Physica D 403 (2020) 132261

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Classification of chaotic time series with deep learning

Nicolas Boullé [a,*], Vassilios Dallas [a,*], Yuji Nakatsukasa [a], D. Samaddar [b]

[a] Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK
[b] United Kingdom Atomic Energy Authority, Culham Centre for Fusion Energy, Culham Science Centre, Abingdon, Oxon, OX14 3DB, UK

ARTICLE INFO

Article history:
Available online 4 December 2019
Communicated by V.M. Perez-Garcia

Keywords:
Dynamical systems
Chaos
Deep learning
Time series
Classification

ABSTRACT

We use standard deep neural networks to classify univariate time series generated by discrete and continuous dynamical systems based on their chaotic or non-chaotic behaviour. Our approach to circumvent the lack of precise models for some of the most challenging real-life applications is to train different neural networks on a data set from a dynamical system with a basic or low-dimensional phase space and then use these networks to classify univariate time series of a dynamical system with more intricate or high-dimensional phase space. We illustrate this generalisation approach using the logistic map, the sine-circle map, the Lorenz system, and the Kuramoto–Sivashinsky equation. We observe that a convolutional neural network without batch normalisation layers outperforms state-of-the-art neural networks for time series classification and is able to generalise and classify time series as chaotic or not with high accuracy.

1. Introduction

Data and in particular time series are generated from numerous observations and experiments across different scientific fields such as atmospheric and oceanic sciences for climate predictions, nuclear fusion for control and safety, biology and medicine for diagnosis. Fourier transforms, radial basis functions approximation and standard numerical techniques have been extensively applied to perform short and long term predictions of chaotic time series [1–4]. On the other hand, the spectacular success of machine learning and deep learning techniques to image classification [5,6], which have recently surpassed human-level performance on the ImageNet data set [7], has inspired the development of neural network techniques for time series forecasting [8,9] and classification [10]. Recently, deep learning approaches have been used to solve partial differential equations in high dimensions [11–13] and identify hidden physics models from experimental data [14–17].

The size of the data sets is often large and analysing these time series represents a huge computational challenge and interest nowadays. For some of the most challenging real-life applications a precise dynamical system is unknown, which makes the identification of the different dynamical regimes impossible. In that spirit, machine learning has been recently employed by Pathak et al. [18,19] to perform model-free predictions of chaotic dynamical systems. Moreover, deep learning requires a large data set to adequately train the artificial neural network, which might not be available in some cases due to the infinite dimensional phase space of the system or experimental constraints.

In this paper, we address the aforementioned challenges by considering the following classification problem: given a univariate time series generated by a discrete or continuous dynamical system, can we determine whether the time series has a chaotic or non-chaotic behaviour? We choose to train neural networks on a different set than the testing set of interest in order to assess the ability of the machine learning algorithms to generalise the classification of univariate time series of a dynamical system with a basic or low-dimensional phase space to a more intricate or high-dimensional one. Our aim is to demonstrate the generalisation ability of neural networks in this classification problem using standard deep learning models. The main challenge here is to learn the chaotic features of a training set, whose chaotic behaviour can be determined a priori using measures from dynamical systems theory, without overfitting, and generalise on a testing data set, which comes from another dynamical system, whose behaviour is different.

The paper is organised as follows. We briefly describe five different neural networks architectures for time series classification in Section 2. Then, in Section 3, we classify signals generated by discrete dynamical systems and compare the accuracy of the neural networks. Finally, Section 4 consists of the classification of time series generated by the Lorenz system and the Kuramoto–Sivashinsky equation.

* Corresponding authors.
E-mail addresses: boulle@maths.ox.ac.uk (N. Boullé), vassilios.dallas@maths.ox.ac.uk (V. Dallas), nakatsukasa@maths.ox.ac.uk (Y. Nakatsukasa), debasmita.samaddar@ukaea.uk (D. Samaddar).

https://doi.org/10.1016/j.physd.2019.132261
0167-2789/

2. Neural networks for time series classification

Time series classification is one of the most challenging problems in machine learning [20] with a wide range of applications in human activity recognition [21], acoustic scene classification [22], and cybersecurity [23]. In this section, we describe five different architectures that we have considered for classifying time series generated by discrete and continuous dynamical systems.

First, we consider a simple shallow neural network (see Section 2.1) in order to compare it with state-of-the-art classifiers. The multilayer perceptron [10], presented in Section 2.2, is chosen because it is a fully connected deep neural network, which does not rely on convolutional layers. Convolutional neural networks have first been introduced to perform handwritten digit [24] and have been successfully applied to images and time series [5,6,25]. The next two neural networks (see Sections 2.3 and 2.4) are then based on convolutional layers and were chosen on the basis that they achieve the best performance on standard time series data sets according to the recent review by Fawaz et al. [26]. Finally in Section 2.5 we consider a convolutional neural network for time series classification with a large kernel size to decrease the computational expense.

In what follows, the input of the neural network is a univariate time series X of size T (in practice we take T to be one thousand). Each of the time series is assigned a class label that we want to recover using the different neural networks: Class NC corresponds to a non-chaotic time series while Class C corresponds to a chaotic time series. The neural networks presented in this section end with a softmax layer, which outputs a vector of probabilities $[p_{NC}, p_C]$, where p_{NC} is the probability that the input time series is non-chaotic and $p_C = 1 - p_{NC}$ is the probability that it is chaotic. A time series is classified as non-chaotic if $p_{NC} \geq 0.5$ and chaotic otherwise. The performance of the neural networks is then assessed using the classification accuracy defined as

$$\text{Accuracy} = \frac{T_{NC} + T_C}{T_{NC} + T_C + F_{NC} + F_C} \times 100,$$

where T_{NC} = true non-chaotic predictions, T_C = true chaotic predictions, F_{NC} = false non-chaotic predictions, and F_C = false chaotic predictions.

Analysing our results with other metrics than accuracy can be important particularly when the testing data sets are not balanced. For this reason we also considered metrics such as precision, recall [27, Chapt. 9], and balanced accuracy [28]. However, we saw that our findings were not influenced by these metrics even though their values change. Thus, we choose to report only the values of the classification accuracy which are not better or worse than the other metrics.

2.1. Shallow neural network

The first type of networks considered in this section is shallow neural networks (ShallowNet), which are simple and efficient networks for fitting functions and perform pattern recognition. These networks differ from deep neural network since they usually contain only one or two hidden layers. We use MATLAB's `patternnet` command from the Deep Learning Toolbox to define a network with one hidden layer, containing one hundred neurons, with the sigmoid as activation function. The network is trained with the Scaled Conjugate Gradient algorithm [29] using MATLAB's default settings.

2.2. Multi layer perceptrons

Multilayer perceptrons (MLP) are standard deep neural network architectures in the field of machine learning and essentially consist of fully connected layers separated by a nonlinear activation function. Wang, Weizhong and Oates [10] use a structure of three hidden layers of five hundred neurons followed by a rectified linear unit (ReLU) to perform time series classification (a Python implementation using TensorFlow is available in [30]). Moreover, a dropout is added at the input, hidden and softmax layers with rates $\{0.1, 0.2, 0.3\}$, respectively. The network is trained on 10 epochs with a batch size of 32 using Adam's algorithm [31]. We use the same parameters and optimisation algorithm to train the following three convolutional neural networks.

