_p such that the CRF can be written as[44]

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{C_p \in \mathscr{C}} \prod_{\Psi_c \in C_p} \psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p), \tag{7}$$

where $\mathbf{x}_c$ and $\mathbf{y}_c$ are the parts of $\mathbf{x}, \mathbf{y}$ with shared weights in a clique and where each factor,

$$\Psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p) = \exp \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\mathbf{x}_c, \mathbf{y}_c), \tag{8}$$

is given by the weights θ_{pk} and the feature functions $\mathscr{F} = \{f_{pk}(\mathbf{x}_c, \mathbf{y}_c)\}_{k=1}^{K(p)}$. The normalization function reads

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{C_p \in \mathscr{C}} \prod_{\Psi_c \in C_p} \psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p). \tag{9}$$

To model more than one temporal state, Eq. (7) can be extended by latent variables to a hidden conditional random field (HCRF),[36] where $\mathbf{h} = (h_1, \ldots, h_m)$ is a vector of latent variables, with $h_j \in \mathscr{H}$, where $\mathscr{H}$ is a finite set of possible hidden states. Practically, this means that the encoded elapsed time steps from the AE are stored in $\mathbf{h}$,

$$p(\mathbf{y}, \mathbf{h}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{C_p \in \mathscr{C}} \prod_{\Psi_c \in C_p} \Psi_c(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c; \theta_p), \tag{10}$$

with parameterized factors

$$\Psi_c(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c; \theta_p) = \exp \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c), \tag{11}$$

and the normalization function

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{C_p \in \mathscr{C}} \prod_{\Psi_c \in C_p} \psi_c(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c; \theta_p). \tag{12}$$

Extended conditinal random fields (eCRFs) are defined by extending the parameterized factors to

$$\Psi_c(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c; \theta_p) = \exp\left[\sum_{j=1}^{m} \theta_p^{(h_j)} f_p(\mathbf{x}_c, \mathbf{y}_c, \mathbf{h}_c, j, \tau) + \theta_p^{(y,h_j)} + \sum_{\forall(j,k)s.t.:j\neq k} \frac{\theta_p^{(y)} f_p(\mathbf{x}_c, \mathbf{y}_c, \mathbf{h}_c, j, k, \tau)}{k} + \theta_p^{(y,h_j,h_k)}\right], \quad (13)$$

i.e., an eCRF is given by Eqs. (10), (12), and (13). The extended parameterized factors in Eq. (13) describe (measure) the conditional probability between a possible forecast $\mathbf{y}$, a set of observations $\mathbf{x}$ and a configuration of past states $\mathbf{h}$, where m is the total number of past states. $f_p(\mathbf{x}_c, \mathbf{y}_c, \mathbf{h}_c, j, \tau)$ is a feature function with cliques that can include any subsequence of τ past states to describe the probability between some subsequence of past states and the observed current state. While $f_p(\mathbf{x}_c, \mathbf{y}_c, \mathbf{h}_c, j, k, \tau)$ describes the probability of the current state and a possible future state, it is divided by k to avoid that this term is added multiple times. The parameter vector $\theta_p = \left\{\theta_p^{(h_j)}, \theta_p^{(y,h_j)}, \theta_p^{(y,h_j,h_k)}, \theta_p^{(y)}\right\}$ consists of the parameter vectors $\theta_p^{(h_j)}$ for the parameters corresponding to the state $h_j \in \mathcal{H}$, the parameters $\theta_p^{(y,h_j)}$ describing how well a forecasted feature $\mathbf{y}$ corresponds to a state h_j, the weights $\theta_p^{(y,h_j,h_k)}$ between two edges of the past states h_j and h_k, and some parameters $\theta_p^{(y)}$ for any element y of $\mathbf{y}$ depending on the features over the past. Parameter $\tau = 10$ specifies the number of inputs from the AE taken into account for the forecast.

1. eCRF features

To get feature functions f_p for Eq. (13), a set of base features functions is used for the introduced eCRF model. These base features are used during the training to induce the desired features. The method of *feature induction* is described in Ref. 27. Four base functions are used, where $\mathbf{y}$ is considered to be part of $\mathbf{x}$, for the *feature induction*:

- Weighted neighbor feature

$$f_{\mathrm{WNF}}(\mathbf{x}, \mathbf{y}, \mathbf{h}, j, \tau) = \begin{cases} \theta_{\mathrm{WNF}}^{(1)} \sum_{i=j}^{\tau+j} \|x_j - x_i\|_2 \text{ if } \tau > 1, \\ \theta_{\mathrm{WNF}}^{(2)} \text{ if } \tau = 0. \end{cases} \quad (14)$$

- Mean feature

$$f_{\mathrm{DF}}(\mathbf{x}, \mathbf{y}, \mathbf{h}, j, \tau) = \frac{x_j + \sum_{i=j}^{j+\tau} h_i}{\tau + 1}. \quad (15)$$

- Distance feature

$$f_{\mathrm{MF}}(\mathbf{x}, \mathbf{y}, \mathbf{h}, j, \tau) = x_j - x_{j+\tau}. \quad (16)$$

