Physica D 472 (2025) 134510

Contents lists available at ScienceDirect

Physica D

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

Full Lyapunov exponents spectrum with Deep Learning from single-variable time series

Carmen Mayora-Cebollero [a,*], Ana Mayora-Cebollero [a], Álvaro Lozano [b], Roberto Barrio [a,*]

[a] IUMA, CoDy and Department of Applied Mathematics, Universidad de Zaragoza, Zaragoza, Spain
[b] IUMA, CoDy and Department of Mathematics, Universidad de Zaragoza, Zaragoza, Spain

ARTICLE INFO

Communicated by Victor M. Pérez-García

Keywords:
Deep Learning
Lyapunov exponents
Chaos
Hyperchaos
Dynamics classification
Lorenz system
Coupled Lorenz system

ABSTRACT

In this article we study if a Deep Learning technique can be used to obtain an approximate value of the Lyapunov exponents of a dynamical system. Moreover, we want to know if Machine Learning techniques are able, once trained, to provide the full Lyapunov exponents spectrum with just single-variable time series. We train a Convolutional Neural Network and use the resulting network to approximate the full spectrum using the time series of just one variable from the studied systems (Lorenz system and coupled Lorenz system). The results are quite surprising since all the values are well approximated with only partial data. This strategy allows to speed up the complete analysis of the systems and also to study the hyperchaotic dynamics in the coupled Lorenz system.

1. Introduction

Lyapunov exponents (LEs) are a classical tool to study the behavior of a dynamical system. For example, a positive maximum LE (MLE) indicates chaotic behavior, or hyperchaotic if the second LE is also positive. Moreover, with the second LE, some bifurcations, such as period-doubling bifurcations, can be inferred. One of the standard algorithms for computing LEs can be found in [1], where all variables and the corresponding variationals are used. Other techniques [1,2] use only the time series of one of the system variables, but only the maximum LE is obtained. Deep Learning (DL) techniques have been used to predict directly the LE of one-dimensional discrete dynamical systems [3] or as a complementary tool. In the latter approach, DL is used for time series forecasting or to obtain, via data assimilation, a conjectured dynamical system, and then the Lyapunov spectrum is estimated through classical methods [4–8].

Deep Learning [9,10] includes all the Machine Learning techniques that allow Deep Artificial Neural Networks (architectures built with layers of artificial neurons) to learn from data with several levels of abstraction. The activation of each neuron in a DL architecture (that is, its value) is computed by applying a non-linear activation function to a linear combination of its inputs using some weights and a bias (the trainable parameters of the network). To fit all these parameters (in order to obtain the desired output for each input) a minimization problem is solved (a loss function is minimized respect to these parameters using training data during a process known as training). Data not used in the training stage, known as test data, is used to check that the network has learned correctly and is able to generalize to new samples.

The computation of biparametric analyses using classical methods has allowed to study in detail the global dynamics of numerous systems [11–13]. Any improvement that enables faster and more detailed studies could be useful. In recent years, some authors have proposed to use Deep Learning as a new technique to analyze the behavior of a dynamical system [3–8,14–17]. This new technique can speed up these parametric studies.

In this paper, we apply DL to directly approximate the LEs values of two dynamical systems chosen as test examples: the classic Lorenz system [18] and a coupled Lorenz system [19]. Among all the possible DL architectures, we have chosen the Convolutional Neural Network (CNN) [20]. Additionally, we aim to demonstrate that Deep Learning can approximate all the Lyapunov exponents of a system using only short time series data from a single variable of the dynamical system. We will use the notation LE_i to indicate the i-th Lyapunov exponent, therefore LE_1 corresponds to the maximum Lyapunov exponent, or MLE.

* Corresponding authors.
E-mail addresses: cmayora@unizar.es (C. Mayora-Cebollero), rbarrio@unizar.es (R. Barrio).

https://doi.org/10.1016/j.physd.2024.134510
Received 25 March 2024; Received in revised form 19 December 2024; Accepted 19 December 2024
Available online 28 December 2024

Lorenz system. The Lorenz system [18] is a classic three-dimensional continuous dynamical system given by the system of equations

$$\begin{cases} \dot{x} &= \sigma(y-x), \\ \dot{y} &= -xz + rx - y, \\ \dot{z} &= xy - bz, \end{cases} \tag{1}$$

where (x, y, z) are the system variables and (σ, r, b) are the bifurcation parameters (σ is the Prandtl number, r is the relative Rayleigh number, and b is a positive constant). This is one of the seminal systems of chaotic dynamics. There are numerous papers in the literature [11,12] where its behavior is described using classical LEs algorithms.

Coupled Lorenz system. To increase the dimensionality of the test problem, we use two coupled Lorenz systems as in [19]:

$$\begin{cases} \dot{x}_1 &= \sigma(y_1 - x_1), \\ \dot{y}_1 &= -x_1 z_1 + r_1 x_1 - y_1 + \lambda_1(x_2 - y_2), \\ \dot{z}_1 &= x_1 y_1 - b z_1, \\ \dot{x}_2 &= \sigma(y_2 - x_2), \\ \dot{y}_2 &= -x_2 z_2 + r_2 x_2 - y_2 + \lambda_2(x_1 - y_1), \\ \dot{z}_2 &= x_2 y_2 - b z_2, \end{cases} \tag{2}$$

where $(x_1, y_1, z_1, x_2, y_2, z_2)$ are the system variables, $(\sigma,\ r_1,\ r_2, b)$ are the bifurcation parameters, and (λ_1, λ_2) are the coupling parameters. In what follows, we set $r_1 = r$ and $r_2 = r + 10$. A dynamical study of the attractors of such system in the case of coupled equal Lorenz systems ($r_1 = r_2$) can be found in [19].

This paper is organized as follows. In Section 2, we briefly introduce the CNN architecture. In Section 3, we focus on the LEs prediction task for the Lorenz system. In Section 4, we show the results obtained in the LEs prediction of the coupled Lorenz system. Finally, in Section 5 we draw some conclusions.

All DL experiments in this work have been performed using PyTorch [21]. The code has been run on a Linux box with dual Xeon ES2697 with 128Gb of DDR4-2133 memory with a RTX2080Ti GPU.

2. DL techniques for Lyapunov exponents approximation

Deep Learning [9,10] is the branch of Machine Learning that uses Deep Artificial Neural Networks to learn from data with several levels of abstraction. These Artificial Neural Networks (ANNs) are formed by artificial neurons (loosely inspired by their biological counterparts) that are organized in layers.

In a previous paper [14], we focused on detecting chaotic behavior in a dynamical system, but now we also want to quantify it and be able to approximate the full Lyapunov exponents spectrum using single-variable time series. In [14] we used three ANN technologies: Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN) and Long Short-Term Memory network (LSTM). Here, we only use the CNN as it seems to work well for this task. In the design of the CNN, we use a not very complicated structure explained in Section 2.1. We are mainly interested in studying the reliability of the methodology rather than finding the best architecture, so we only present a brief hyperparameter optimization study in Section 2.1.1.

CNNs are a type of ANN originally developed for image recognition [20]. They are organized into convolutional and pooling layers that capture features and reduce dimensions, respectively. These convolutional layers have different channels (also known as feature maps), each one containing different information (same idea as in RGB images which contain three channels: channel R for red color information, G for green, and B for blue). One of the main characteristics of CNNs is that neurons are not connected to all neurons in the previous layers as it occurs with other ANNs like the MLP. The receptive field of a neuron is the (reduced) set of neurons that send information to it. Another useful property of CNNs is that they can handle different input formats (vectors, matrices, ...) depending on the type of convolution used. In this paper, the input data is in vector form, so we use the so-called 1D

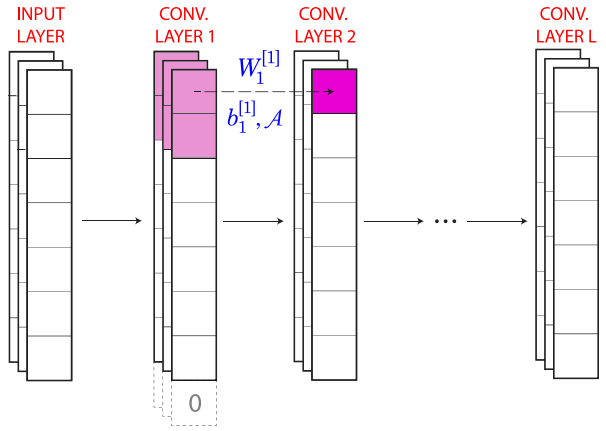

Fig. 1. Schematic representation of a 1D CNN architecture with three channels in the input and depicted convolutional (conv.) layers. Zero-padding is added to convolutional layer 1 to maintain the size for convolutional layer 2 (dashed neurons with a 0 in the middle).

CNNs (see Fig. 1 for a graphical example), and in what follows, the explanations will be particularized to this type of CNN.

The value (activation) of a neuron of a 1D CNN is calculated as follows (the first index for neurons, channels and convolutional layers is 1, and layer 0 refers to the input layer):

$$x_{i,j}^{[l]} = \mathcal{A}\left(b_j^{[l]} + \sum_{k=1}^{K^{[l]}} \sum_{c=1}^{C^{[l-1]}} w_{k,c,j}^{[l]}\, x_{k+(i-1)s^{[l]}+(k-1)(d^{[l]}-1),c}^{[l-1]} \right), \tag{3}$$

where

- $x_{i,j}^{[l]}$ is the activation of neuron i of channel j in layer l.
- $\mathcal{A}$ is the (non-linear) activation function.
- $b_j^{[l]}$ is the bias term shared by all neurons of channel j in layer l.
- $W_j^{[l]} = (w_{k,c,j}^{[l]})_{k=1,\ldots,K^{[l]};c=1,\ldots,C^{[l-1]}}$ is the weight matrix (also called filter or kernel) for neurons of channel j in layer l. Notice that the weights are shared among all the neurons in the same channel and layer.
- $K^{[l]}$ is the kernel size, i.e., the dimension of the receptive field of neurons in layer l.
- $C^{[l]}$ is the number of channels in layer l.
- $s^{[l]}$ denotes the stride, that is, the step size between the receptive field of neuron i of channel j in layer l and the receptive field of neuron $i+1$ of the same channel and layer.
- $d^{[l]}$ refers to the dilation, that is, the distance between neurons of the same receptive field.

In the special case where we want channels in consecutive layers to have the same dimension (i.e., same number of neurons), padding can be applied. A common type is zero-padding where zeros are added around the previous layer.

In the case of Fig. 1, to calculate the activation of the dark magenta neuron (whose receptive field includes the light magenta neurons) Eq. (3) is used with $i = 1$, $j = 1$, $l = 2$, $K^{[2]} = 2$, $C^{[1]} = 3$, $s^{[2]} = 1$ and $d^{[2]} = 1$. Moreover, note that zero-padding has to be applied to have the convolutional layers 1 and 2 with the same number of neurons (see the dashed neurons with a 0 in the middle).

For more details see [22], and the animations referenced there for examples of stride, dilation and padding concepts, and [23] for a discussion of some properties of CNNs.

2.1. CNN architecture for LEs approximation

The CNN architecture we use for the full LEs spectrum approximation is based on the one in [14] used for chaos detection (a classification

task from a DL point of view). The network has only one input channel in the input layer since only single-variable time series are used as known information. It has two convolutional layers: the first one with 15 channels, kernel size 10, stride 1 and dilation equal to 2; and the second one with 30 channels, kernel size 5, stride 1 and dilation 4. The Rectified Linear Unit (ReLU) activation function is applied after adding a bias term in each convolutional layer. Zero-padding is used to ensure that the length of the input sequences remains constant throughout the convolutional layers since the stride is 1. A global average pooling layer is applied following the last convolutional layer to prepare the data for the subsequent linear layer. The aforementioned linear layer has a number of neurons equal to the system's dimension—each corresponding to a Lyapunov exponent value—and a bias term. No activation function is applied to this output layer.