2.3. Fully convolutional neural network

The fully convolutional neural network (FCN) architecture considered in [10] is a succession of three convolutional blocks, followed by a global averaged pooling layer [32] and a softmax layer. The first (resp. second, third) convolutional block that we consider is composed of a convolution layer of kernel size 8 (resp. 5, 3) with 64 (resp. 128, 64) feature channels, a batch normalisation layer [33] and a ReLU activation layer. The fully convolutional neural network studied by [10] is implemented in [30].

2.4. Residual network

The last network considered by Wang, Weizhong and Oates [10] in the context of time series classification is a residual network (ResNet). Residual networks are examples of very deep neural network and are designed by stacking the convolutional blocks arising in the FCN (see Section 2.3). Then, the ResNet is created by assembling three blocks of the FCN to generate a residual block. Three residual blocks, with $\{64, 128, 128\}$ respective number of feature channels, are then stacked and followed by a global average pooling layer and a softmax layer to output the classification of the different input time series. Ref. [30] provides a practical Python implementation.

2.5. Large kernel convolutional neural network

We consider a standard convolution neural network with large kernel size (LKCNN). The network is composed of two convolutional layers with five feature channels and kernel size of a hundred, followed by ReLU activation, a maximum pooling layer with a pool size of two, a flatten layer, and two fully connected layers of respective size one hundred and two. The ReLU activation function is chosen because it is easy to optimise due to its piecewise linearity [34, Chap. 6]. Moreover, a dropping out unit (dropout) with rate 0.5 is added after the maximum pooling layer to improve the generalisation ability of the network [35]. The architecture of the network is shown in Fig. 1.

Standard implementations of convolutional neural networks usually consider a larger number of feature channels and a much smaller kernel size [10,26]. However, we observed that the classification accuracy of this network was not affected by the kernel size and decreased as we increased the number of feature channels (see Appendix). Thus, we choose to use a large kernel size and a small number of feature channels in order to reduce the number of trainable parameters and thus reduce the computational expense of this network. Similarly, we verified that increasing the number of convolutional layers from two to three

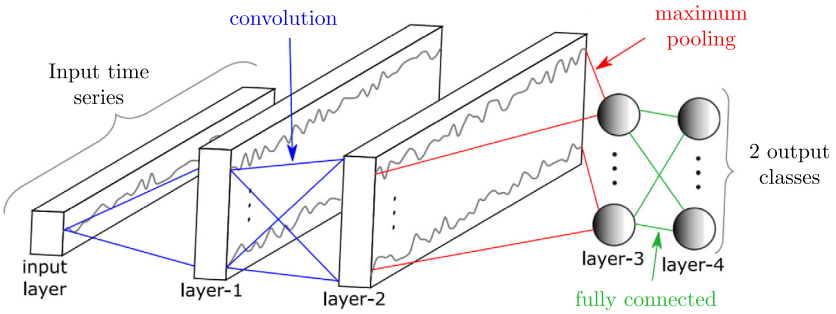

Fig. 1. The large kernel convolutional neural network (LKCNN) architecture for time series classification. *Source:* Figure adapted from [26].

or four does not significantly improve the performance of LKCNN overall.

The main difference that determines the classification accuracy between LKCNN, FCN, and ResNet is that FCN and ResNet have batch normalisation layers. Batch normalisation was introduced to speed up the optimisation algorithm that occurs at the training phase by normalising the training input data at the internal layers [33].

In this work, we study the generalisation ability from a training set to a testing set with time series that span different range of values. For this reason, we normalise both the training set and the testing set (see details in the following sections) so that the values vary within the same range. Note that the scaling and shifting parameters of the batch normalisation layers are determined in the training phase [33]. Therefore, applying a batch normalisation to a data set with different mean and variance would be wrong. In our problem this happens with the testing data, which can have different mean and variance from the training data after the convolutional layers. This explains the lack of performance of the FCN and ResNet on the testing data sets as we will show in the following sections.

3. Discrete dynamical systems

In this section we consider two discrete dynamical systems called the logistic map and the sine-circle map. The first one is the logistic map, popularised by Robert May [36], which is defined by the sequence

$$x_{n+1} = \mu x_n(1 - x_n), \quad x_0 = 0.5, \tag{1}$$

where μ is the bifurcation parameter varying between zero and four. This system exhibits periodic or chaotic behaviour depending on the value of μ. Periodic and chaotic signals of the logistic map are plotted in Fig. 2.

The bifurcation diagram showing the orbits of the logistic map is represented in Fig. 3 (top). The behaviour of the attractors for different parameters μ has been extensively studied [37, Chap. 10] and a highlight is the period-doubling cascade happening for $\mu \in [0, 3.54409]$.

The second dynamical system considered in this section is the sine-circle map [38, Chap. 6], which is sometimes referred to as the circle map. It takes the form of the following nonlinear map

$$\theta_{n+1} = \theta_n + \Omega - \frac{\mu}{2\pi}\sin(2\pi\theta_n) \mod [1], \quad \theta_0 = 0.5, \tag{2}$$

where $\Omega = 0.606661$ and $\mu \in [0, 5]$ is the parameter that measures the strength of the nonlinearity. Similarly to the logistic map, iterating Eq. (2) leads to periodic or chaotic signals depending on the bifurcation parameter μ chosen. Fig. 4 illustrates two signals with different behaviours, generated using a bifurcation parameter of $\mu = 2.1$ (top) and $\mu = 2.3$ (bottom). The bottom

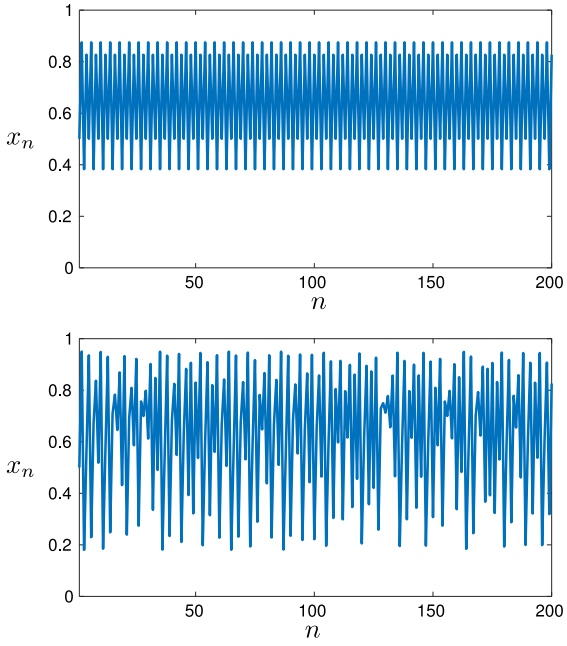

Fig. 2. A periodic (top) and a chaotic (bottom) signal of the logistic map of size two hundred with $\mu = 3.5$ and $\mu = 3.8$, respectively.

panel of Fig. 3 shows the bifurcation diagram of the sine-circle map.

We now want to classify signals generated by the logistic and sine-circle maps according to their chaotic and non-chaotic behaviour. Our main goal, and challenge, is to find a neural network that is able to learn the features characterising chaotic signals of the logistic map and generalise on signals generated by the sine-circle map. To do this, we generate two data sets by computing signals of length one thousand of the logistic and sine-circle maps. This is done for five thousand different values of the parameter μ, uniformly distributed in the interval $[0, 5]$. The logistic (resp. sine-circle) data set is then composed of 24% (resp. 35%) of chaotic time series. First, we randomise the logistic map data set across the bifurcation parameter and we choose two thirds of the data to be the training set. Then, with the rest (one third) of the logistic map data set, we test the training of the five neural networks described in Section 2. Note that to classify the time series of the sine-circle data set we use these five neural networks, which have been trained on the logistic map training set.