- For $f_{\mathrm{PM}}(\mathbf{x}, \mathbf{h}, j, k, \tau)$, a probabilistic movement feature based on Ref. 29 is calculated,

$$\mathrm{pm}_t = \begin{bmatrix} x_j(t) \\ \dot{x}_j(t) \end{bmatrix} = \phi_t^T w + \varepsilon_{\mathrm{pm}_t}, \quad (17)$$

$$p(w|\tau) = \prod_{t=0}^{\tau} \mathcal{N}(\mathrm{pm}_t | \phi_t^T w, \Sigma_{\mathrm{pm}_t}), \quad (18)$$

where a weight vector w is used to compactly represent a single trajectory (changes of x_j over the time window τ) given as a linear basis function model ϕ_t and $\varepsilon_{\mathrm{pm}_t} \sim \mathcal{N}(0, \Sigma_{\mathrm{pm}_t})$ is a zero-mean i.i.d Gaussian noise. This model is used to predict the value of x_j at time point $t + 1$,

$$f_{\mathrm{PM}}(\mathbf{x}, \mathbf{h}, j, k, \tau) = x_j + \dot{x}_j, \quad (19)$$

where $x_j + \dot{x}_j$ is the predicted value of x_j at time $t + 1$ based on the subsequence $\{h_j, \ldots, h_k\} \subseteq \{h_1, \ldots, h_m\}$.

C. Training

The parameter determination of the presented approach consists of two phases: First the AE, consisting of the encoding and decoding part, is trained such that $\sum_r |f(x^{(r)}; \theta_{AE}) - x^{(r)}| \stackrel{!}{=} 0$. Then, the parameters θ_{AE} of the AE are fixed and the eCRF is deployed. The combined AE with the eCRF is then used to train the parameters θ_{eCRF} of the eCRF. For this, the likelihood function,

$$\mathcal{L}(\theta_{eCRF}) = \sum_{i=1}^{R} P(\mathbf{y}_i \mid \mathbf{x}_i, \theta_{eCRF}) - \frac{1}{2\sigma^2} \|\theta_{eCRF}\|^2, \quad (20)$$

is used, where R is the number of training examples and σ^2 the L_2 regularizer (in this work, $\sigma^2 = 11$). By maximizing the likelihood for the true prediction on the training data, the optimal parameter set θ^*_{eCRF} is determined. To find θ^*_{eCRF}, Eq. (20) can be maximized by the gradient descent method which is used for optimizing/training the θ_{AE}. The AE part of the method was implemented with Keras[5] and Tensorflow[1] version 1.10, the eCRF was implemented with C++ and Python[38] version 3.6 from scratch.

D. Prediction

To predict the input sequence with the eCRF for one time step, it is necessary to find

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p(\mathbf{y}, \mathbf{h} \mid \mathbf{x}; \theta^*_{eCRF}), \quad (21)$$

where $\hat{\mathbf{y}}$ is calculated elementwise for every element in $\mathbf{y}$ and for each element $\mathbf{x}$ in the feature space. The field of $\hat{\mathbf{y}}$ is then used by the AE decoding part to map the features back to generate the desired output at $t + \Delta t$. The future evolution of the spatiotemporal time series is predicted by using a feedback loop from the output of the AE back to the input.

III. RESULTS

In order to benchmark our method, spatiotemporal time series from two systems were examined. The first system is the Lorenz-96 model,[24] a system with an adjustable number of ordinary differential equations exhibiting high-dimensional chaotic behavior. This system is often used as a benchmark in data assimilation.[35,37] The second set of time series was generated by the Kuramoto-Sivashinsky equation,[21,40,41] a nonlinear PDE which was also used as benchmark