We train the network using the Adam algorithm with a learning rate of 0.008 and apply L^2-regularization with a weight decay of 10^{-5} to prevent overfitting and enhance generalization. In this work, we minimize the Huber loss function [24,25] given by

$$\text{Huber Loss} = \frac{1}{N}\sum_{j=1}^{N} h_j,$$

$$\text{with } h_j = \begin{cases} 0.5\,(\hat{y}_j - y_j)^2 & \text{if } |\hat{y}_j - y_j| < \delta, \\ \delta\,(|\hat{y}_j - y_j| - 0.5\,\delta) & \text{otherwise,} \end{cases}$$

where N is the batch size (dimension of the subsets into which the datasets involved during training are subdivided), y_j refers to the labels of the dataset, and $\hat{y}_j$ to the output of the ANN. The hyperparameter δ is set to 0.6. The Huber loss is less sensitive to outliers than the Mean Squared Error (MSE) loss (usually used in prediction tasks). In the Huber loss the advantages of the MSE loss and the L^1-loss are combined. We expect that this fact will allow the CNN to focus more on fitting values close to zero than large values (note that a large error in predicting a zero LE may lead to an incorrect detection of its behavior). The number of epochs (that is, the number of times the training dataset is revisited during training) is 2000. An early stopping technique [26] is applied, so the trained network used is the one with the weights and biases values that give the lowest Huber loss value for the validation dataset (data different to training dataset that is used to avoid problems as overfitting) during training.

In this paper, the numerical values of the Huber loss will be given as [mean $\pm$ standard deviation] for 10 randomly initialized networks. This will show that the performance of the DL process does not depend on the initialization of the trainable parameters (weights and biases). To conclude that the DL training has been successful, we will verify that the mean and standard deviation are small (we have set a threshold of 0.3 for the mean, and 0.05 for the standard deviation) for all the datasets involved (training, validation and test sets).

2.1.1. A brief hyperparameter optimization study

As indicated before, we are mainly interested in the reliability of the methodology rather than finding the best architecture for the LEs approximation task. However, we perform a brief analysis to show how the network size/number of layers can affect the performance in the LEs approximation task. To do so, we consider two other networks derived from the described above (call it `LE-Network`):

- `Less Layers-Network`: It has only one convolutional layer with the same architecture as the second convolutional layer of `LE-Network`.
- `More Layers-Network`: It is like `LE-Network` but with a third convolutional layer with the same architecture as the last of `LE-Network`.

To compare the performance of each network we carry out the analyses on the Lorenz system trained with non-random data (see Section 3.1). Notice that the network used in all the systems and data approaches in this article is the same, so we consider appropriate to show the dependence on network size/number of layers in the first example (Lorenz system with non-random data).

Table 1 shows the results of the brief hyperparameter optimization study. We can see that if less layers are used (`Less Layers-Network`), the mean loss values of all the datasets are larger than the corresponding ones for `LE-Network`. If more layers are considered (`More Layers-Network`), the mean loss values are lower than those of `LE-Network`, but the standard deviation values are larger on both training and test cases. Moreover, the increase in the number of weights and biases is considerable taking into account the small improvement (more weights and biases usually correspond to more training time). Therefore, according to this brief study, the two convolutional layers `LE-Network` seems to be the appropriate one.

3. LEs approximation for the Lorenz system

In this section, two approaches are presented to approximate LEs with DL in the Lorenz system. In the first approach, the CNN is trained (and validated) using information from a few r-parametric lines (all parameter values of the dynamical system are fixed except for parameter r, whose value is varied in a given interval, and time series are computed for each value of r), and its performance is tested. In the second approach, the same network architecture is trained (and validated) from scratch using time series with random parameter values from an (r, b)-parametric plane (the value of parameter σ is fixed and the values of parameters r and b are varied in given intervals, and time series are computed for each combination of values). These two approaches will show that DL techniques can be used to expand a partial classical study of the system (first approach) or to perform the analysis from random data (second approach). The advantages and disadvantages of each approach are also compared.

In this section, unless otherwise stated, time series and LEs obtained with classical techniques are computed as follows. The time series are obtained using the DOPRI5 integrator (a well-known Runge–Kutta of order 5) with initial conditions $(x, y, z) = (1, 1, 1)$, but only the x-time series and the full LEs spectrum will be used. For more precision, a transient integration is performed up to time $t = 100{,}000$ with time step 0.01, then the integration is continued for 10,001 time units with time step 0.001. The LEs obtained with classical techniques (which are used as the ground truth in the DL process) are computed during this last integration. The time series used as input by the network are built with 1 out of every 100 of the last 100,000 computed points. If two time series p_1 and p_2 are near, that is, $\|p_1 - p_2\|_\infty < 10^{-4}$, one of them is removed. The remaining ones are normalized (x-coordinate is linearly normalized by mapping its range to the interval $[0, 1]$; if the time series is constant in time, a constant random value between 0 and 1 is assigned).

3.1. Non-random data

For this approach, the available data belongs to four r-parametric lines with $b \in \{2, 2.4, 8/3, 2.8\}$ ($\sigma = 10$ for all lines). For each line we consider 6000 different values of $r \in (0, 300]$, which makes a total of 24,000 time series (some samples will be removed to avoid similar time series, and the remaining ones will be normalized). From the samples satisfying $b \in \{2, 8/3\}$, 8000 are randomly chosen for the training set (batch size 128). From the data in the r-parametric line with $b = 2.4$, we select 2000 random points for validation (batch size 100). Finally, 2000 random samples from the set $b = 2.8$ are used for the test set (batch size 100). Note that, during the training process, only data from three lines is used: two lines correspond to training (see light green lines in top-left panel of Fig. 3) and one to validation (see dark green line in the aforementioned figure). The network architecture is the one presented in Section 2.1.

The resulting value of the Huber loss for the training dataset is $[0.049 \pm 0.009]$ (remember that numerical results of the DL performance

Table 1
Brief hyperparameter optimization study performed to show the dependence on the network size/number of layers of a CNN in the LEs approximation task. The analyses are performed in the Lorenz system with non-random data. *Network size* is the number of trainable parameters (weights and biases) of each network.

Network	Network size	Training loss	Validation loss	Test loss
`Less Layers-Network`	273	0.087 ± 0.006	0.131 ± 0.006	0.122 ± 0.007
`LE-Network`	2538	0.049 ± 0.009	0.118 ± 0.004	0.113 ± 0.012
`More Layers-Network`	7068	0.036 ± 0.015	0.110 ± 0.004	0.110 ± 0.013

are given as [mean ± standard deviation] for 10 randomly initialized networks). The corresponding values for the validation and test datasets are $[0.118 \pm 0.004]$ and $[0.113 \pm 0.012]$, respectively. All mean values are close to zero and standard deviations are small. The value of the loss function on the test set is slightly larger than that on the training set, however, we consider that there is not a large enough difference to discard the network due to overfitting. The data used for the analyses with the trained CNN are just time series of the x-variable of length 1000.

In Section 3.1.1, a 1D analysis is performed to show that the trained network is able to generalize to an r-parametric line on which it was not trained. In Section 3.1.2, the analysis is extended to an (r, b)-parametric plane.

3.1.1. 1D analysis

In Fig. 2, the trained CNN has been used for LE approximation in one r-parametric line with $\sigma = 10$ and $b = 2.2$ (6000 equidistant r-values in the range $(0, 300]$ are considered) parallel to the ones used to create the training, validation and test datasets. LE_1 is shown in the top panel, LE_2 can be found in the middle panel, and finally, LE_3 is in the bottom panel. In each panel, the ground truth of the LEs (obtained with the algorithm in [1]) is in black, the mean of the predicted values with the 10 networks is in red, and the interval [mean ± standard deviation (std)] obtained with the prediction of the 10 networks is in green. The obtained value of the Huber loss is equal to $[0.091 \pm 0.005]$. As already explained, the prediction is performed with 10 networks trained with the same architecture and data, but with different initialization of trainable parameters (weights and biases of the ANN). Since the mean value of the loss and the standard deviation are small, we can conclude that the prediction is good enough.

At first glance at the results in Fig. 2, it can be seen that the parts where DL prediction seems to fail the most are those shaded in orange and purple. The orange one corresponds to orbits that converge to equilibrium points (EPs). If we analyze our time series normalization rule we can deduce that this failure is expected: the time series of equilibrium points are normalized to a random value between 0 and 1, so the CNN cannot extract information from them to predict the LE value (it is remarkable that the network assigns almost a constant value to the LEs of all equilibrium point time series, so it has detected this kind of behavior). As the LEs of an equilibrium point do not provide useful information beyond its negative sign in the first exponent (which is correctly predicted), we consider that the DL prediction is successful. The purple part corresponds to a region where long transient chaotic dynamics occurs [11,12,27], and therefore, as we consider quite short time series as data, it is logical that the DL technique can assume chaotic dynamics for them in some cases.

Overall, the CNN that was trained only with x-time series is able to predict the three LE values correctly with a small variability, and failing only in expected regions (as already explained) where it would be necessary to use complementary techniques to obtain good results.

3.1.2. 2D analysis

In Fig. 3 the CNN trained with non-random data has been used for LE prediction in the (r, b)-parametric plane with $\sigma = 10$, $r \in [0, 300]$ and $b \in [2, 3]$ (1000 point values for each varying parameter, which makes a total of 10^6 points). To obtain the time series for this biparametric analysis with the CNN, a transient integration is performed for 1000

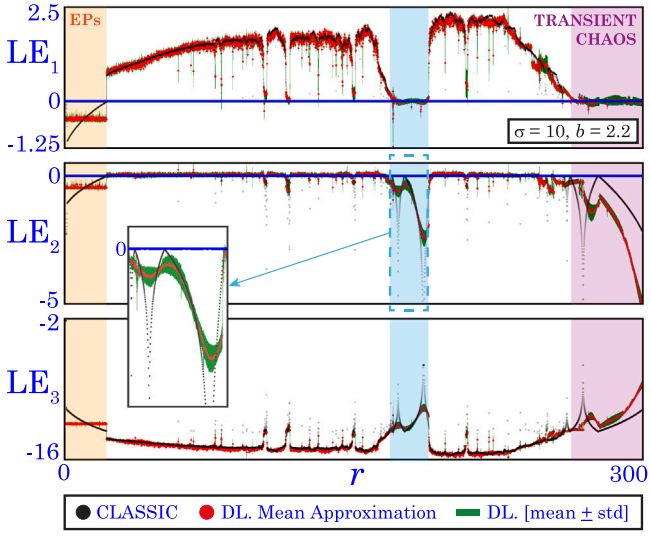

Fig. 2. 1D analysis ($\sigma = 10$, $b = 2.2$) of the Lyapunov exponents in the Lorenz system when training the CNN with non-random data (Huber loss value $[0.091 \pm 0.005]$). The LEs obtained with classical techniques are in black, and the results given by DL are in red (mean value of the predicted values with the 10 networks) and green (interval [mean ± standard deviation (std)] obtained with the prediction of the 10 networks). The orange and purple shading correspond to the regions where the DL technique seems to give worse results. The blue shaded region is used in a comparison with a subsequent analysis in Fig. 5.

time units and later the integration is continued for 100 more time units (time step 0.01 for the whole integration). The input time series of length 1000 for the CNN are constructed with 1 out of every 10 of the last integration points. Remember that for the classical technique of LEs, transient integration is performed for 100,000 time units (with time step 0.01) and 10,001 more time units (with time step 0.001) are used to compute the LEs. As the LEs are defined as a limit, with this classical technique, it takes a long time to ensure a good approximation of the LEs. Therefore, with DL, it takes less integration time to approximate the full LEs spectrum.