The classification of the time series is done using two measures from dynamical systems theory. The first measure is the

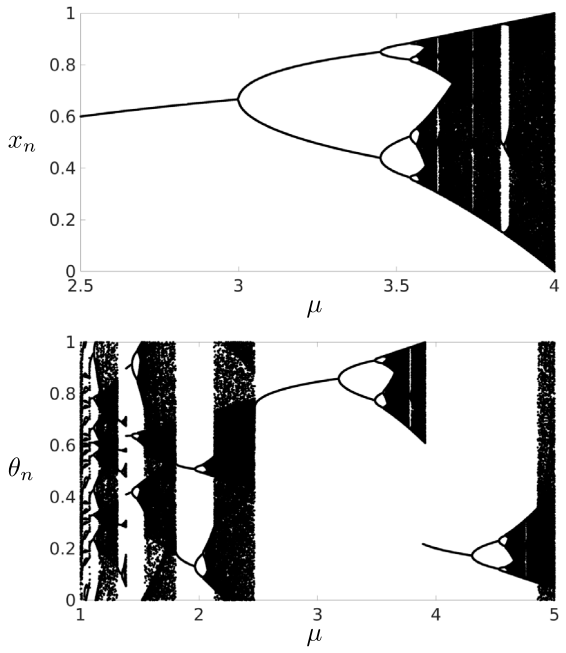

Fig. 3. Bifurcation diagrams of the logistic map (top) and the sine-circle map (bottom).

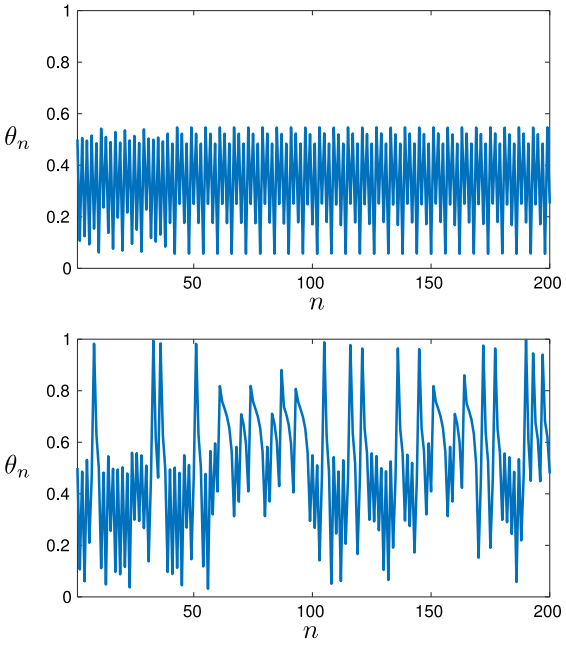

Fig. 4. A periodic (top) and a chaotic (bottom) signal of the sine-circle map of size two hundred with $\mu = 2.1$ and $\mu = 2.3$, respectively.

Lyapunov exponent which is defined as

$$\lambda = \lim_{n\to+\infty} \frac{1}{n} \sum_{i=0}^{n-1} \log |f'(x_i)| \tag{3}$$

for a discrete dynamical system $x_{n+1} = f(x_n)$ and expresses the exponential separation, viz. $d(t) = d_0 e^{\lambda t}$, of two nearby trajectories originally separated by distance $d_0 = \epsilon \ll 1$ at time $t = 0$. The second measure is the Shannon entropy, which uses the probability distribution function of a trajectory to quantify a range of accessible states for a dynamical system and relates to

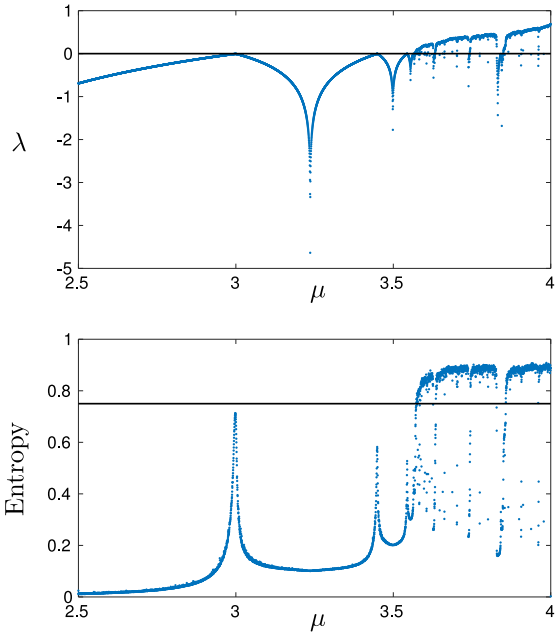

Fig. 5. Lyapunov exponents (top) and Shannon entropy (bottom) of the logistic map. A time series is chaotic if its Lyapunov exponent is greater than zero and its entropy greater than 0.75. The horizontal black lines in the plots indicate these thresholds.

the potential topological transitivity of the system [38, Chap. 9]. Hence, we expect a chaotic system to have well distributed trajectories in space compared to a periodic one and the aim is then to count the number of accessible states for the system. Thus, we define the Shannon entropy of a time series x_n to be

$$S_N = -\frac{1}{\log(N)} \sum_r p_r \log(p_r), \tag{4}$$

where p_r is the probability to be on the state r reached by the system with $p_r = \frac{1}{N}\#\{x_i = r | 1 \le i \le N\}$ and N is the number of sample points. Here, the entropy S_N has been normalised to lie in $[0, 1]$ so that the entropy of a constant signal $S_N \to 0$ while the entropy of a chaotic time series $S_N \to 1$.

We classify a given signal as chaotic when its Lyapunov exponent is strictly positive and its entropy is greater than a given threshold, experimentally set at 0.75 (see also Fig. 5), and non-chaotic otherwise. It is crucial to classify the training data set accurately in order to reduce misclassifications on the testing set. On that note, by using the Shannon entropy in addition to the Lyapunov exponent as a measure of chaos we gained an incremental improvement in accuracy. This is because the Lyapunov exponent was misclassifying some quasi-periodic signals as chaotic.

The Lyapunov exponent and the Shannon entropy of the logistic map as a function of the bifurcation parameter μ are illustrated in Fig. 5. In real applications, computing these quantities over the whole range of parameters and in some cases without knowing the expression of the underlying dynamical system can be unfeasible or computationally expensive, which justifies the approach of using a machine learning algorithm to perform the classification automatically.

The average classification accuracy of the neural networks ShallowNet, MLP, FCN, ResNet, and LKCNN is reported in Table 1. The ShallowNet, MLP, FCN, and ResNet architectures classify signals from the sine-circle map with an accuracy less than 65%. The LKCNN network however seems to override overfitting issues on the training set by capturing the main features of chaotic and periodic signals and gets an average classification accuracy of 89.8%.

Table 1
Classification accuracy on the logistic and sine-circle maps data sets. The neural networks are trained on logistic signals and the accuracy is averaged over five training cycles.