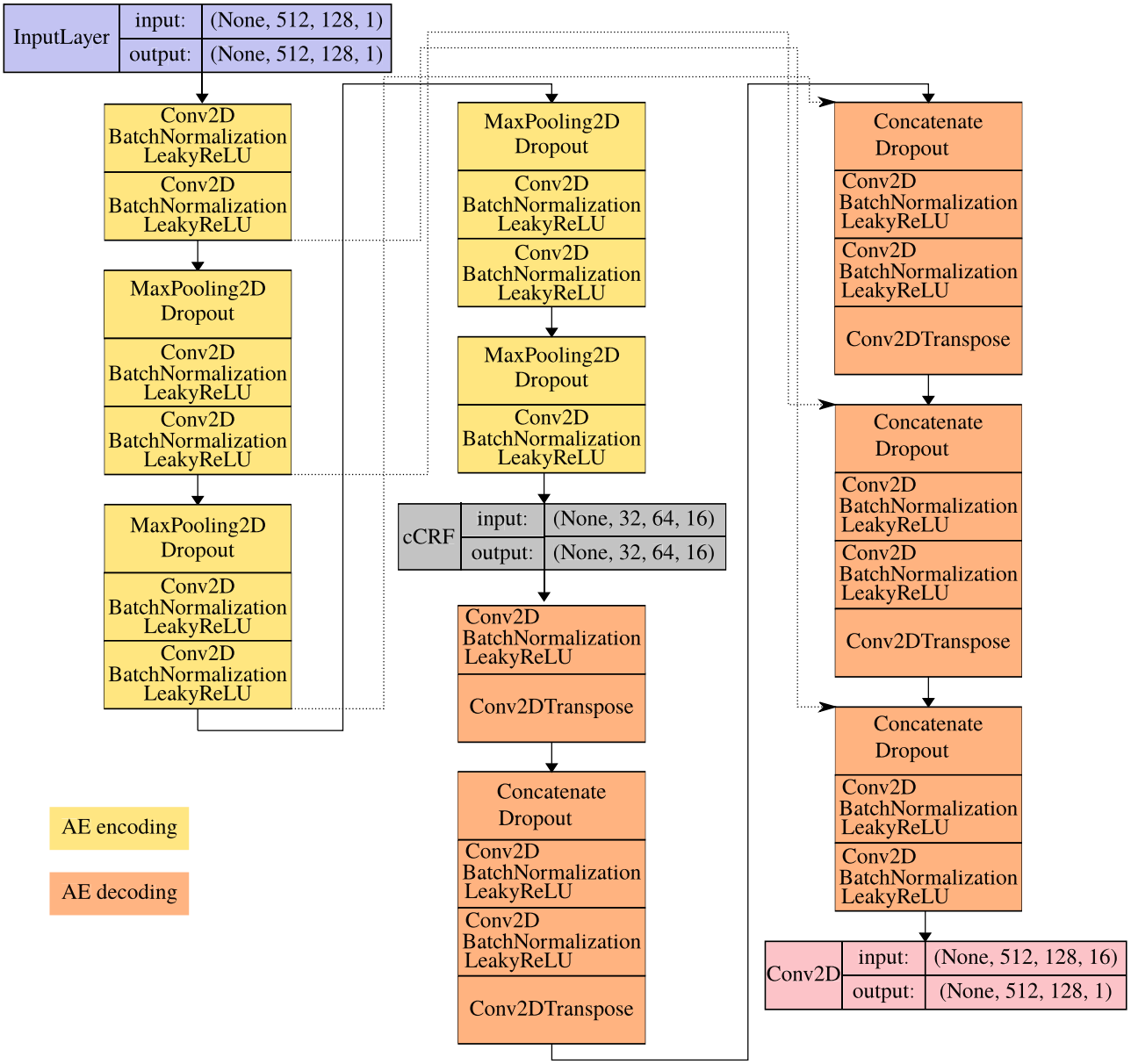

FIG. 2. Proposed architecture for prediction, consisting of a split autoencoder and the eCRF. Each rectangular block is a set of layers. The first dimension in the input and output boxes is the batch size which is **None** meaning that this dimension is variable. The second variable 512 is the size of state vector, and the 128 the number of time steps taken into account for performing one input for the eCRF, the last size is the number of channels or filters used in the layer.

system in Ref. 31. In both cases, we investigate how long the mean absolute error between the ground truth data and the predicted data is smaller than some given $\varepsilon_{\text{error}}$. This prediction horizon is then compared to the Lyapunov time, a characteristic time scale given by the inverse of the largest Lyapunov exponent $\Lambda_{\max}$. As input, we always use $\tau = 10$ fields where the first dimension of each field is the system size and the second dimension is the number of time steps (here: 128).

A. Lorenz-96 model

The Lorenz-96 model is a dynamical system

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \tag{22}$$

introduced by Lorenz[24], where x_i $(i = 1, \ldots, N)$ is a component of the system state $\mathbf{x} = [x_1, \ldots, x_N]$ and $F \in \mathbb{R}$ is a forcing constant.

The components of $\mathbf{x}$ are arranged in a ring with periodic boundary conditions $x_{-1} = x_{N-1}$, $x_0 = x_N$, and $x_{N+1} = x_1$.

In the context of this manuscript, $F = 8$ and $N \in \{32, 64, 128, 256\}$. For each N, 100 cases were generated where the initial state was set to F and 10% of the components were perturbed with some random values from a uniform distribution between $[-0.01, 0.01]$. To integrate the system, we used $\Delta t = 0.01$ and $t_{\max} = 45.6$ rendering in total 4560 time steps. For integration, the python implementation of lsoda[34] was used. However, only the last 2560 time steps were considered and the first 2000 time steps of the transient to the attractor were discarded to avoid initialization artifacts. The data were then divided into sequences of 128 time steps, such that $X \in \mathbb{R}^{N\times 128}$, and randomly divided in 50% training, 25% validation and 25% test data. The following results are all from the test data. In Figs. 3 and 4, cases $N = 32$ and $N = 128$ are shown as examples. By setting the threshold $\varepsilon_{\text{error}} = 0.35$ for the mean absolute error, we define the prediction horizon, i.e., the point in time until successful forecasts are achieved. It should be noted that the choice of this value was subjective, it was the point where a deviation from the ground truth can be seen by bare eyes. With this threshold, the Lorenz-96 model is successfully predicted for $N = 32$ over a period of time $9.55t$ corresponding to 955 time steps. By multiplying t with the largest Lyapunov exponent $\Lambda_{\max}$ (see Table I), this corresponds to over 15 Lyapunov times. A similar behavior can be seen for the case $N = 128$ (Fig. 4). Although the prediction time-interval decreases, it is found that in terms of Lyapunov time a similar prediction duration exists, since $\Lambda_{\max}$ has increased. However, Figs. 3 and 4 only provide a general intuition. The duration of how long the forecast remains correct depends also on the initial state on the attractor. Therefore, in Fig. 5a, the prediction statistics for all 500 test sequences for $N \in \{32, 64, 128, 256\}$ is shown with different initial conditions in the test data. The median for the prediction horizon (in units of the Lyapunov time) is 15.48 for $N = 32$, 15.03 for $N = 64$, 15.00 for $N = 128$ and 14.86 for $N = 256$.