In Fig. 3, from left to right, the analyses of LE_1, LE_2 and LE_3 are depicted. In the first row, the results obtained with the classical technique in [1] are represented, the second row corresponds to the DL prediction, and in the third row the signed difference between the value given by classical techniques and DL is studied. To obtain the plots in first and second rows, black is assigned to LEs with a value around 0, a gray scale is used for negative values (different gray scales have been used in the colorbars for a better interpretation: in all panels white is assigned to the smallest colorbar value in the negative range and a sufficiently dark gray is assigned to the largest colorbar value), and a warm color gradation is used for positive values. Moreover, a minimum and a maximum value are fixed for each LE (that means that, for example, for the case of LE_1, LEs of magnitude greater than 2.5 are represented with the same color as Lyapunov exponents of magnitude 2.5, those with magnitude less than -1.5 are in the same color as those of magnitude -1.5, and the intermediate values follow the aforementioned gradation). The choice of such minimum and maximum values does not affect the results since it is a usual way of representation for LEs. The

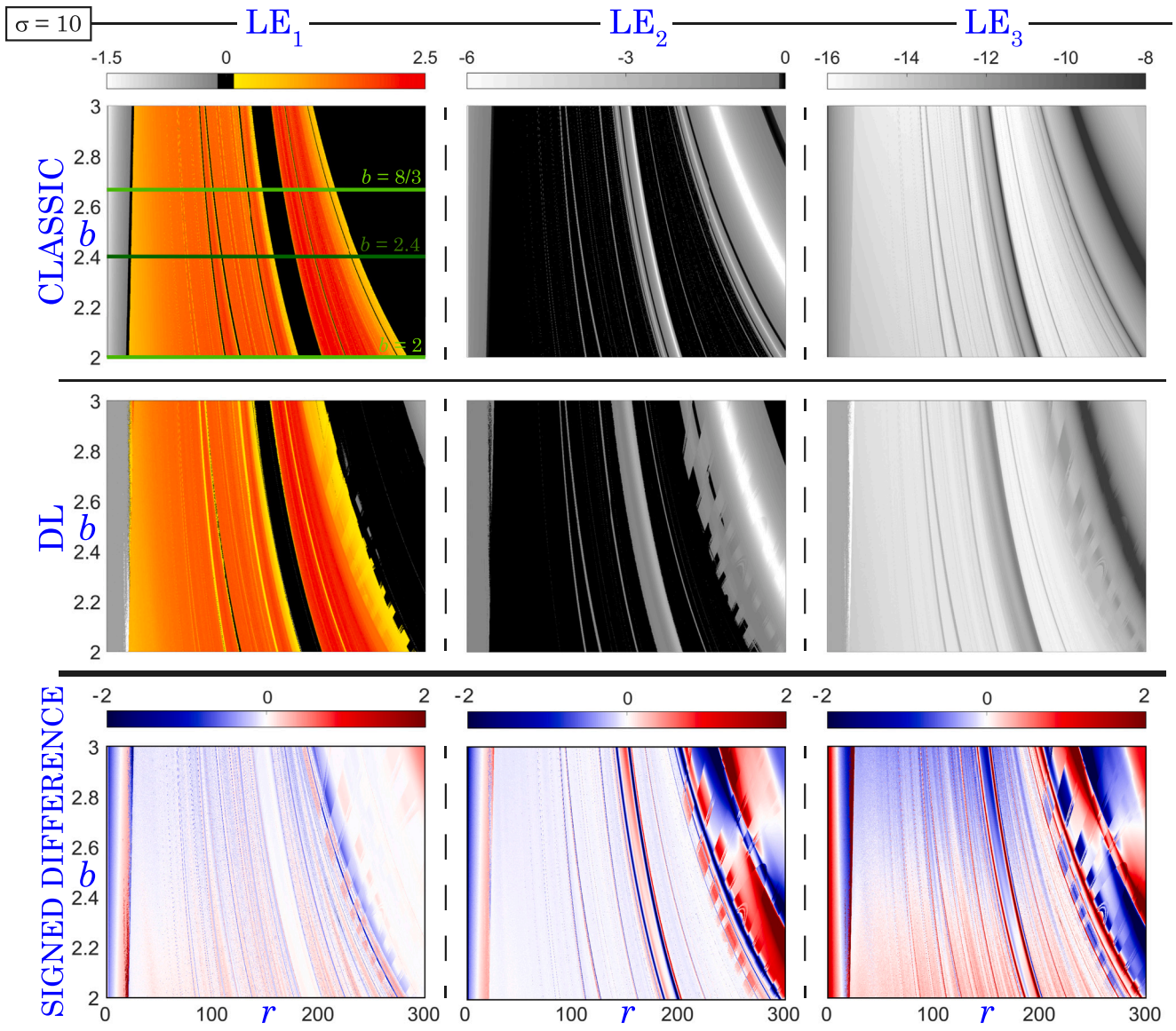

Fig. 3. 2D analysis of Lyapunov exponents in the Lorenz system ($\sigma = 10$) when training with non-random data (Huber loss value $[0.115 \pm 0.005]$). From left to right, LE_1, LE_2 and LE_3 analyses are depicted. From top to bottom, results with classical techniques, with DL techniques, and signed differences between both approximations (classical values minus DL values). The lines in the top-left panel correspond to r-parametric lines from which the training data (light green) and the validation dataset (dark green) are obtained. (See the text for more details.).

third-row plots show the signed difference between the LE obtained with classical techniques minus the approximation obtained with DL. Dark red represents positive differences greater than or equal to 2, fading to white as they approach zero. Blue intensifies with increasing negative differences, with dark blue indicating values less than or equal to −2.

At first glance, comparing the graphical results of first and second rows in Fig. 3 obtained with classical and DL techniques, it can be seen that, even predicting only with the short time series of the first variable x of the Lorenz system, DL is able to reproduce quite well the magnitude of all the LEs. For all three LEs, the regions where the network seems to fail the most are the right boundary of the right chaotic region, and the upper right corner. At this boundary, a long transient chaos occurs. As it can be seen in the third row of Fig. 3, the DL approximations for the first LE are really good (soft colors correspond to positive or negative differences close to 0). For the remaining two LEs, the largest differences (dark blue and dark red) are in the right part (transient chaos) of the biparametric plane. Despite these small areas with non-precise approximations due to the short temporal dynamics of the time series, DL predictions would allow to perform a first dynamical analysis of the represented biparametric plane providing useful qualitative and quantitative approximations of the LEs. The Huber loss value is $[0.115 \pm 0.005]$, not a large value considering the demanding task.

Fig. 4 assets the quality of the prediction of the three Lyapunov exponents (LEs) with respect to the traditional methods [1]. The absolute differences between the LEs estimated by the classical algorithm and the CNN are computed. The proportion of samples within each error interval—$[0, 0.05]$, $(0.05, 0.1]$, $(0.1, 0.5]$, and $(0.5, +\infty)$—is determined, and each interval is assigned a color (green, blue, yellow, and red, respectively) as indicated in the legend at the bottom of the figure, resulting in the final error plots.

For LE_1 (left panel) the largest errors are committed for $LE_1 \leq -0.4$. As already mentioned in the 1D analysis, such LE values correspond to equilibrium points whose LE magnitude is not significant at all for a dynamical study. It is remarkable that for $|LE_1|$ around 0, for more than half of the samples, the error is less than or equal to 0.05. For $LE_1 > -0.4$ in almost all samples, such prediction error is less than or equal to 0.5. Taking into account that the prediction is performed

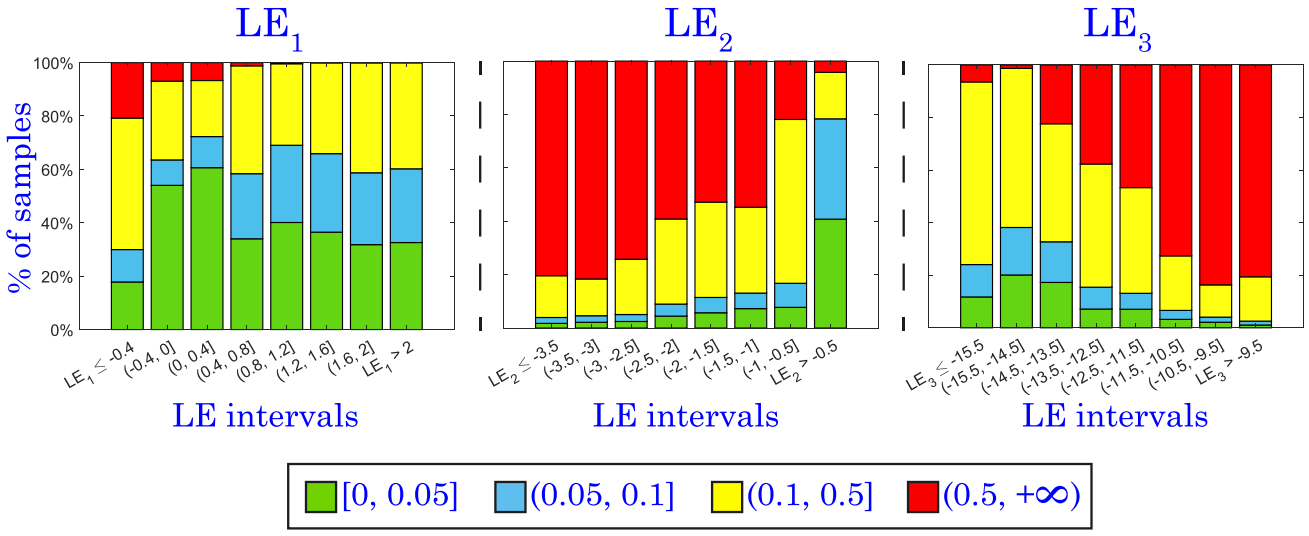

Fig. 4. Error analysis of Lyapunov exponents prediction in the (r, b)-parametric plane (studied in Fig. 3) of the Lorenz system when training with non-random data. From left to right, LE_1, LE_2 and LE_3 results are depicted. The color code is shown at the bottom. (See the text for more details.).

using a short time series of only one system variable, and without any additional dynamical information, the predictions are quite accurate. For LE_2 (middle panel), perhaps the most important LE magnitude values for a dynamical analysis are those around 0, which can provide some insight into, for instance, period-doubling bifurcations. The error analysis indicates that for $|LE_2|$ around 0, for 40% of the samples the prediction is performed with an error no larger than 0.05. Finally, the errors in the LE_3 prediction (right panel) are quite good since for all LE intervals the error is less than or equal to 0.5 in more than 20% of the samples and this is a difficult approximation. We conclude that this DL technique allows us to predict all LEs with only one short single-variable time series, and no other dynamical information of the system, with a good approximation (to the best of the authors' knowledge, using other techniques only the MLE is obtained when just one variable information is used).

3.2. Random data

In Section 3.1, we assumed that we already have the Lyapunov exponents values in a few parametric lines and used such data for training and validation. Now we assume that we do not have any previous Lyapunov exponents data and hence we have to generate it to train the neural network. Therefore, we consider random data along the entire parametric plane that we want to study in detail. For this approach, we randomly choose 24,000 (r, b)-values ($\sigma = 10$) with $b \in [2, 3]$ and $r \in (0, 300]$, and we obtain the x-time series (then, similar samples are removed, and the remaining ones are normalized). Finally, 8000 samples are randomly chosen for training (batch size 128), 2000 for validation (batch size 100), and 2000 for test (batch size 100). Notice that again only 8000 samples are directly related to training. The network architecture is the one presented in Section 2.1.

The value of the Huber loss for training dataset is $[0.054 \pm 0.003]$. The corresponding values for validation and test datasets are $[0.052 \pm 0.003]$ and $[0.057 \pm 0.003]$, respectively. Note that the mean and standard deviation values are sufficiently small for all datasets. Although the value of the test mean is larger than this one for training, we consider that the difference is not significant enough to confirm that the DL technique suffers from overfitting. Therefore, it can be concluded that the training process has been successful. As indicated for the non-random case, the data used for the analyses with the trained CNN are just time series of x-variable of length 1000.

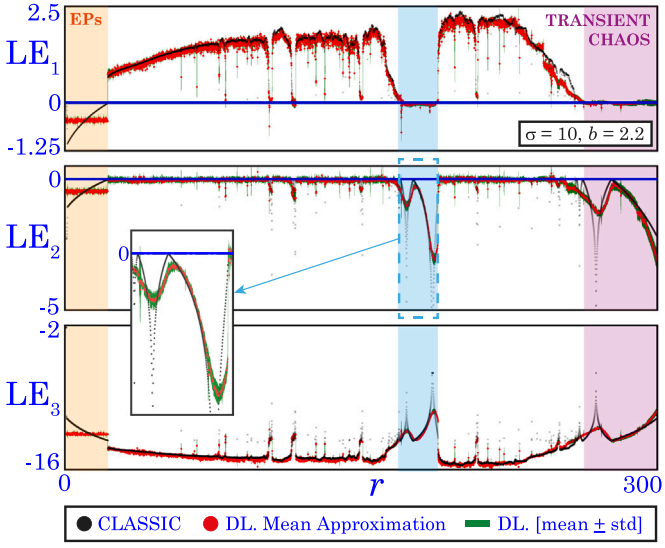

Fig. 5. 1D analysis ($\sigma = 10$, $b = 2.2$) of the Lyapunov exponents in the Lorenz system when training the CNN with random data (Huber loss value $[0.055 \pm 0.003]$). The LEs obtained with classical techniques are in black, and the results given by DL are in red (mean value of the predicted values with the 10 networks) and green (interval [mean ± standard deviation (std)] obtained with the prediction of the 10 networks). The orange and purple shaded regions correspond to the zones where the DL technique seems to give worse results. The blue shading is used in a comparison with the analysis in Fig. 2.

3.2.1. 1D analysis

As a first test, we consider the study of a one-parameter line. In Fig. 5, the CNN trained with random data from the (r, b)-parametric plane has been used for the LEs approximation in one r-parametric line ($\sigma = 10$, $b = 2.2$ and 6000 equidistant r-values in $(0, 300]$) of such plane. As in Fig. 2 (non-random case), in top panel LE_1 is shown, LE_2 is in the middle panel, and finally, in the bottom panel, LE_3 is studied. The value of the Huber loss is $[0.055 \pm 0.003]$. The mean and standard deviation values are small, so we can confirm that the prediction task was successful.