Networks	ShallowNet	MLP	FCN	ResNet	LKCNN
Logistic	99.5	83.4	95.3	96.7	98.8
Sine-circle	64.9	60.2	54.0	44.8	89.8

It is of interest to notice that the shallow neural network reaches an accuracy greater than state-of-the-art time series classification networks on the sine-circle data set despite its simplicity. Improving the accuracy of LKCNN on the sine-circle map might be challenging because this dynamical system leads to signals with behaviour that is absent in the training set of the logistic map (see e.g. the regime $\mu \in [1, 1.3]$ in Fig. 3 (bottom)).

4. Continuous dynamical systems

We now consider continuous dynamical systems of ordinary and partial differential equations that exhibit temporal and spatiotemporal chaos, respectively. The aim here is to determine whether a neural network trained on a low dimensional dynamical system is able to generalise and classify univariate time series generated by a higher dimensional dynamical system. We will first consider the Lorenz system since it is one of the most typical continuous dynamical systems with a chaotic behaviour which has been widely studied in the twentieth century [39].

4.1. Lorenz system

The Lorenz system [40] consists of the following three ordinary differential equations:

$$\dot{x} = \sigma(y - x), \tag{5a}$$

$$\dot{y} = x(\rho - z) - y, \tag{5b}$$

$$\dot{z} = xy - \beta z. \tag{5c}$$

Taking the parameters $\sigma = 10$, $\beta = 8/3$, and varying ρ in $[0, 250]$ yields convergent, periodic, and chaotic solutions. We numerically solve Eq. (5) using MATLAB's function `ode45` with $[x, y, z] = [1, 1, 1]$ as initial condition. Integrating the equations for $t \in [0, 100]$ we obtain time series for $x(t)$, $y(t)$, and $z(t)$ of length one thousand, and we carry out this operation for five thousand values of the bifurcation parameter ρ in the range $[0, 250]$.

The time series $x(t)$, $y(t)$, and $z(t)$ are normalised by the linear transformation $x(t) \mapsto (x(t) - m)/(M - m)$, where M and m are respectively the maximum and minimum of the time series, such that their range are in the interval $[0, 1]$ (see time series in Fig. 6 for $\rho = 70$). Note that normalising the time series is crucial to obtain good generalisation performance because the x, y, and z time series do not have the same range of values. Fig. 7 depicts four time series of the variable $x(t)$ generated by numerically solving Eq. (5) for $\rho = 15$, 28, 160, and 180.

We classify the time series of the Lorenz system as chaotic or non-chaotic according to the sign of the Lyapunov exponent at the corresponding regimes of the bifurcation parameter ρ in order to generate training and testing data sets for the neural networks. Here, we compute the Lyapunov exponents for the time series of the variable $x(t)$ starting from some initial condition $x(0)$ as follows

$$\lambda = \lim_{t \to +\infty} \lim_{\epsilon \to 0} \frac{1}{t} \log \left(\frac{|x(t) - x_\epsilon(t)|}{\epsilon} \right), \tag{6}$$

where $|x(0) - x_\epsilon(0)| < \epsilon \ll 1$. Fig. 8 shows the Lyapunov exponents of the variable $x(t)$, which determine the classification

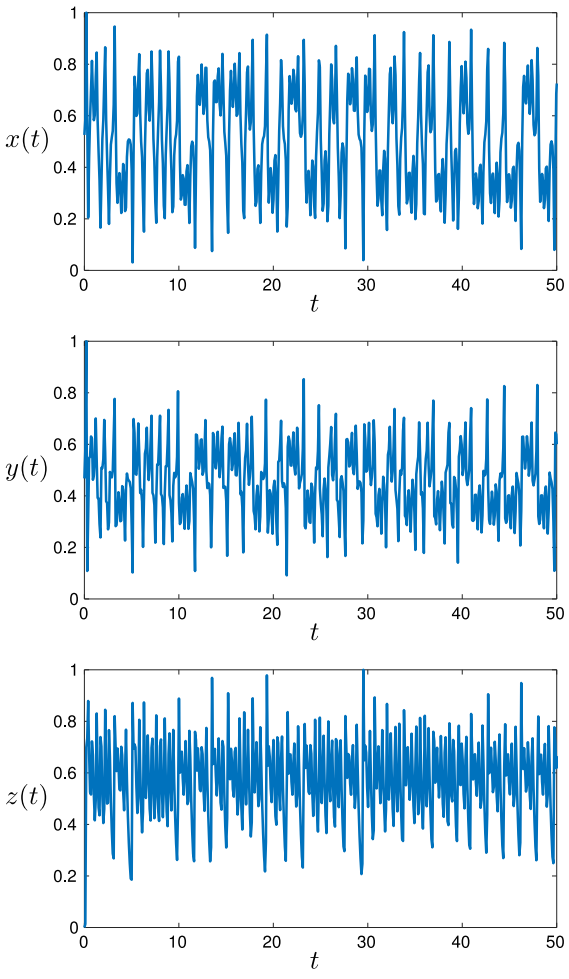

Fig. 6. Normalised time series of the x, y, and z components of the Lorenz system with bifurcation parameter $\rho = 70$.

of the testing set of time series given to the neural networks described in Section 2. For example, the chaotic time series plotted in Fig. 7(d) corresponds to a bifurcation parameter of $\rho = 180$ and has a strictly positive Lyapunov exponent as shown in Fig. 8. For continuous dynamical systems the Shannon entropy did not appear to be a precise measure of chaotic behaviour. In fact, Shannon entropy requires a threshold to determine if a given time series is chaotic or not. Setting a good threshold might not always be possible and requires a lot of experimentation. Moreover, in this case, the distribution of values of the time series is continuous, giving a broad probability distribution function, which makes the task of choosing a threshold troublesome. Therefore, we do not consider it to classify the time series of the Lorenz system.

The different neural networks are trained on time series of the x component of the Lorenz system and tested on the y and z components. These three data sets are composed of two thirds of chaotic time series. Similarly to the logistic map data set (see Section 3), the x component set is divided in the following way: two thirds for training and one third for testing. We then compare the classification accuracy of the networks described in Section 2 on the two data sets. The results are presented in Table 2.

The convolutional neural network LKCNN (see Section 2.5) outperforms the other networks on all the testing sets composed of time series of the x, y, and z components of the Lorenz system. In particular, it is able to generalise well on the z component by determining whether a given time series is chaotic or not correctly with an average accuracy of 78.2%. The other neural