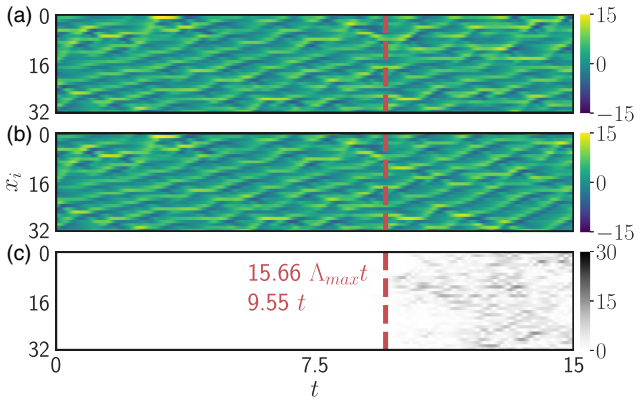

FIG. 3. Comparison between real (ground truth) data (a) with the predicted data (b) for the Lorenz-96 model with $N = 32$. (c) shows the absolute error between real data and the forecast. The red line indicates the time when the mean absolute error is larger than $\varepsilon_{\text{error}} = 0.35$ corresponding to an error of about 1.2 %. Using this conservative criterion for this example, a prediction horizon of 15.66 $\Lambda_{\max}t$ is achieved.

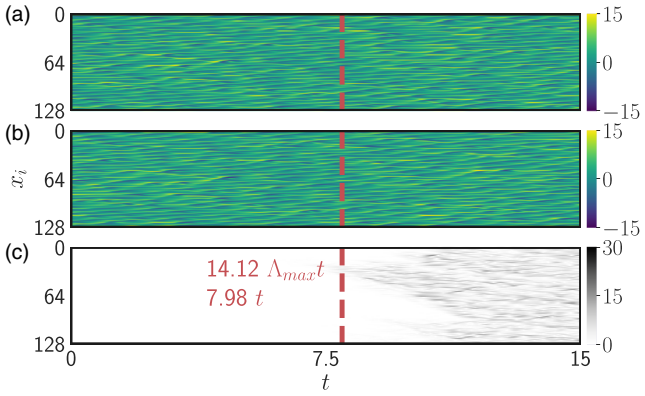

FIG. 4. Same as Fig. 3 for $N = 128$, where we achieve a prediction horizon of 14.12 $\Lambda_{\max}t$ under the same constraints as in Fig. 3.

B. Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky equation (KSE),

$$u_t + \nu\nabla^4 u + \nabla^2 u + \frac{1}{2}|\nabla u|^2 = 0,$$
$$(x,t) \in \mathbb{R}\times\mathbb{R}_+,$$
$$u(x,0) = u_0(x), \quad u(x+L,t) = u(x,t), \tag{23}$$

is a fourth-order nonlinear partial differential equation introduced for modeling diffusive instabilities in a laminar flame front,[41] where ∇^2 is the Laplacian, ∇^4 is the biharmonic operator, ∇ is the gradient, ν is a positive constant and u_0 is L-periodic, where L is the size of a pattern cell.[41] In the specific limit of large nondimensional surface tension, it was proved[42] that for $y(x,t) = \nabla u(x,t)$ and $\nu = 1$ the KSE can be written as

$$y_t + \nabla^4 y + \nabla^2 y + y\nabla y = 0. \tag{24}$$

In order to achieve a comparison with recent results by Pathak *et al.*[31] representing the current the state of the art of data-driven time series prediction, we also extended Eq. (24) with a spatial inhomogeneity term, as in Ref. 31,

$$y_t + \nabla^4 y + \nabla^2 y + y\nabla y - \mu\cos\left(\frac{2\pi x}{\lambda}\right) = 0. \tag{25}$$

Like in Ref. 31, Eq. (24) is integrated on a grid of Q equally spaced points with $\Delta t = 0.25$ and $t_{\max} = 2500$ rendering 10 000 time

TABLE I. Largest Lyapunov exponent $\Lambda_{\max}$ and Kaplan-Yorke attractor dimension (D_{KY})[17] for Lorenz-96 systems with different dimensions N.