As in the case of non-random data (see Section 3.1, Fig. 2), the orange part (equilibrium points, EPs) and the purple one (transient chaos) are those where the network seems to fail the most. However,

if we compare the results obtained with each data creation technique, it can be seen that training with random data can give more accurate results in the critical region shaded in purple: in LE_1 the variability is smaller, and in LE_2 and LE_3, the shape followed by the DL results is more similar to that of the ground truth. This is an expected fact as a random sweep to create the training data allows to obtain a greater variability in the dynamical behavior for training than restricting to only a few r-parametric lines.

Outside of these problematic regions, the LE predictions of the CNN trained with random data are accurate and with small variability. Let us compare the results in these zones with those in Section 3.1 (Fig. 2). Taking into account the graphical representations, the predictions of LE_1 and LE_3 seem to present no major differences. However, in the prediction of LE_2 it is remarkable that the random case allows the network to predict more accurately and with less variability the LEs in the blue shaded region (a magnification of this zone is shown in the figure). According to the loss function value, the predictions of the random case are better than the ones of the non-random approach as the loss interval [0.052, 0.058] is closer to the origin than [0.086, 0.096].

3.2.2. 2D analysis

Now we show the results of applying the trained CNN for LEs prediction in an (r, b)-parametric plane ($\sigma = 10$, 1000 equidistant values for $r \in [0, 300]$ and 1000 for $b \in [2, 3]$). Fig. 6 is equivalent to Fig. 3, but now the CNN trained with random data has been used. Notice that only 8000 samples were taken for training (and 2000 for validation) from the (r, b)-parametric plane in the figure. As in the non-random case, to compute the time series used by the DL technique, a transient integration is performed for 1000 time units and then the integration is continued for 100 more time units (fixed time step 0.01). The input time series of length 1000 of the CNN are constructed with 1 out of every 10 of the last integration points. Note that, with respect to the classical technique of LEs, less time is needed to obtain the time series used by the CNN to calculate the full LEs spectrum.

In Fig. 6 we compare the results obtained with classical and DL techniques, and it can be seen that, even predicting only with the first variable x of the Lorenz system, DL is able to reproduce quite well the LEs study of this region. At first sight, only the right boundary of the right chaotic region seems to present possible errors as it is blurred. This is the zone where transient chaos occurs and DL is expected to fail. However, if we compare it with the corresponding boundary in the non-random case (see Fig. 3), it can be seen that the quality of the DL prediction is better for all LEs when random dataset is used. In fact, this improvement occurs throughout the parametric plane in general. For example, the upper right corner whose values where incorrectly predicted in the non-random case (see Fig. 3) is now correctly represented. With a deeper visual analysis, the reader may notice that in the LE_2 approximations, the predictions of the random case can help to detect some bifurcation lines (or dynamical changes) that the non-random technique did not allow (or not so clearly). For instance, the thin black line around the r parameter values between 150 and 200 (in the middle of the two large chaotic regions) in the LE_2 classic panel appears in the DL panel of Fig. 6 for large values of b and there is a darker gray line for smaller ones. In Fig. 3 this change cannot be seen so clearly. Another example is the black line around r parameter values between 250 and 300. In the non-random case, there are some black segments that do not give a clear idea of a continuous line. However, for the random prediction, even when black is almost not present, a continuous darker gray line highlights it. Regarding the signed difference between classical and DL approximation (third row of Figs. 3 and 6), we can see that the sign of the difference is, in general, the same for non-random and random cases (same regions with red, white and blue colors), although the magnitude is considerably reduced with random approach in the right zone of the biparametric plane. The value of the Huber loss for this random case is equal to $[0.079 \pm 0.006]$. This interval [0.073, 0.085] is closer to zero than the one of the non-random data case [0.110, 0.120], so the results are more accurate for the random case as already shown in the 1D analysis.

In the second row of Fig. 7 an error analysis equivalent to that in Fig. 4 is given for the random case. For LE_1 (left panel), as in the non-random case, the largest errors occur for $LE_1 \leq -0.4$. It is remarkable that for $|LE_1|$ around 0, for about 60% of the samples, the error is less than or equal to 0.05 (little improvement over the non-random case). For $LE_1 > -0.4$ the error is less than or equal to 0.5 for almost 100% of the samples. For LE_2 (middle panel) and LE_3 (right panel) the advantage of training with random data instead of non-random is notable. For LE_2, when the values are around 0, the percentage of samples with an error less than or equal to 0.05 is 40% in non-random case and more than 60% in the current case. For LE_3, the results are better in the random case. Notice that these predictions are quite good considering that the network did not have much information for training and, to the best of the authors' knowledge, there is no other technique able to approximate LE_3 under these conditions.

To complement this analysis, in the first row of Fig. 7, we have drawn on the (r, b)-parametric plane the error value for each time series (following the color code of the bar plots). Such representation allows to identify the regions with each magnitude of error in the approximation of the LEs. Note that the largest error (red color for errors in the interval $(0.5, +\infty)$) is concentrated mainly for the three LEs in the left part of the plane (that corresponds to equilibrium points), in the boundary regions, and in the right part (where transient chaotic dynamics is present). As explained before, these are zones where the worst approximations are expected to occur. However, the green color (error less than or equal to 0.05) is the predominant one despite the demanding task.

With all the studies performed on the Lorenz system, it can be concluded that a good LE analysis can be performed with DL whether non-random or random data is used for training (the latter providing better results). It is important to highlight that a small number of short time series are used for training (only 8000 for training, and 2000 more for validation, of length 1000), and only the x-variable of the system is used, making this a simple but powerful technique. Moreover, it is also a fast technique. As indicated in the part *Lorenz system* of Table 2, it takes less than 1 h and 40 min to compute a biparametric analysis from scratch with DL on the Lorenz system. Around 36 min (36% of the total time used by the DL process) are needed to obtain the raw data that will be used to create training, validation and test datasets (CPU with parallel computing). Less than 40 min (40% of the total DL time) are spent on data selection (CPU), that is, preparing data and creating the three mentioned datasets. To train one CNN (CUDA with PyTorch) less than 10 min (10% of the total DL time) are used (the results in the paper are obtained from 10 randomly initialized CNNs, but as the standard deviation of the loss function is small, and as in the 1D case the variability seems to be small, it is expected that using a single trained CNN will be sufficient to obtain good results). Finally, around 14 min are used to obtain the full LEs spectrum with the trained CNN in a biparametric plane with dimension 1000×1000: only around 3 s are spent on the network prediction performed in CUDA with PyTorch, and the remaining time is used to obtain the time series used as input to the network (CPU with parallel computing for some computations). Notice that most of the time used by DL (76%) is focused on obtaining suitable data to train the network properly. With classical techniques (CPU with parallel computing), around 25 h are needed to perform such biparametric analysis. Therefore, comparing both techniques (classical and DL), with DL, time is reduced by 93.3% approximately. In fact, once the network has been trained, the time needed to obtain the biparametric analysis is less than 1% of the time used by classical techniques. Furthermore, if the time series are given and only the LEs are computed with the CNN, it only takes 3 s to obtain the full LEs spectrum of the system.

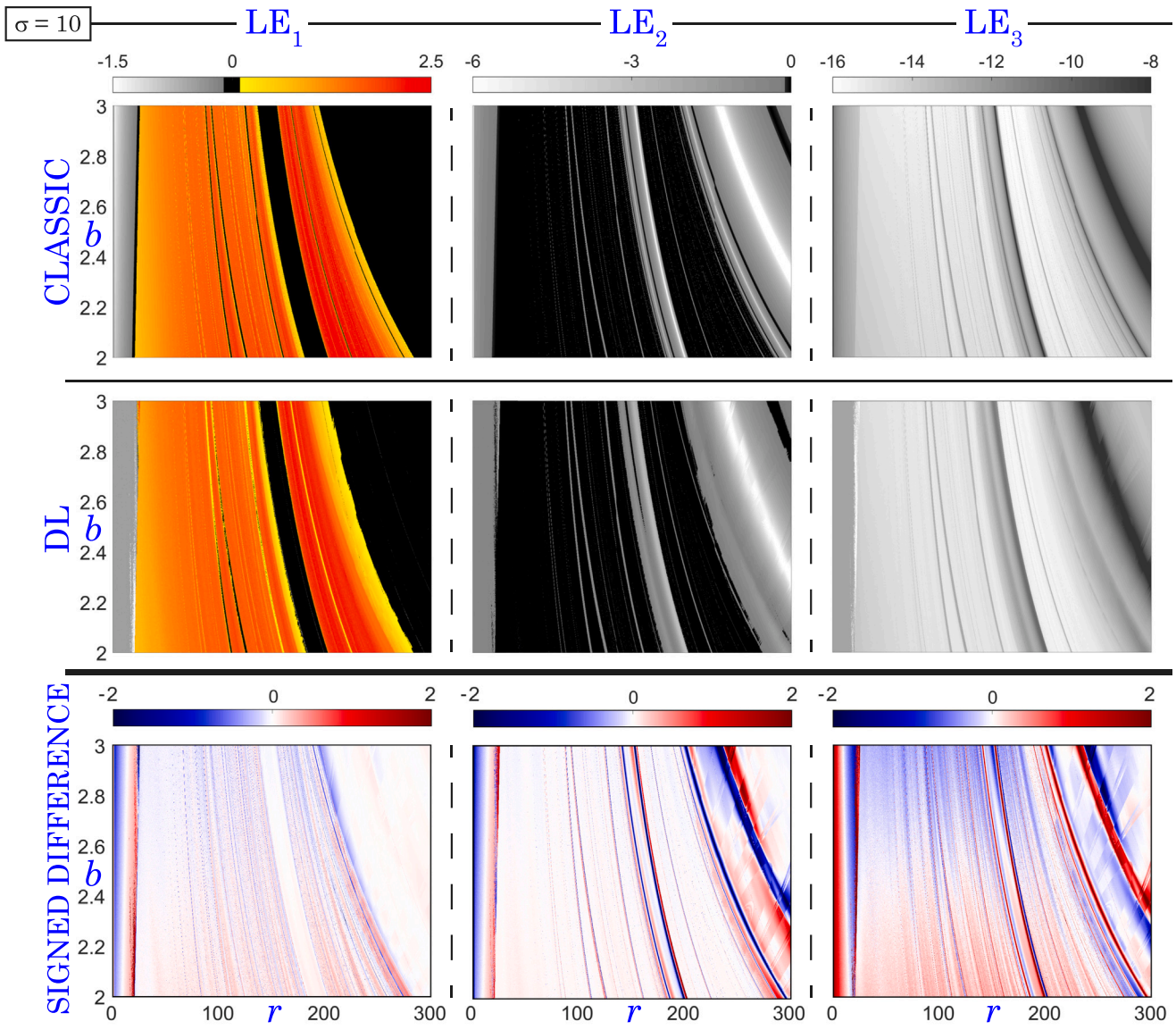

Fig. 6. 2D analysis of Lyapunov exponents in the Lorenz system ($\sigma = 10$) when training with random data (Huber loss value $[0.079 \pm 0.006]$). From left to right, LE_1, LE_2 and LE_3 analyses are depicted. From top to bottom, results with classical techniques, with DL techniques, and signed differences between both approximations (classical values minus DL values). (See the text for more details.).

Table 2
Time analysis for an LE biparametric study with DL and classical techniques. Top: Approximate time needed to perform a biparametric analysis with DL from scratch for the classic Lorenz system and a coupled Lorenz system. For each system, the left column corresponds to the time needed for each DL task (total time is given in the last row), the middle column is for the percentage of time involved in each DL task, and the right column is dedicated to show the percentage of time used by DL with respect to the time needed by the classical technique of LEs for the same analysis. Bottom: Table with the approximate time used by the classical technique of Lyapunov exponents for both systems and the same biparametric analysis. Same meaning for the columns as explained for DL.