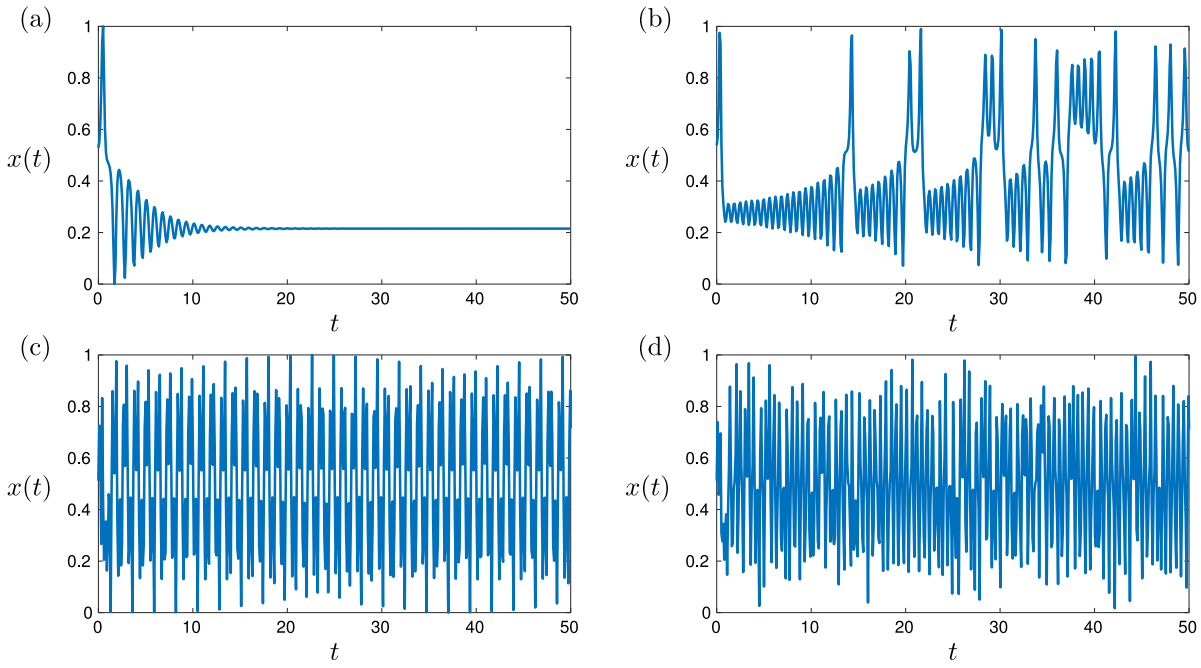

Fig. 7. Normalised time series of the x component of the Lorenz system with bifurcation parameter $\rho = 15$ (a), 28 (b), 160 (c), and 180 (d).

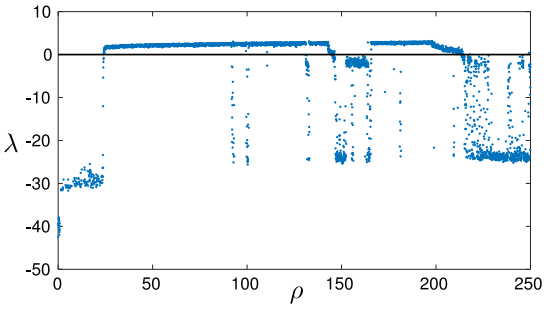

Fig. 8. Lyapunov exponents of the $x(t)$ component of the Lorenz system for $\sigma = 10$, $\beta = 8/3$, and $\rho \in [0, 250]$. A positive Lyapunov exponent (points above the horizontal black line) indicates a chaotic solution to the Lorenz equations.

Table 2
Classification accuracy on the Lorenz system. The network is trained on x component of the Lorenz system and the accuracy is averaged over five training cycles.

Networks	ShallowNet	MLP	FCN	ResNet	LKCNN
Lorenz X	98.5	90.2	80.3	88.8	98.6
Lorenz Y	75.9	75.5	74.6	73.6	95.4
Lorenz Z	58.2	54.9	65.5	45.8	78.2

networks seem to overfit the training set and fail to classify time series of the z component correctly. Note that the y component of the Lorenz system is highly correlated with the x component, unlike the z component (see Fig. 6), which explains the relative good classification accuracy (around 75%) of all the neural networks on the y component.

4.2. Kuramoto–Sivashinsky equation

In this section, we consider the Kuramoto–Sivashinsky (KS) equation, which is an example of a fourth-order nonlinear partial differential equation, which exhibits spatiotemporal chaos. This equation was originally derived by Kuramoto [41–43] and Sivashinsky [44–46] to model instabilities in laminar flame fronts and arises in a wide range of physical problems such as plasma physics [43], flame propagation [44], or free surface film flows [47–49]. In particular, we study the Kuramoto–Sivashinsky system normalised to the interval $[0, 2\pi]$:

$$\begin{aligned} &u_t + 4u_{xxxx} + \alpha\left[u_{xx} + \frac{1}{2}(u_x)^2\right] = 0, \\ &u(x,0) = u_0(x), \quad u(x+2\pi, t) = u(x,t), \end{aligned} \tag{7}$$

where $x \in [0, 2\pi]$, $t \in \mathbb{R}^+$, and α is the bifurcation parameter.

We refer to the study of the attractors by Hyman and Nicolaenko [50] and follow the approach of Papageorgiou and Smyrlis [51,52] by considering the initial condition $u_0(x) = -\sin(x)$ to ensure that the integral of the solution over the spatial domain vanishes. Varying the bifurcation parameter α in Eq. (7) yields a wide range of attracting solutions such as periodic, bimodal, travelling wave, or chaotic, numerically studied in [50].

We spatially discretise Eq. (7) using the Fourier spectral method with the 2/3 dealiasing rule [53] and temporally using the ETDRK4 scheme of Cox and Matthews [54]. We use the stiff partial differential equation integrator [55] in the Chebfun software [56] with a spectral resolution of 512 and a time step of 2.5×10^{-4} to numerically solve Eq. (7) for $t \in [0, 10]$. The regimes we considered are listed below based on the values of the bifurcation parameter α:

1. One hundred values of α are uniformly distributed in each of the following intervals: $[18, 22]$, $[23, 33]$, $[43, 45]$, $[56, 65]$, $[95, 115]$. These intervals are chosen to cover a wide range of behaviours according to [50].
2. Five hundred values of α are uniformly distributed in $[120, 130]$.

This leads to a data set of one thousand realisations, equally divided between chaotic and non-chaotic behaviour.

Fig. 9 shows oscillatory solutions to the KS equation for $\alpha = 20$ (a), 44 (b) and a quadrimodal solution (c). A chaotic solution to the KS equation is depicted in Fig. 9(d). This spatiotemporal chaotic behaviour is hard to analyse because of the high dimensionality of the system, i.e. the large number of Fourier modes of