N	$\Lambda_{\max}$	D_{KY}
32	1.64	21.7
64	1.72	43.0
128	1.77	86.6
256	1.79	173.3

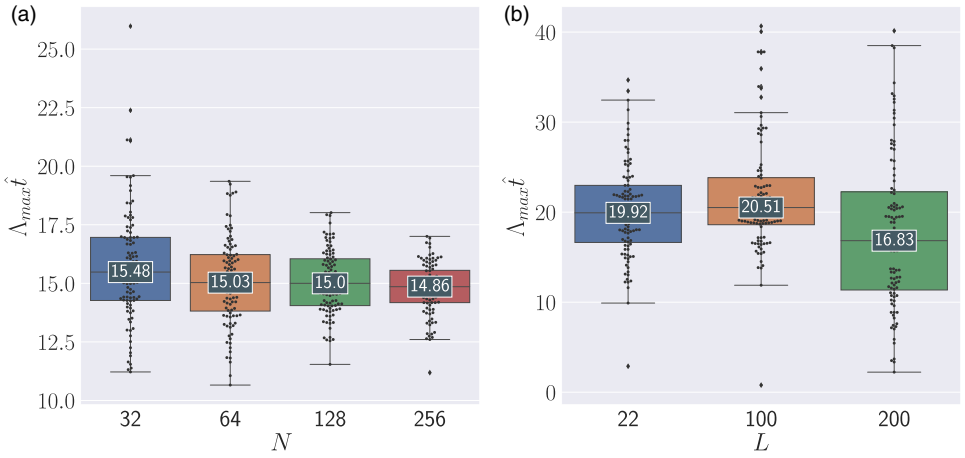

FIG. 5. Lyapunov time statistics for (a) Lorenz-96 systems of dimensions $N \in \{32, 64, 128, 256\}$ and b) KSEs with $L = 22, Q = 64, \mu = 0$, $L = 100, Q = 256, \mu = 0.01, \lambda = 100$, and $L = 200, Q = 512, \mu = 0.01, \lambda = 100$. $\hat{t}$ is the time when the mean absolute error for the prediction exceeds $\varepsilon_{error} = 0.35$ for the Lorenz-96 system and $\varepsilon_{error} = 0.07$ for the KSE.

steps. For integration, the exponential time differencing fourth-order Runge-Kutta algorithm (ETDRK4)[6] was used. To compare our results with Pathak *et al.*,[31] we considered the cases $L \in \{22, 100, 200\}$.

Similar to the analysis of the Lorenz-96 model, for each L, 100 different cases were generated with random initialization and the last 7936 time steps, such that subsequences $X \in \mathbb{R}^{L\times 128}$ representing 128 time steps for later training, validation, and test purposes. Figure 6 shows the prediction results for the KSE with $L = 22, Q = 64$, and $\mu = 0$. In this specific case a prediction horizon of 22.97 Lyapunov times was achieved with an error threshold of $\varepsilon_{error} = 0.07$. For the case of $L = 200, Q = 512, \mu = 0.01$, and $\lambda = 100$ (Fig. 7), a comparable result of 22.18 $\Lambda_{\max} t$ was found.

As with the Lorenz-96 model, the prediction duration of the KSE is also dependent on the initial state on the attractor. The corresponding statistics are shown in Fig. 5(b). According to our knowledge, the current best prediction results for the KSE were reported in Ref. 31 for an echo state network which achieved ≈ 3 Lyapunov times (read from Fig. 2 in Ref. 31) for $L = 22$ and ≈ 8 Lyapunov times for $L = 200$ (reported in text of Ref. 31). Unfortunately, there it was not reported at which error threshold for the prediction horizon these values were taken. We exceed the prediction horizon obtained with the echo state network, even when only considering median values, substantially with 19.92 Lyapunov times for $L = 22$ and 16.83 for $L = 220$. Of course, training of our hybrid approach is much more time consuming than the learning procedure with reservoir computing.

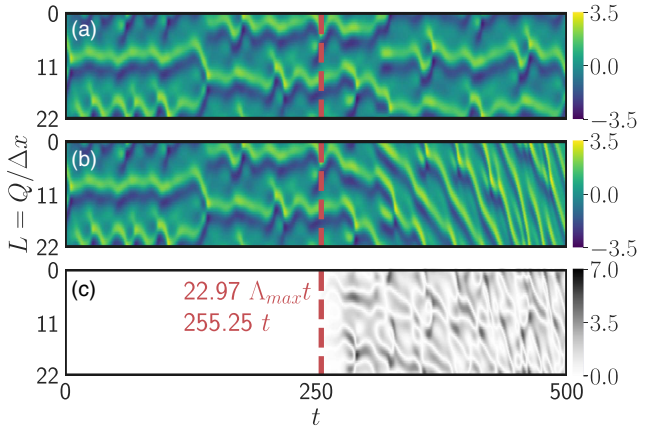

FIG. 6. Comparison between real data (a) with predicted data (b) of the KSE with $L = 22, Q = 64$, and $\mu = 0$. (c) shows the absolute error between ground truth and prediction. The red line indicates the point at which the mean absolute error is greater than $\varepsilon_{error} = 0.07$ corresponding to an error of about 1.2%, using this criterion a prediction horizon of 22.97 $\Lambda_{\max} t$ is achieved.