DEEP LEARNING	Lorenz system			Coupled Lorenz system		
	Time	% w.r.t. DL	% w.r.t classical	Time	% w.r.t. DL	% w.r.t classical
Creation of raw data	36 min	36%	–	68 min	49.275%	–
Data selection	40 min	40%	–	44 min	31.884%	–
Training one CNN	10 min	10%	–	10 min	7.246%	–
Biparametric analysis. Data	14 min	14%	**0.933%**	16 min	11.594%	**0.494%**
Biparametric analysis. Prediction	3 s	0.05%	**0.003%**	3 s	0.036%	**0.002%**
Total time	1 h 40 min	100%	**6.667%**	2 h 18 min	100%	**4.259%**

CLASSICAL TECHNIQUE LEs	**Lorenz system**			**Coupled Lorenz system**		
	Time	**% w.r.t. DL**	**% w.r.t classical**	**Time**	**% w.r.t. DL**	**% w.r.t classical**
Biparametric analysis. Whole process	25 h	–	100%	54 h	–	100%

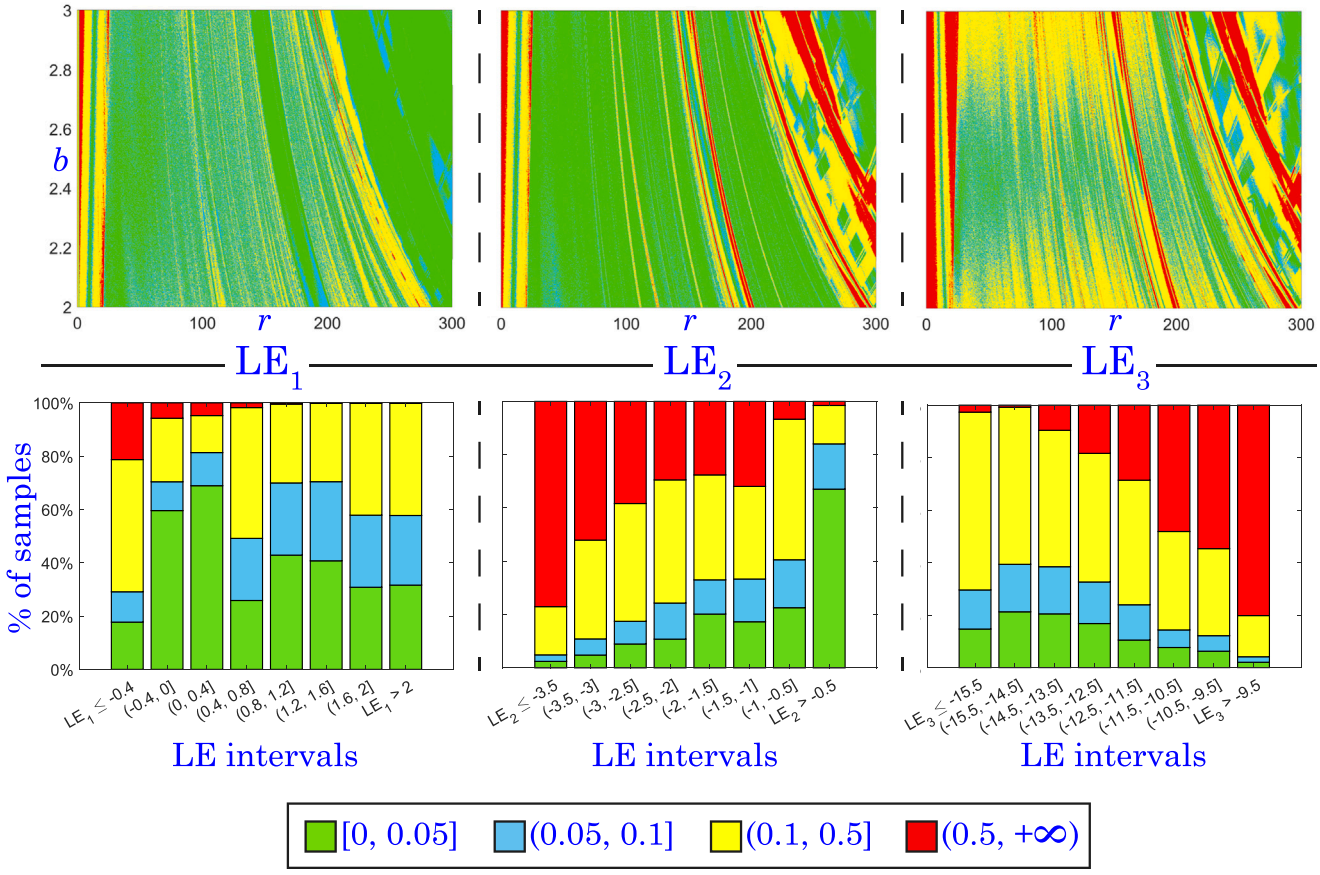

Fig. 7. Error analysis of Lyapunov exponents prediction in the (r, b)-parametric plane (studied in Fig. 6) of the Lorenz system when training with random data. From left to right, LE_1, LE_2 and LE_3 results are depicted. The first row corresponds to the errors represented in the parametric plane. In the second row, the bar plots display the errors associated with different intervals of Lyapunov exponent values. The color code is shown at the bottom. (See the text for more details.).

4. LEs approximation for the coupled Lorenz system

In this section, the two approaches used for the Lorenz system to approximate the full LEs spectrum with DL (and only single-variable time series) are applied to the system obtained by coupling two almost identical Lorenz systems (see Eq. (2)). Remember that in the first approach (non-random case) a CNN is trained using time series of a few r-parametric lines, and in the second one (random case), the same architecture is trained from scratch using information with random parameter values from an (r, b)-parametric plane. As already mentioned, these two approaches will show that DL techniques can be used to extend a partial classical study of the system (first approach) or to do the analysis from random data (second approach). The performance of both approaches in approximating the LEs in the coupled Lorenz system will be compared. Moreover, we will show that DL techniques allow to locate hyperchaotic behavior and to find regions with invariant tori using only one variable of the system (in this case, x_1).

The architecture of the CNN used for this task (in both approaches) is the one presented in Section 2.1. The training, validation and test datasets from the coupled Lorenz system for both approaches are obtained in an equivalent way to how such sets are constructed for the isolated Lorenz model (see Section 3). The initial conditions are $(x_1, y_1, z_1, x_2, y_2, z_2) = (1, 1, 1, 1, 1, 1)$, the coupling parameters λ_1 and λ_2 are set to 0.1, and the time series used by DL will be x_1-time series.

In the non-random approach, the value of the Huber loss for the training dataset is $[0.016 \pm 0.002]$. For the validation and test datasets, the value of the loss function is $[0.048 \pm 0.001]$ and $[0.056 \pm 0.004]$, respectively. Although the mean value for the test set is a little larger than the corresponding value for the training set, we consider that the difference is not sufficiently large to discard the network due to overfitting. In the random approach, the value of the Huber loss for training, validation and test datasets is $[0.024 \pm 0.003]$, $[0.024 \pm 0.004]$ and $[0.023 \pm 0.003]$, respectively. Notice that the mean and standard deviation values are small enough for all three datasets.

As the study we perform on the coupled Lorenz system is quite similar to that performed on the Lorenz system in Section 3, for simplicity, we present together the results obtained with both approaches. In Section 4.1 and Section 4.2, 1D and 2D analyses are performed with both approaches to show that the approximation of the full LEs spectrum of a high-dimensional system using DL and short (length 1000) single-variable time series is possible. Moreover, in Section 4.3 we use the approximation of the LEs to perform a dynamics classification of a biparametric plane.

4.1. 1D analysis

In Fig. 8, the trained CNN has been used to obtain the LEs approximation of an r-parametric line with $b = 2.2$, $\sigma = 10$ and $\lambda_1 = \lambda_2 = 0.1$ (6000 equidistant r-values in the range $[0, 300]$ are considered). Note that this line is parallel to the r-parametric lines used in the non-random approach to create the training and validation datasets (see top-left panel of Fig. 9 where light green r-parametric lines are the ones used for the training dataset, and dark green one is for the validation set). The panels in the left column of Fig. 8 correspond to the approximation of the full LEs spectrum obtained with the non-random approach, and those on the right show the results of the random approach. In all panels, the ground truth of the LEs (computed with the algorithm in [1]) is in black, the mean of the predicted values with the 10 networks is in red, and the variability (i.e., the interval [mean ± standard deviation (std)] obtained with the prediction of the

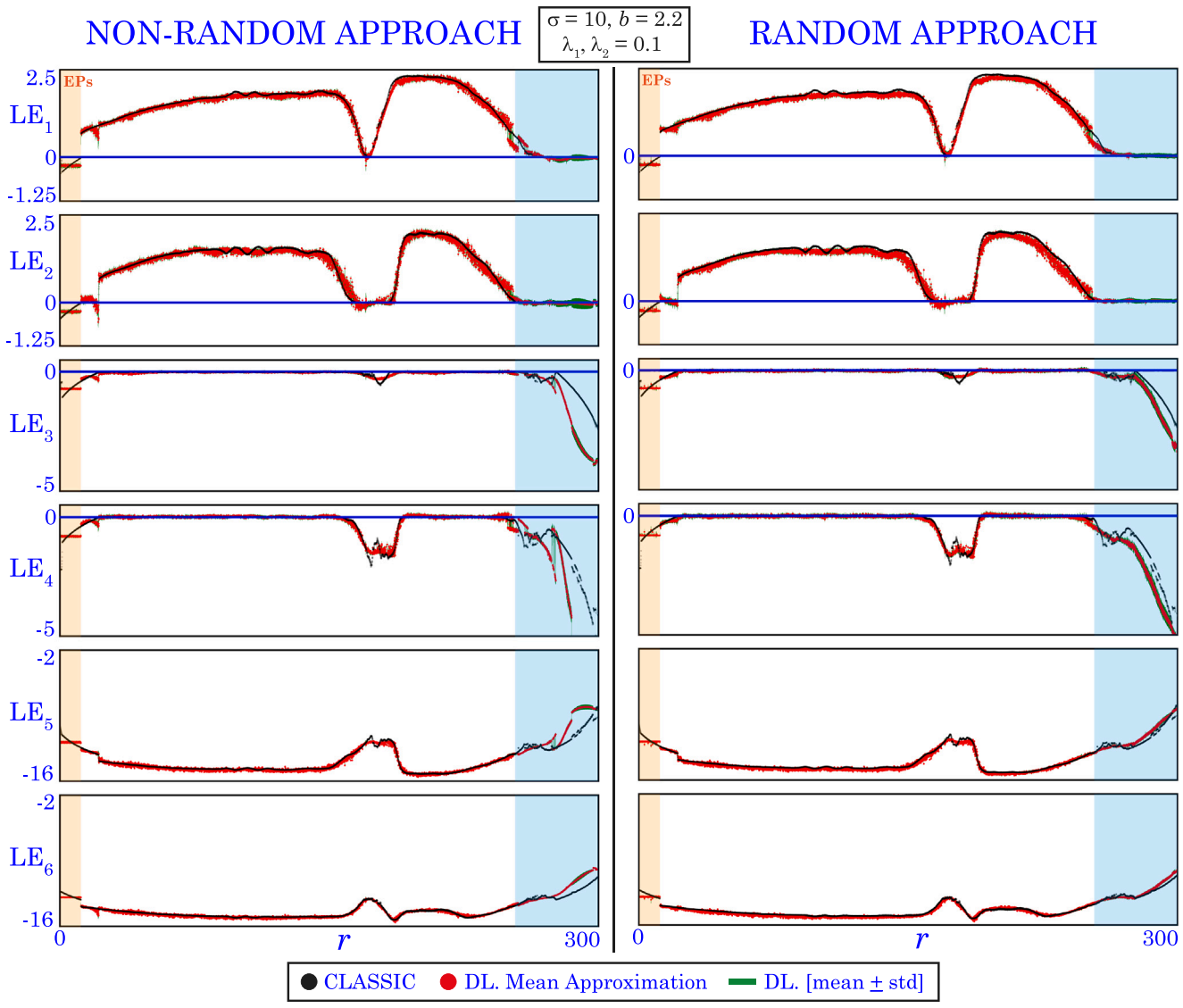

Fig. 8. 1D analysis ($\sigma = 10$, $b = 2.2$, $\lambda_1 = \lambda_2 = 0.1$) of the Lyapunov exponents in the coupled Lorenz system when training the CNN with non-random data (left column) and random data (right column). The Huber loss value in the non-random approach is $[0.274 \pm 0.023]$, and in the random approach it is $[0.273 \pm 0.027]$. The LEs obtained with classical techniques are in black, and the approximations given by DL are in red (mean value of the predicted values with the 10 networks) and green (interval [mean ± standard deviation (std)] computed with the prediction of the 10 networks). Orange shading corresponds to equilibrium points (EPs) where the network is expected to fail. Blue shading is used to compare non-random and random approaches.

10 networks) is in green. The value of the Huber loss is $[0.274 \pm 0.023]$ in the non-random case, and $[0.273 \pm 0.027]$ in the random approach.