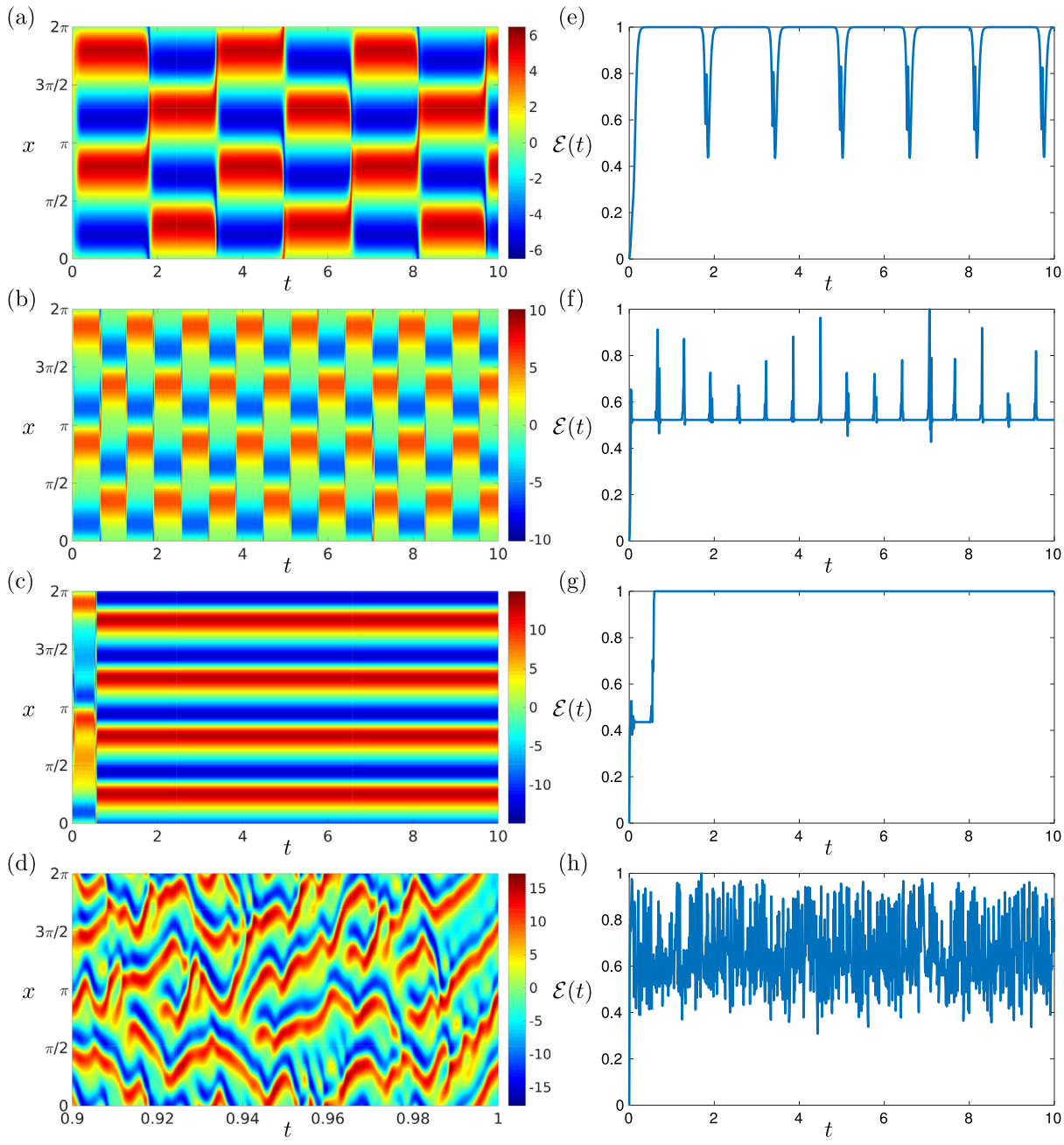

Fig. 9. Solutions to the KS equation with $\alpha = 20$ (a), 44 (b), 100 (c), and 125 (d). The right panels show the corresponding normalised energy $\mathcal{E}(t)$. The chaotic solution depicted in (d) has been zoomed to $t \in [0.9, 1]$. (c) and (g) illustrate the transient regime of the solution for $t \in [0, 0.5]$ before convergence to the global quadrimodal attractor.

the solutions. Thus, we analyse the behaviour of the solutions by considering the energy time series

$$\mathcal{E}(t) = \int_0^{2\pi} u(x,t)^2 \, dx, \tag{8}$$

normalised by the transformation $\mathcal{E}(t) \mapsto (\mathcal{E}(t) - m)/(M - m)$, where

$$m = \min_{t\in[0,10]} \int_0^{2\pi} u(x,t)^2 \, dx,$$

$$M = \max_{t\in[0,10]} \int_0^{2\pi} u(x,t)^2 \, dx,$$

such that it lies in the interval $[0, 1]$. The normalised energy time series of the solutions to the KS equation for $\alpha = 20$, 44, 100, and 125 is plotted in Fig. 9(e)–(h), respectively. These figures illustrate the relation between $\mathcal{E}(t)$ and the behaviour of the solution $u(x, t)$.

Similarly to Section 4.1, we train the large kernel convolutional neural network described in Section 2.5 on the x component of the Lorenz system and test it on the time series from the data set of the KS equation described above. The global accuracy that we obtain to classify the time series between chaotic and non-chaotic is 94.4%. The accuracy for the different classes of attracting solutions in the testing set is reported in Table 3.

We observe that the LKCNN classifies correctly time series of bimodal, highly oscillatory, trimodal, and quadrimodal solutions, corresponding to $\alpha \in [23, 33]$, $[43, 45]$, $[56, 65]$, $[95, 115]$, as non-chaotic with an accuracy of 96.4%, 99.2%, 95.4%, and 99.6%, respectively. Moreover, the network achieves 99.8% accuracy on

Table 3
Classification results of the energy time series of the KS equation for various α. The neural network LKCNN is trained on the x component of the Lorenz system. The classification accuracy is reported for the different intervals of α composing the data set and averaged over five training cycles.

Range of α	Solutions behaviour	Accuracy
[18, 22]	Periodic	54.8
[23, 33]	Bimodal	96.4
[43, 45]	Periodic	99.2
[56, 65]	Trimodal	95.4
[95, 115]	Quadrimodal	99.6
[120, 130]	Chaotic	99.8

the set of chaotic time series. However, the energy time series of low-frequency periodic solutions to the Kuramoto–Sivashinsky equation for $\alpha \in [18, 22]$ are misclassified by the neural network since only 54.8% of them are identified as non-chaotic. We expect this misclassification to be due to qualitative differences between the corresponding energy time series of the KS equation and the periodic time series of the Lorenz system. In particular, the KS data set contains periodic time series with low frequency oscillations in this regime (see Fig. 9(b)), while the Lorenz system generates periodic time series with high frequency oscillations (see Fig. 7(c)). The neural network is then unable to classify features that are not present in the training set and hence fails to generalise to the low frequency periodic time series of the Kuramoto–Sivashinsky equation.

However, we identified two regimes (i.e. $\rho \in [140, 160]$ and $\rho \in [200, 220]$) for which the z component time series of the Lorenz system are low frequency. Therefore, we trained LKCNN on the z component of the Lorenz system over the whole range of parameters $\rho \in [0, 250]$ and tested it on the KS system. Then, we find that the low frequency time series of the KS system (corresponding to $\alpha \in [18, 22]$) are now classified correctly with an accuracy of 95.6%, (instead of 54.8% when the network is trained on the x component of the Lorenz system). Moreover, the overall accuracy improves to 98.7% instead of 94.4%.

4.3. Accuracy dependence on the training data set size and the time series length

We test the robustness of the neural network LKCNN on the classification problem of the KS equation by studying how the accuracy depends on the size of the training data set and the length of the time series. Remember that the network is trained on time series of the x component of the Lorenz system of same length, whose chaotic classification is obtained using the Lyapunov exponent.

Fig. 10 shows the effect of the size of the training data set on the classification accuracy of LKCNN. We observe that the neural network achieves a median accuracy greater than 90% when the amount of training data available is above 10%. Note that this 10% corresponds to five hundred time series from the whole data set of the Lorenz system. This high accuracy is due to the accurate classification of the neural network on most regimes of the time series of the data set (see Table 3). It is of interest to notice that the accuracy is slightly affected only for the first quartile above 10% of the size of the training set. The fluctuations of the accuracy in Fig. 10 are explained by the difficulty to classify low frequency time series of the KS data set.