IV. DISCUSSION

The presented algorithm delivers promising results, but one can ask, why this is so? Alas, this hybrid method does not easily lend itself to intuition. While the system necessarily uses the temporal history of

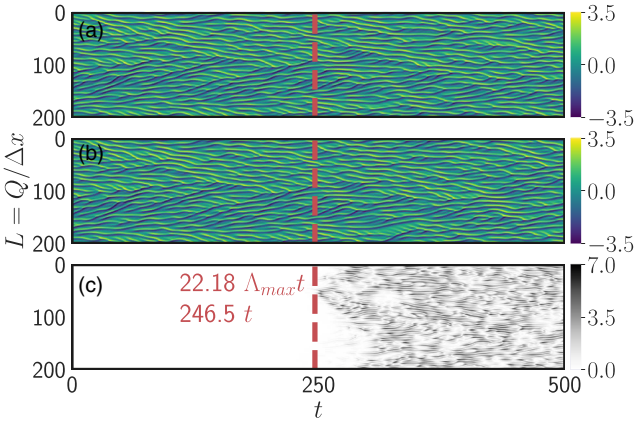

FIG. 7. Same as in Fig. 6 with $L = 200, Q = 512, \mu = 0.01$, and $\lambda = 100$, where we get a prediction horizon of 22.18 $\Lambda_{\max} t$ under the same constraints as in Fig. 6.

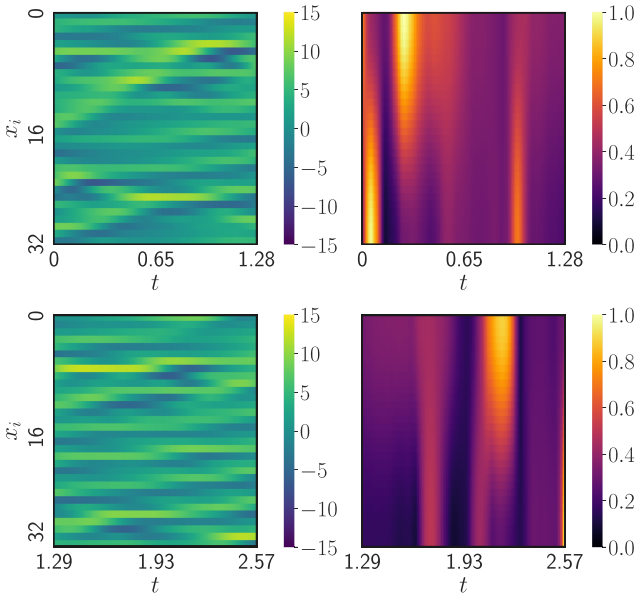

FIG. 8. Attention map for the Lorenz-96 system with $N = 32$. The left side is the input image and the right side is the corresponding attention map. The panels at the bottom show the same for a different moment in time along the time series.

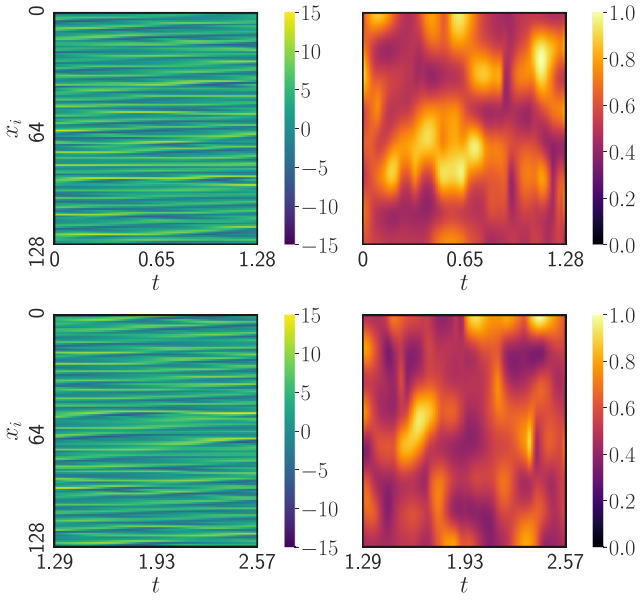

FIG. 9. As Fig. 8 but here with system size $N = 128$.

the time series, this does not happen in any straightforward manner. For example, the simple assumption that there is a continuous decay of the importance of data that is longer in the past is not true.