At first glance, in Fig. 8 we can observe how both DL approaches provide very good approximate values of the six Lyapunov exponents using just short single-variable time series! The interval with convergence to an equilibrium point (orange shaded region) is the zone where the DL technique seems to provide the worst results for both approaches. As indicated for the Lorenz system, inexact results can be expected in this region due to the normalization rule. However, as the LEs of this type of dynamics do not provide useful information beyond the negative sign in the first exponent, we can consider that the DL prediction has been successful in this part as well.

Comparing both approaches, we observe that training with random data gives more accurate results in the right part of the line (blue shading): the DL results for the last four Lyapunov exponents are closer to the ground truth values and the shape is more similar. As in the Lorenz system, using a random sweep to create the training data allows to obtain more variability in the dynamical behavior for training (and therefore better approximations of the LEs spectrum) than restricting this data to only a few r-parametric lines.

In any case, the CNN that was trained *only* with a small number of x_1-time series is able to predict the full LEs spectrum, giving correct values with only a small variability, using both non-random and random training data. This is a remarkable result, since only short single-variable time series are sufficient to approximate all six Lyapunov exponents using a properly trained CNN, and, to the best of the authors' knowledge, this is not possible with other techniques.

4.2. 2D analysis

In this subsection we use the trained CNN to perform a biparametric analysis of the coupled Lorenz system. In Fig. 9 (non-random approach) and Fig. 10 (random approach) we have the LEs study (with classical and DL techniques) and the signed difference between the approximations of both methods for the (r, b)-parametric plane with $\sigma = 10$, $r \in [0, 300]$, $b \in [2, 3]$ and $\lambda_1 = \lambda_2 = 0.1$ (1000 point values for each varying parameter, which makes a total of 10^6 points). These figures are equivalent to Fig. 3 (non-random approach) and Fig. 6 (random approach) of the Lorenz system (same explanation for the gradation

of the colorbars). As in the isolated system, to obtain the time series used as input by the DL technique, a transient integration is performed for 1000 time units and then the integration is continued for 100 more time units (time step 0.01 for the whole integration). The CNN input time series of length 1000 are built with 1 out of every 10 of the last integration points. Therefore, with respect to classical techniques, less integration time is needed to approximate the full LEs spectrum.

From Figs. 9 and 10 we can observe how the DL technique seems to work correctly for both approaches. The non-random approach shows worse approximations in the transition areas from regular to chaotic behavior and vice versa, but in any case the overall result is quite good taking into account the demanding task. The Huber loss value for this biparametric plane in the non-random case is $[0.046 \pm 0.002]$, and for the random approach it is $[0.023 \pm 0.004]$. In both cases the mean value and standard deviation are small, indicating that the process has been successful. It is remarkable that even using only a small number of short single-variable time series (and without additional dynamical information), the approximation for all LEs (in both approaches) is similar to that provided by the classical technique (which uses more integration time and all system variables). This good approximation will allow to perform dynamics classification to determine, for example, hyperchaotic regions (see Section 4.3).

In Fig. 11 (non-random approach) and in the second and fourth rows of Fig. 12 (random approach), we have the error analysis of each approach. Such analyses are equivalent to the ones in Fig. 4 (non-random approach) and the second row of Fig. 7 (random approach) for the isolated Lorenz system. As in the Lorenz system case, these analyses are given separately for each Lyapunov exponent. For each exponent, the samples are divided into different groups according to their LE value given by the classical technique (see the LE intervals on the horizontal axis of the plots). For each LE interval, the percentage of samples belonging to each of the error intervals ($[0, 0.05]$, $(0.05, 0.1]$, $(0.1, 0.5]$ and $(0.5, +\infty)$) is calculated and a color is assigned to each error interval (green, blue, yellow and red, respectively). As expected, the advantage of training with random instead of non-random data is remarkable for the performance of the Lyapunov exponents approximation: the percentage of samples whose error is larger that 0.5 (red color) is considerably lower for the random case. For both approaches, the approximation is better for LE_1 and LE_2 (error is less than or equal to 0.5 in almost 100% of the samples), but the results for the remaining LEs are still surprising considering that the six LEs are obtained only from single-variable time series.

To enhance the error analysis, in the first and third rows of Fig. 12 we have drawn in the (r, b)-parametric plane, for the random case, the error for each time series (same color code as in the bar plots). With this representation we can identify the regions with each magnitude of error in the LEs approximation. Notice that the red color (largest possible error interval) is not visible in the graphical representation of the first two LEs, and for the remaining LEs it is present mainly in the right part of the biparametric plane. In general, for all LEs, the green color (smallest error) is the predominant one.

From the studies performed on the coupled Lorenz system, it can be concluded that a good LE analysis is obtained with DL whether non-random or random data is used for training (the second approach provides better results). It is remarkable that a small number of short (length 1000) time series have been used for training (8000 samples for training, and 2000 more for validation), and that only one of the six variables of the system is used (without other dynamical information). Therefore, this is a simple but powerful technique. In addition, it is also a fast technique. As indicated in the *Coupled Lorenz system* part of Table 2, to obtain a biparametric analysis (of this coupled Lorenz system) from scratch with DL, approximately 2 h and 18 min are needed. About 68 min (49.275% of total DL time) are spent on obtaining the raw data used later to create training, validation and test datasets (CPU with parallel computing). It takes less than 44 min (31.884% of total time used by DL process) to perform data selection, i.e., to prepare the data and to create the three mentioned datasets (CPU). To train one CNN (CUDA with PyTorch), about 10 min (7.246% of total DL time) are used (as already indicated for the isolated Lorenz system, the results in the paper are obtained from 10 randomly initialized CNNs, but as the results obtained for this coupled system are good and with small variability, it is expected that using a single CNN will be sufficient to obtain good results). Finally, it takes around 16 min to obtain the full LEs spectrum on a biparametric plane with dimension 1000×1000 using the trained CNN: only 3 s (0.036% of the total DL time) are needed for the network prediction on CUDA with PyTorch, the remaining time (11.594% of total DL time) is used to compute the time series used as network input (CPU with parallel computing for some calculations). With classical techniques (CPU with parallel computing), it takes almost 54 h to obtain such biparametric analysis. So, comparing both techniques (classical and DL), with DL the time is reduced by approximately 96%. In fact, once the network is trained, the time needed to obtain the biparametric analysis is less than 0.5% of the time used by classical techniques. Moreover, if the time series are given and only the LEs need to be computed with the trained CNN, it only takes 3 s to obtain the full LEs spectrum of the system.

4.3. Dynamics classification

Finally, in Fig. 13 we use the LEs approximation obtained with DL (random approach) to study different dynamical regimes that can be found in the parameter space of the coupled Lorenz system (comparing the results with those given by classical techniques).

In panels (A1) and (A2), we can see the study of Lyapunov exponents with classical and Deep Learning techniques (random approach), respectively, in the (r, b)-parametric region analyzed in Section 4.2. Each color range corresponds to a different dynamical regime (see colorbars above). To obtain these analyses, we have used the approximate value of the LEs (given by each technique) to classify the dynamics and we have chosen the most appropriate LE to represent in each case. In particular, for classification with both techniques (classical and DL), the threshold 0.1 is used in such a way that if $LE \in [-0.1, 0.1]$, then the LE value is considered to be zero (notice that the definition of LEs involves a limit, so it is necessary to set a threshold). The region with tori is marked with blue gradation and the value of the third LE is represented (notice that the first and second LEs are 0 in this case). The red and green gradations coincide with chaotic and hyperchaotic dynamics, respectively (the value of the first LE is drawn in both cases). The yellow part corresponds to the case where the dynamics evolves to an equilibrium point (EP). Note that less information is used in the DL approximation (only single-variable time series that are shorter than the ones used by classical techniques), but all regions are well defined and correspond to those indicated by the LE approximation given by classical techniques.

In panels (B1) and (B2), we have represented the first (black), second (red), and third (purple) LE of the r-parametric line studied in Section 4.1 given by the classical and DL (random approach) techniques, respectively. Note that this line is the dashed dark purple line in panels (A1) and (A2), so we are going to analyze the LEs value to verify the behavior represented in these biparametric panels. From left to right, a region of equilibrium points is found (first LE is negative and corresponds to the yellow region in panels (A1) and (A2)). Then, a small chaotic zone is identified since the first LE is positive and the second one is zero (red region in panels (A1) and (A2)). Later, a large hyperchaotic region is represented (first two LEs are positive) that coincides with the green region in panels (A1) and (A2). Next, a small chaos-torus-chaos zone is shown. Note that the torus region corresponds to the blue part of panels (A1) and (A2), and with the first two LEs equal to zero. Finally, there is a large hyperchaotic region followed by a small chaotic region and a large part with tori. Notice that all these different zones are clearly identifiable in panel (A2) obtained with DL.

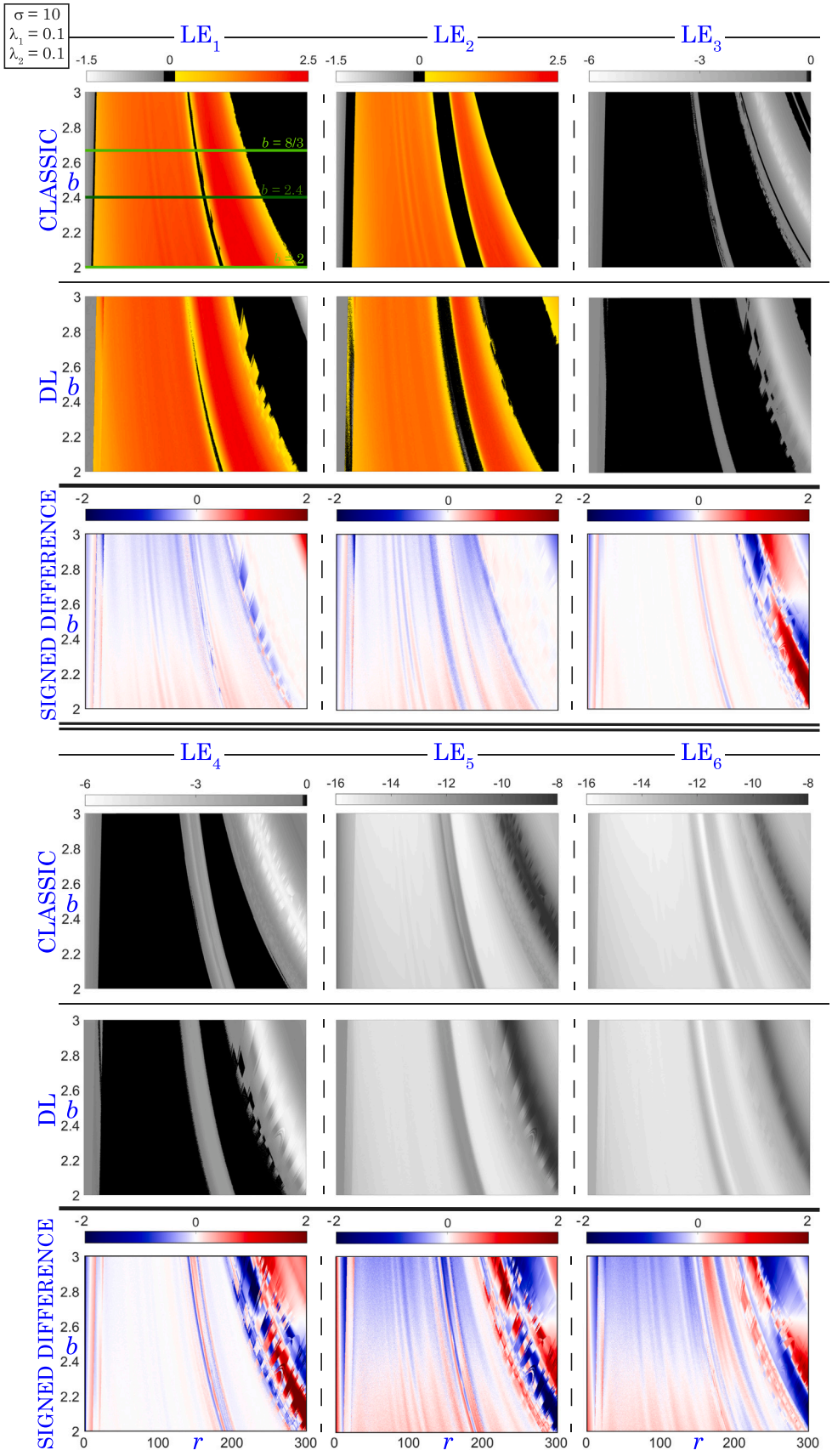

Fig. 9. 2D analysis of Lyapunov exponents in the coupled Lorenz system ($\sigma = 10$, $\lambda_1 = \lambda_2 = 0.1$) when training with non-random data (Huber loss value is $[0.046 \pm 0.002]$). The analyses of the six LEs of the system are depicted for classical and DL techniques. The signed difference between classical and DL approximations is also studied. The lines in the top-left panel contain the data used to create the training data (light green) and the validation dataset (dark green). (See the text for more details.).