In Fig. 11, we analyse the classification ability of LKCNN on shorter time series. It is interesting that for all the lengths of the time series we considered, LKCNN reaches a median accuracy greater than 90% on the KS problem. This indicates that the sign of the Lyapunov exponent even for short time series of length three hundred of the training set is determined as accurate as for time

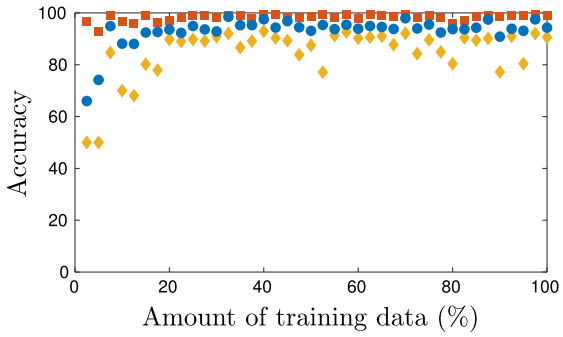

Fig. 10. Classification accuracy of the KS data set versus the amount of training data (percentage from five thousand realisations). We show the first quartile (◇), the median (○), and the third quartile (□) of the accuracy obtained by 20 training cycles of the LKCNN neural network on time series of the x component of the Lorenz system.

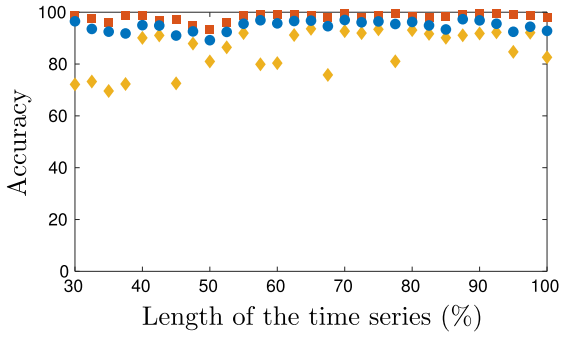

Fig. 11. Classification accuracy of the KS time series with respect to the length of the time series (percentage from one thousand timesteps). We show the first quartile (◇), the median (○), and the third quartile (□) of the accuracy obtained by 20 training cycles of the LKCNN neural network on time series of the x component of the Lorenz system.

series of length thousand. Consequently, this result suggests that the Lyapunov exponent is a good measure of chaotic behaviour in the Lorenz system. The fact that we train and test the network on time series of the same length might also be a justification of this high accuracy. On the other hand, the fluctuations of the first quartile show the inability of the network to classify the time series of the KS data set in a few cases. Note that we have not considered time series of lengths less than three hundred (or 30% of the time series length). This is because LKCNN includes two convolutional layers of kernel size one hundred.

Overall, our results show the robustness of LKCNN on this problem. In particular, this network is able to generalise well on time series generated by the KS equation and achieves a median classification accuracy greater than 90%, independently of the size of the training data set or the length of the time series.

Conclusions

For some of the most challenging real-life applications the expression of a precise underlying dynamical system is unknown or the phase space of the system is infinite dimensional, which makes the identification of the different dynamical regimes unfeasible or in the best case scenario computationally expensive. For this reason, in this study we have introduced a deep learning approach for classifying univariate time series generated by discrete and continuous dynamical systems. Our approach is to train standard neural networks on a given dynamical system with a basic or low-dimensional phase space and generalise by using this

network to classify univariate time series of a dynamical system with more intricate or high-dimensional phase space.

The convolutional neural network with large kernel size (LKCNN) is able to learn the chaotic features of these dynamical systems and classify their time series with high accuracy. On the other hand, state-of-the-art neural networks tend to overfit the training data set and do not perform as well. In detail, our generalisation approach has been applied to classify univariate time series generated by the sine-circle map and the Kuramoto–Sivashinsky equation, using the logistic map and the Lorenz system as training data sets, respectively. We observed a classification accuracy greater than 80% on the sine-circle map and more than 90% on the KS equation. This great performance can be justified from the fact that there are no batch normalisation layers in this network and thus we avoid rescaling the testing data with the wrong normalisation parameters in contrast with FCN and ResNet. Moreover, we observed that the classification accuracy of the KS time series increased further when low frequency time series were involved in the training phase. We also found that LKCNN is able to generalise with high classification accuracy even when the size of the training data set or the length of the time series is very small. This demonstrates the robustness of this neural network in this problem.

Finally, our study suggests that deep learning techniques, which can generalise the knowledge acquired from a training set to a different testing set, can be valuable to classify time series into chaotic and non-chaotic obtained from real-life applications. There are many directions in which the present results can be pursued further. First of all, attempting to classify time series obtained from real-life applications is crucial. In that respect, the effect of noise in the training and testing data sets is an important aspect to be considered and study the influence of the noise to the accuracy of the networks to classify the time series. We hope to address such an analysis in our future work.

Acknowledgements

The authors would like to thank Samuel Cohen for useful discussions and Nicos Pavlidis for providing useful comments on an early draft of this manuscript. We thank the two anonymous reviewers for their valuable comments and suggestions that improved our manuscript. This work was supported by the EPSRC, UK Centre For Doctoral Training in Industrially Focused Mathematical Modelling (EP/L015803/1) in collaboration with Culham Centre for Fusion Energy. This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014–2018 and 2019–2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

Appendix. Sensitivity of LKCNN's performance to the kernel size and feature channels

We vary the size of the convolutional kernel and number of feature channels in the large kernel convolutional neural network (LKCNN) to study the sensitivity of the classification accuracy with respect to these parameters. LKCNN is trained on the logistic signals and tested on signals generated by the sine-circle map (see Section 3).

In Fig. A.12, we fix the number of feature channels to five and vary the size of the convolutional kernel from five to a hundred. We obtain a classification accuracy between 80% and 90%. This indicates that the network is not very sensitive to the size of the convolutional kernel, which is usually chosen to be small.

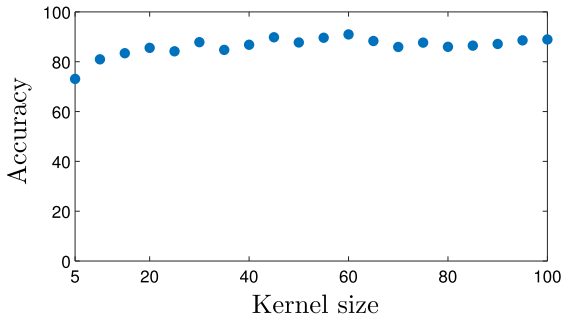

Fig. A.12. Classification accuracy of the sine-circle map signals with respect to the convolutional kernel size. LKCNN is trained on logistic signals with five feature channels and the accuracy is averaged over five training cycles.

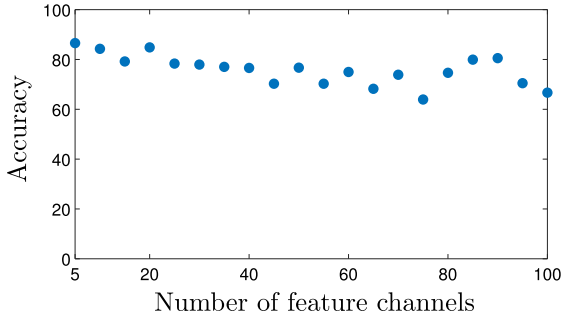

Fig. A.13. Classification accuracy of the sine-circle map signals with respect to the number of feature channels. LKCNN is trained on logistic signals with a kernel size of a hundred and the accuracy is averaged over five training cycles.