The autoencoder (AE) compresses the data at its bottleneck layers, leading to a much reduced and importance-weighted representation of the input. To make this visible, we need to backproject the importance-weighting into the input space. For this, one can use so-called gradient-based attention maps, which are maps where the value of the positive gradients from the AE, after training, is weighted by the loss of Eq. (4) and projected back to the input space. Hence, they allow us to inspect which parts of the input are most strongly represented in the bottleneck of the AE. The method to generate such attention maps was presented in Ref. 39. Figure 8 shows the input situation (left) for a certain moment in time of the Lorenz-96 system for a period of time and the corresponding attention map (right) for $N = 32$. Hence, only those parts of the input, where the attention map has high values, are considered by the encoder-pathway of the AE. A similar diagram for $N = 128$ is shown in Fig. 9. There is no simple temporal structure visible in either of these attention maps and the resulting structures are complex. Furthermore, it is important to note that these attention maps will look different for different time points along the time series. On more theoretical grounds, a certain commonality, however, exists for the importance-weighing performed by the AE across all cases. Autoencoders have been investigated in depth in the recent past.[8,12,45] These works show that AEs are capable of learning disentangled representations of the data to construct a latent representation that reconstructs existing (or generates new) samples from complex, high-dimensional distributions. We suggest that exactly this behavior is a key component for the performance of our approach.

The disentanglement property creates features in the bottleneck layer of the AE that can effectively be understood as independent random variables. Under such conditions, conditional random fields (CRF) are known to operate with high performance due to optimal factorization and representation. We suggest that these aspects may underlie the high prediction performance observed with our hybrid prediction method.

The here-presented method delivers quite a large prediction horizon on complex time series. As a consequence, different applications for such a system are conceivable. It can be used to perform data-driven modeling for systems where no mathematical models, based on first principles, exist, or which are very expensive to solve. A second, potentially interesting application would be the acceleration of complex model calculations, where a hybrid approach is used that iterates conventional numerical simulations with predictions by our method. This way, costly and potentially slow numerical simulations do not have to be performed for every timestep. Instead, missing data would be filled in by the predictions made by our approach.

The work we have done so far is intended to be a proof of concept; there are still many components that can probably be improved to some degree. In this work, we have not tried to tune hyperparameters, including the architecture of the AE or of the eCRF. The way in which we have trained the parameters meets the state of the art in machine learning, but also, here, possibilities exist for optimization. One option would be to use a bias-variance balancing objective function.[32] Another possibility, which could even be considered in addition, would be population-based training.[16]

ACKNOWLEDGMENTS

We thank Stefan Luther for continuous support and inspiring discussions on applications of nonlinear time series analysis in cardiac dynamics and beyond, and Jonas Isensee for support with the implementation of the Kuramoto-Sivashinsky equation and interesting exchanges about state reconstructions of nonlinear systems. S.H. acknowledges funding by the International Max Planck Research Schools of Physics of Biological and Complex Systems. This work was supported by the European Commission's H2020-Framework FET-Proactive Grant Plan4Act (No. 732266).

REFERENCES

[1]M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems" (2015), see tensorflow.org.

[2]A. J. Abebe, D. P. Solomatine, and R. G. W. Venneker, "Application of adaptive fuzzy rule-based models for reconstruction of missing precipitation events," Hydrolog. Sci. J. **45**, 425–436 (2000).

[3]G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control* (Holden-Day Inc., San Francisco, CA, 1990).

[4]Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep convolutional autoencoder-based lossy image compression," in *2018 Picture Coding Symposium, PCS 2018—Proceedings* (Institute of Electrical and Electronics Engineers Inc., 2018), pp. 253–257.

[5]F. Chollet *et al.*, see https://keras.io for "Keras" (2015).

[6]S. Cox and P. Matthews, "Exponential time differencing for stiff systems," J. Comput. Phys. **176**, 430–455 (2002).

[7]C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Computer Vision—ECCV 2014*, edited by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Springer International Publishing, Cham, 2014), pp. 184–199.

[8]S. Gao, R. Brekelmans, G. V. Steeg, and A. Galstyan, "Auto-encoding total correlation explanation," e-print arXiv:1802.05822[cs.LG] (2018).

[9]R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," Nature **405**, 947 (2000).

[10]K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2016), pp. 770–778; see https://ieeexplore.ieee.org/document/7780459.

[11]S. Herzog, F. Wörgötter, and U. Parlitz, "Data-driven modeling and prediction of complex spatio-temporal dynamics in excitable media," Front. Appl. Math. Stat. **4**, 60 (2018).

[12]I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner, "Early visual concept learning with unsupervised deep learning," e-print arXiv:1606.05579 (2016).

[13]G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science **313**, 504–507 (2006).

[14]S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," e-print arXiv:1502.03167[cs.LG] (2015).

[15]J. Isensee, G. Datseris, and U. Parlitz, "Predicting spatio-temporal time series using dimension reduced local states," J. Nonlinear Sci. (published online).

[16]M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks," e-print arXiv:1711.09846[cs.LG] (2017).

[17]J. L. Kaplan and J. A. Yorke, "Chaotic behavior of multidimensional difference equations," in *Functional Differential Equations and Approximation of Fixed Points*, edited by H.-O. Peitgen and H.-O. Walther (Springer, Berlin, 1979), pp. 204–227.