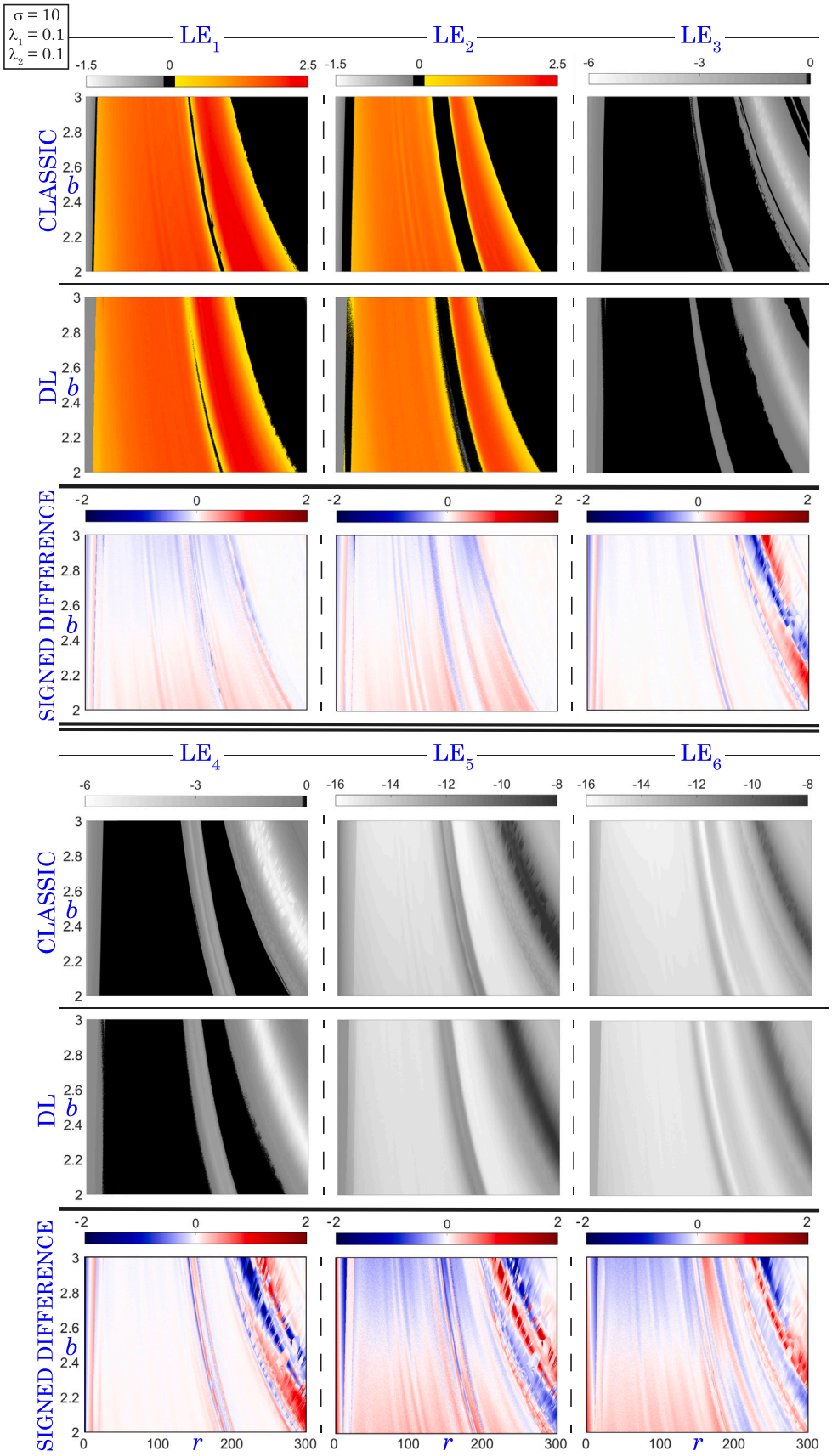

Fig. 10. 2D analysis of Lyapunov exponents in the coupled Lorenz system ($\sigma = 10$, $\lambda_1 = \lambda_2 = 0.1$) when training with random data (Huber loss value is $[0.023 \pm 0.004]$). The analyses of the six LEs of the system are depicted for classical and DL techniques. The signed difference between the approximations given by each method is also studied. (See the text for more details.).

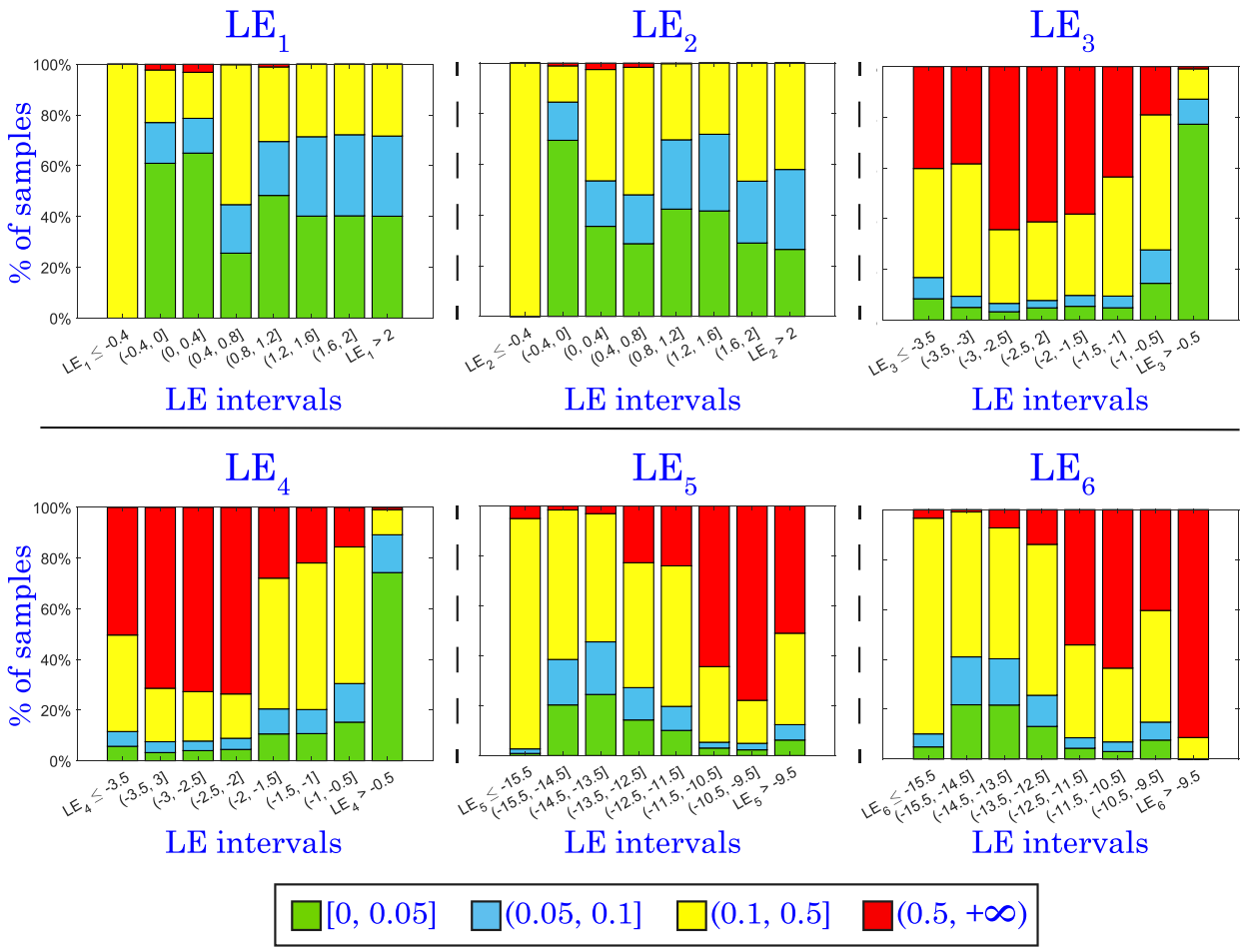

Fig. 11. Error analysis of Lyapunov exponents prediction in the (r, b)-parametric plane (studied in Fig. 9) of the coupled Lorenz system when training with non-random data. The color code is shown at the bottom. (See the text for more details.).

Finally, in panels (C1), (C2) and (C3), the main dynamics of this biparametric region are represented (colored dots have been included on the horizontal axis of panels (B1) and (B2) to locate the orbits on the one-parameter line). In panel (C1), a 3D representation of a torus ($\sigma = 10$, $b = 2.2$, $r = 280$, $\lambda_1 = \lambda_2 = 0.1$) and its 2D projection are drawn. In panel (C2), a 3D representation of a chaotic orbit ($\sigma = 10$, $b = 2.2$, $r = 17$, $\lambda_1 = \lambda_2 = 0.1$) of the coupled Lorenz system is shown. In panel (C3), a 3D representation of a hyperchaotic orbit ($\sigma = 10$, $b = 2.2$, $r = 210$, $\lambda_1 = \lambda_2 = 0.1$) and its 2D projection are represented.

5. Conclusions

The Lyapunov exponents spectrum of a dynamical system is possibly one of its most fundamental properties as it permits to characterize the dynamics of the system. Its computation can be highly computationally expensive, especially if one focuses on a classification problem in a parametric plane. But this information can provide a global overview of the dynamics, so it is quite important.

In this paper, a well-known Deep Learning network (Convolutional Neural Network, CNN) has been built and trained to carry out the approximation of the full Lyapunov exponents spectrum of a dynamical system. The training process is performed using as data only single-variable time series and the full Lyapunov exponents spectrum of a small number of points in the parameter space we want to study. Once trained, the network only needs short time series of a single variable of the system to approximate the LEs spectrum, which means a great reduction in time and memory. The methodology has been applied to two test problems: the Lorenz system and the coupled Lorenz system.

For the Lorenz system, we have used the trained CNN to study the behavior of an r-parametric line and an (r, b)-parametric plane of the parameter space. For the coupled Lorenz system, we study the behavior on the corresponding r-parametric line and (r, b)-parametric plane as in the isolated Lorenz model, but now as the system has dimension six, we use the network to approximate the six Lyapunov exponents. We highlight that the training process uses just a few lines of one-parameter data or a short number of random points to create a network capable of performing biparametric studies. This is a remarkable result that shows the power of DL techniques in dynamical systems studies.

For the biparametric study of the Lorenz system, it takes about 25 h to perform such analysis with classical techniques, while with the CNN less than 2 h are needed for the same task. A 93.3% of the time is saved with DL techniques (if time series are given, the prediction time is just 3 s). In the case of the biparametric study of the coupled Lorenz system, it takes almost 54 h to obtain this analysis with classical techniques, while with the CNN just over two hours are necessary. Therefore, a 96% of the time is saved with DL (if time series are given, the prediction time is just 3 s).