Now, we analyse the dependence of the classification accuracy of LKCNN in Fig. A.13 by keeping the kernel size fixed to a hundred and varying the number of feature channels. We observe that the accuracy decreases from 85% to approximatively 65% when the number of channels increases. An explanation of this behaviour is that the number of parameters for LKCNN dramatically increases as we increase the number of feature channels. This makes the training phase computationally more challenging because the optimisation algorithm struggles to converge due to the small number of epochs that we have chosen (see Section 2).

Based on these results, we chose to have large convolutional kernels to decrease the computational expense of the network and small number of feature channels to avoid overfitting the data. With these choices we improve the generalisation ability and the computational performance of the network.

References

[1] M. Casdagli, Nonlinear prediction of chaotic time series, Physica D 35 (3) (1989) 335–356.
[2] Y.-C. Lai, N. Ye, Recent developments in chaotic time series analysis, Int. J. Bifurcation Chaos 13 (6) (2003) 1383–1422.
[3] S. Mukherjee, E. Osuna, F. Girosi, Nonlinear prediction of chaotic time series using support vector machines, in: Proceedings of the 1997 IEEE Signal Processing Society Workshop, IEEE, 1997, pp. 511–520.
[4] G. Sugihara, Nonlinear forecasting for the classification of natural time series, Phil. Trans. R. Soc. A 348 (1688) (1994) 477–495.
[5] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, NIPS, 2012, pp. 1097–1105.
[6] Y. LeCun, Y. Bengio, The Handbook of Brain Theory and Neural Networks, MIT Press, 1998.
[7] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, IEEE, 2015, pp. 1026–1034.

[8] J.B. Elsner, A.A. Tsonis, Nonlinear prediction, chaos, and noise, Bull. Am. Meteorol. Soc. 73 (1) (1992) 49–60.
[9] T. Kuremoto, S. Kimura, K. Kobayashi, M. Obayashi, Time series forecasting using a deep belief network with restricted Boltzmann machines, Neurocomputing 137 (2014) 47–56.
[10] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: International Joint Conference on Neural Networks, IEEE, 2017, pp. 1578–1585.
[11] J. Han, A. Jentzen, E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (34) (2018) 8505–8510.
[12] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[13] E. Weinan, B. Yu, The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[14] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (1) (2018) 932–955.
[15] M. Raissi, G.E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, J. Comput. Phys. 357 (2018) 125–141.
[16] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[17] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614.
[18] J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, Phys. Rev. Lett. 120 (2) (2018) 024102.
[19] J. Pathak, Z. Lu, B.R. Hunt, M. Girvan, E. Ott, Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data, Chaos 27 (12) (2017) 121102.
[20] P. Esling, C. Agon, Time-series data mining, ACM Comput. Surv. 45 (1) (2012) 12.
[21] H.F. Nweke, Y.W. Teh, M.A. Al-Garadi, U.R. Alo, Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges, Expert Syst. Appl. 105 (2018) 233–261.
[22] T.L. Nwe, T.H. Dat, B. Ma, Convolutional neural network with multi-task learning scheme for acoustic scene classification, in: Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, IEEE, 2017, pp. 1347–1350.
[23] G.A. Susto, A. Cenedese, M. Terzi, Time-series classification methods: review and applications to power systems data, in: Big Data Application in Power Systems, Elsevier, 2018, pp. 179–220.
[24] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.
[25] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[26] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. (2019) 1–47.
[27] D.L. Olson, D. Delen, Advanced Data Mining Techniques, Springer Science & Business Media, 2008.
[28] K.H. Brodersen, C.S. Ong, K.E. Stephan, J.M. Buhmann, The balanced accuracy and its posterior distribution, in: 20th International Conference on Pattern Recognition, IEEE, 2010, pp. 3121–3124.
[29] M.F. Møller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Netw. 6 (4) (1993) 525–533.
[30] Z. Wang, GitHub repository, 2019, https://github.com/cauchyturing/UCR_Time_Series_Classification_Deep_Learning_Baseline.
[31] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2015.
[32] M. Lin, Q. Chen, S. Yan, Network in network, in: ICLR, 2014.
[33] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[34] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT press, 2016.
[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[36] R.M. May, Simple mathematical models with very complicated dynamics, Nature 261 (1976) 459–467.
[37] S.H. Strogatz, Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering, Westview Press, 2014.
[38] R.C. Hilborn, Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers, Oxford University Press on Demand, 2000.
[39] C. Sparrow, The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors, Vol. 41, Springer Science & Business Media, 2012.
[40] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141.
[41] Y. Kuramoto, Diffusion-induced chaos in reaction systems, Prog. Theor. Phys. Suppl. 64 (1978) 346–367.
[42] Y. Kuramoto, T. Tsuzuki, On the formation of dissipative structures in reaction-diffusion systems: Reductive perturbation approach, Progr. Theoret. Phys. 54 (3) (1975) 687–699.
[43] Y. Kuramoto, T. Tsuzuki, Persistent propagation of concentration waves in dissipative media far from thermal equilibrium, Progr. Theoret. Phys. 55 (2) (1976) 356–369.
[44] G.I. Sivashinsky, Nonlinear analysis of hydrodynamic instability in laminar flames–I. Derivation of basis equations, Acta Astronaut. 4 (11–12) (1977) 1177–1206.
[45] G.I. Sivashinsky, On flame propagation under conditions of stoichiometry, SIAM J. Appl. Math. 39 (1) (1980) 67–82.
[46] G.I. Sivashinsky, Instabilities, pattern formation, and turbulence in flames, Annu. Rev. Fluid Mech. 15 (1) (1983) 179–199.
[47] D.J. Benney, Long waves on liquid films, J. Math. Phys. 45 (1–4) (1966) 150–155.
[48] A.P. Hooper, R. Grimshaw, Nonlinear instability at the interface between two viscous fluids, Phys. Fluids 28 (1) (1985) 37–45.
[49] G.I. Sivashinsky, D.M. Michelson, On irregular wavy flow of a liquid down a vertical plane, Progr. Theoret. Phys. 63 (1980) 2112–2114.
[50] J.M. Hyman, B. Nicolaenko, The Kuramoto–Sivashinsky equation: a bridge between PDE's and dynamical systems, Physica D 18 (1–3) (1986) 113–126.
[51] D.T. Papageorgiou, Y.S. Smyrlis, The route to chaos for the Kuramoto–Sivashinsky equation, Theor. Comput. Fluid Dyn. 3 (1) (1991) 15–42.
[52] Y.S. Smyrlis, D.T. Papageorgiou, Predicting chaos for infinite dimensional dynamical systems: the Kuramoto–Sivashinsky equation, a case study, Proc. Natl. Acad. Sci. 88 (24) (1991) 11129–11132.
[53] S.A. Orszag, On the elimination of aliasing in finite-difference schemes by filtering high-wavenumber components, J. Atmos. Sci. 28 (6) (1971) 1074.
[54] S.M. Cox, P.C. Matthews, Exponential time differencing for stiff systems, J. Comput. Phys. 176 (2) (2002) 430–455.
[55] H. Montanelli, N. Bootland, Solving periodic semilinear stiff PDEs in 1D, 2D and 3D with exponential integrators, 2016, arXiv preprint arXiv:1604.08900.
[56] T.A. Driscoll, N. Hale, L.N. Trefethen, Chebfun Guide, Pafnuty Publications, Oxford, 2014.