[18]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," e-print arXiv:1412.6980 (2014).

[19]D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning Series)* (The MIT Press, 2009).

[20]A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems—Volume 1*, series and number NIPS'12 (Curran Associates Inc., 2012), pp. 1097–1105.

[21]Y. Kuramoto, "Diffusion-induced chaos in reaction systems," Prog. Theor. Phys. Suppl. **64**, 346–367 (1978).

[22]J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the 18th International Conference on Machine Learning* (Morgan Kaufmann Publishers Inc., 2001), pp. 282–289.

[23]Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2*, edited by D. S. Touretzky (Morgan-Kaufmann, 1990), pp. 396–404.

[24]E. Lorenz, "Predictability: a problem partly solved," in *Seminar on Predictability, 4–8 September 1995*, ECMWF (ECMWF, Shinfield Park, Reading, 1995), Vol. 1, pp. 1–18.

[25]E. N. Lorenz, "Deterministic nonperiodic flow," J. Atmos. Sci. **20**, 130–141 (1963).

[26]A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the 30th International Conference on Machine Learning* (Atlanta, Georgia, 2013); see also: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

[27]A. McCallum, "Efficiently inducing features of conditional random fields," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, series and number UAI'03 (Morgan Kaufmann Publishers Inc., San Francisco, CA, 2003), pp. 403–410.

[28]J. Nassar, S. Linderman, M. Bugallo, and I. M. Park, "Tree-structured recurrent switching linear dynamical systems for multi-scale modeling," in *International Conference on Learning Representations* (International Conference on Learning Representations (ICLR), 2019).

[29]A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems 26*, edited by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Curran Associates, Inc., 2013), pp. 2616–2624.

[30]U. Parlitz and C. Merkwirth, "Prediction of spatiotemporal time series based on reconstructed local states," Phys. Rev. Lett. **84**, 1890–1893 (2000).

[31]J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).

[32]M. Pavlovski, F. Zhou, N. Arsov, L. Kocarev, and Z. Obradovic, "Generalization-aware structured regression towards balancing bias and variance," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18* (International Joint Conferences on Artificial Intelligence Organization, 2018), pp. 2616–2622.

[33]B. Penkovsky, X. Porte, M. Jacquot, L. Larger, and D. Brunner, "Coupled nonlinear delay systems as deep convolutional neural networks," Phys. Rev. Lett. **123**, 054101 (2019).

[34]L. Petzold, "Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations," SIAM J. Sci. Stat. Comput. **4**, 136–148 (1983).

[35]F. R. Pinheiro, P. J. van Leeuwen, and U. Parlitz, "An ensemble framework for time delay synchronization," Q. J. R. Meteorolog. Soc. **144**, 305–316 (2018).

[36]A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," IEEE Trans. Pattern Anal. Mach. Intell. **29**, 1848–1852 (2007).

[37]D. Rey, M. Eldridge, M. Kostuk, H. D. Abarbanel, J. Schumann-Bischoff, and U. Parlitz, "Accurate state and parameter estimation in nonlinear systems with sparse observations," Phys. Lett. A **378**, 869–873 (2014).

[38]G. van Rossum and F. L. Drake, *The Python Language Reference Manual* (Network Theory Ltd., 2011).

[39]R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," e-print arXiv:1610.02391[cs.CV] (2016).
[40]G. Sivashinsky, "Nonlinear analysis of hydrodynamic instability in laminar flames–i. derivation of basic equations," Acta. Astronaut. **4**, 1177–1206 (1977).
[41]G. I. Sivashinsky, "On flame propagation under conditions of stoichiometry," SIAM J. Appl. Math. **39**, 67–82 (1980).
[42]G. I. Sivashinsky and D. M. Michelson, "On irregular wavy flow of a liquid film down a vertical plane," Progr. Theor. Phys. **63**, 2112–2114 (1980).
[43]N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res. **15**, 1929–1958 (2014); available at http://dl.acm.org/citation.cfm?id=2627435.2670313.
[44]C. Sutton and A. McCallum, "An introduction to conditional random fields," Found. Trends Mach. Learn. **4**, 267–373 (2012).
[45]M. Tschannen, O. Bachem, and M. Lucic, "Recent advances in autoencoder-based representation learning," e-print arXiv:1812.05069 (2018).
[46]V. Vemuri, *Artificial Neural Networks: Theoretical Concepts*, Computer Society Press Technology Series: Neural networks (Computer Society Press of the IEEE, 1988).
[47]P. Vlachas, W. Byeon, Z. Yi Wan, T. P. Sapsis, and P Koumoutsakos, "Data-driven forecasting of high-dimensional chaotic systems with long-short term memory networks," Proc. R. Soc. A Math. Phys. Eng. Sci. **474**, 1 (2018).
[48]W. Zhang, B. Wang, Z. Ye, and J. Quan, "Efficient method for limit cycle flutter analysis based on nonlinear aerodynamic reduced-order models," AIAA J. **50**, 1019–1028 (2012).