We conclude that Deep Learning can be used not only to analyze the behavior (regular, chaotic or hyperchaotic) of a dynamical system, but also to quantify the values of the Lyapunov exponents spectrum, that is, to go beyond a classification problem. Our results show that even dense 2D parametric studies can be carried out in a very reasonable time using data from only a small portion of the global phase space. However, a deeper study would be necessary to know how far we can go using these techniques in this and other dynamical systems tasks. In summary, we have shown that inference of the full Lyapunov exponents

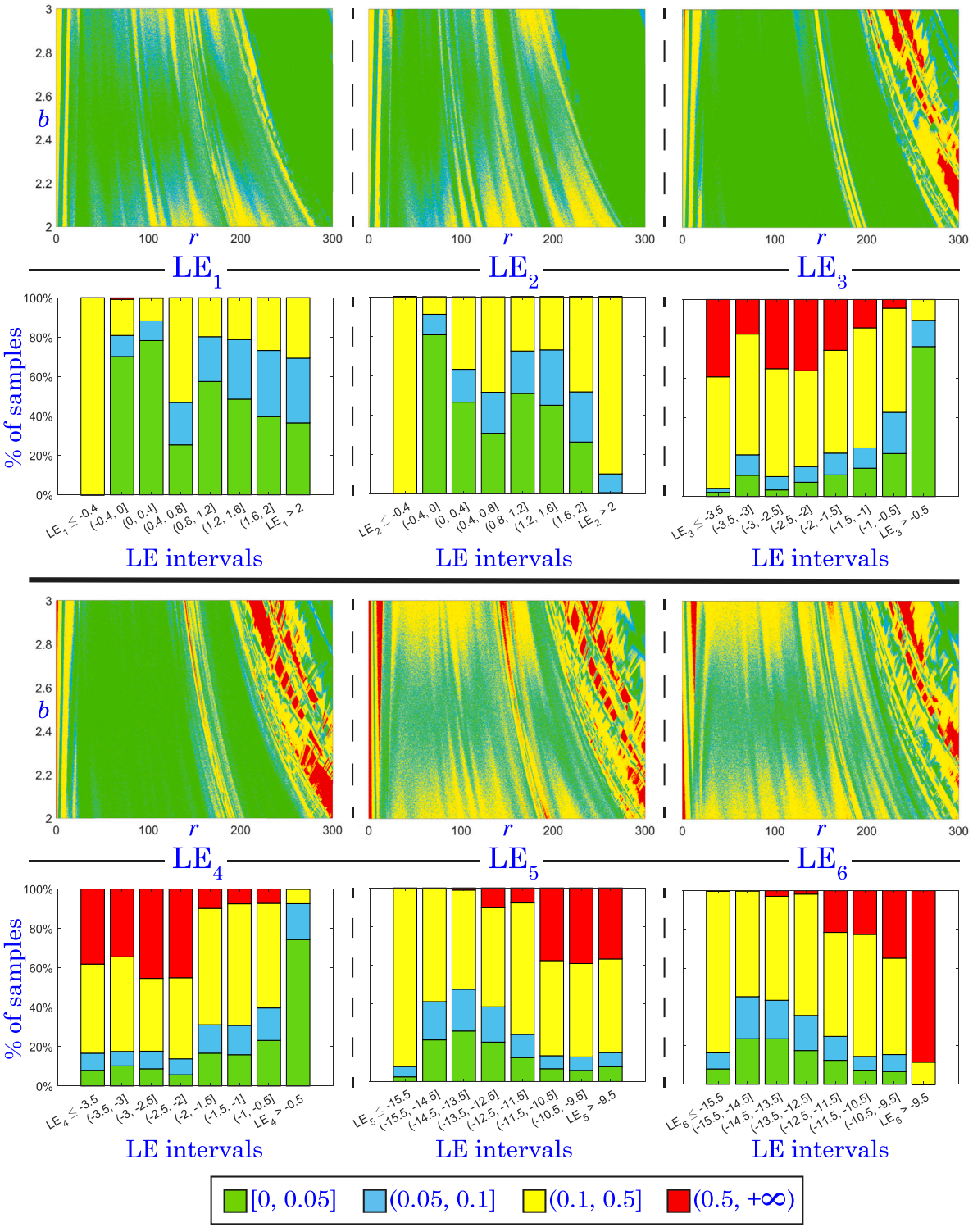

Fig. 12. Error analysis of Lyapunov exponents prediction in the (r, b)-parametric plane (studied in Fig. 10) of the coupled Lorenz system when training with random data. The first and third rows correspond to the errors represented on the parametric plane. In the second and fourth rows, the bar plots show the error for the six LEs. The color code is in the box below. (See the text for more details.).

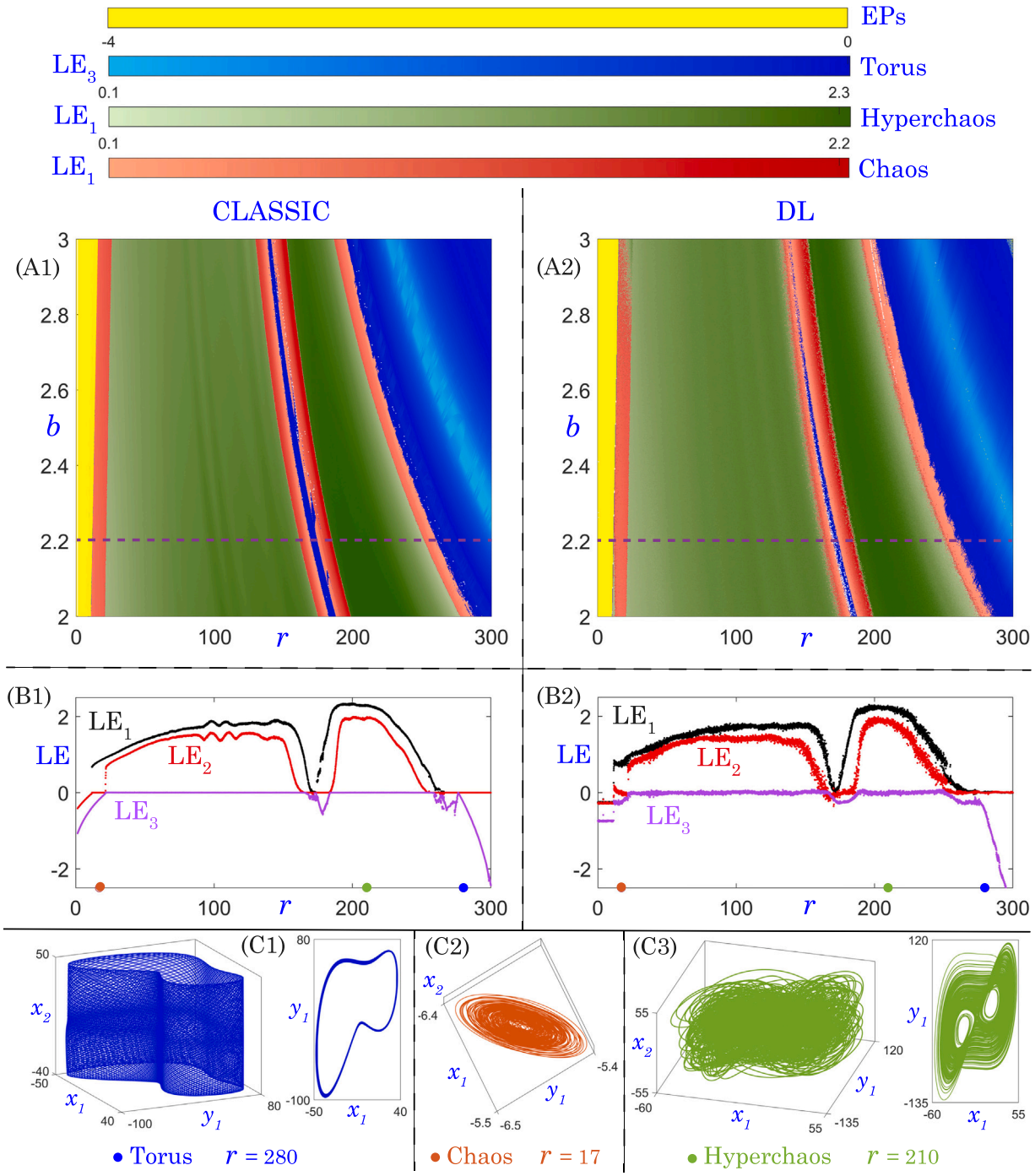

Fig. 13. Study of different dynamics in the coupled Lorenz system. (A1)-(A2) (r, b)-parametric study ($\sigma = 10$, $\lambda_1 = \lambda_2 = 0.1$) of LEs with the classical and DL techniques (random approach), respectively. Each color corresponds to different dynamics as indicated in the colorbars above: yellow for equilibrium points (EPs), blue for tori, green for hyperchaos, and red for chaos. (B1)-(B2) r-parametric study ($\sigma = 10$, $b = 2.2$, $\lambda_1 = \lambda_2 = 0.1$) of the first, second and third LE with classical and DL techniques, respectively. This one-parameter line is marked with the dashed dark purple line in panels (A1) and (A2). (C1)-(C2)-(C3) Representations of a torus ($r = 280$), a chaotic orbit ($r = 17$) and a hyperchaotic orbit ($r = 210$), respectively (points on the horizontal axis in panels (B1) and (B2) locate the orbits in the parameter space). In all panels, the initial conditions are set to 1 for all the variables.

spectrum from a short single-variable time series is possible and robust with DL.

CRediT authorship contribution statement

Carmen Mayora-Cebollero: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Ana Mayora-Cebollero:** Writing – review & editing, Supervision, Investigation, Conceptualization. **Álvaro Lozano:** Writing – review & editing, Supervision, Investigation. **Roberto Barrio:** Writing – review & editing, Supervision, Project administration, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

RB, AL, AMC and CMC have been supported by the Spanish Research project, Spain PID2021-122961NB-I00. RB, AMC and CMC have been supported by the European Regional Development Fund and Diputación General de Aragón, Spain (E24-23R). RB has been supported by the European Regional Development Fund and Diputación General de Aragón, Spain (LMP94-21). AL has been supported by the European Regional Development Fund and Diputación General de Aragón, Spain (E22-23R).

Data availability

Data will be made available on request.

References

[1] A. Wolf, J.B. Swift, H.L. Swinney, J.A. Vastano, Determining Lyapunov exponents from a time series, Phys. D 16 (3) (1985) 285–317.
[2] M.T. Rosenstein, J.J. Collins, C.J. De Luca, A practical method for calculating largest Lyapunov exponents from small data sets, Physica D 65 (1) (1993) 117–134.
[3] A.V. Makarenko, Deep learning algorithms for estimating Lyapunov exponents from observed time series in discrete dynamic systems, in: 2018 14th International Conference Stability and Oscillations of Nonlinear Control Systems(Pyatnitskiy's Conference), STAB, IEEE, 2018, pp. 1–4.
[4] V. Golovko, Estimation of the Lyapunov spectrum from one-dimensional observations using neural networks, in: Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings, IEEE, 2003, pp. 95–98.
[5] Y. Savitsky, A. Savitsky, Technique of learning rate initialization for efficient training of MLP: Using for computing of Lyapunov exponents, in: 2015 International Conference on Information and Digital Technologies, IEEE, 2015, pp. 298–301.
[6] L.A. Dmitrieva, Y.A. Kuperin, N.M. Smetanin, G.A. Chernykh, Method of calculating Lyapunov exponents for time series using artificial neural networks committees, in: 2016 Days on Diffraction, DD, IEEE, 2016, pp. 127–132.
[7] S. Bompas, B. Georgeot, D. Guéry-Odelin, Accuracy of neural networks for the simulation of chaotic dynamics: Precision of training data vs precision of the algorithm, Chaos 30 (11) (2020).
[8] 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).
[9] C.F. Higham, D.J. Higham, Deep learning: An introduction for applied mathematicians, SIAM Rev. 61 (4) (2019) 860–891.
[10] A. Géron, Hands-ON Machine Learning with Scikit-Learn & TensorFlow, O'Reilly Media, Inc, Sebastopol, 2019.
[11] R. Barrio, S. Serrano, A three-parametric study of the Lorenz model, Phys. D 229 (1) (2007) 43–51.
[12] R. Barrio, S. Serrano, Bounds for the chaotic region in the Lorenz model, Phys. D 238 (16) (2009) 1615–1624.
[13] M.R. Gallas, M.R. Gallas, J.A. Gallas, Distribution of chaos and periodic spikes in a three-cell population model of cancer, Eur. Phys. J. Special Topics 223 (11) (2014) 2131–2144.
[14] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, R. Vigara, Deep Learning for chaos detection, Chaos 33 (7) (2023) 073146.
[15] N. Boullé, V. Dallas, Y. Nakatsukasa, D. Samaddar, Classification of chaotic time series with deep learning, Phys. D 403 (2020) 132261.
[16] W.S. Lee, S. Flach, Deep learning of chaos classification, Mach. Learn.: Sci. Technol. 1 (4) (2020) 045019.
[17] A. Celletti, C. Gales, V. Rodriguez-Fernandez, M. Vasile, Classification of regular and chaotic motions in Hamiltonian systems with deep learning, Sci. Rep. 12 (1) (2022) 1–12.
[18] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141.
[19] G. Grassi, F.L. Severance, D.A. Miller, Multi-wing hyperchaotic attractors from coupled Lorenz systems, Chaos Solitons Fractals 41 (1) (2009) 284–291.
[20] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE, vol. 86, 1998, pp. 2278–2324.
[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, L. Zeming, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance Deep Learning library, in: Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, pp. 8024–8035.
[22] V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning, 2016, arXiv preprint arXiv:1603.07285.
[23] M.M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021, arXiv preprint arXiv:2104.13478.
[24] P.J. Huber, Robust estimation of a location parameter, Ann. Math. Stat. (1964) 73–101.
[25] T. Hastie, R. Tibshirani, J.H. Friedman, J.H. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, vol. 2, Springer, 2009.
[26] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.
[27] T. Tél, Y.-C. Lai, Chaotic transients in spatially extended systems, Phys. Rep. 460 (6) (2008) 245–275.