TESIS DE LA UNIVERSIDAD
DE ZARAGOZA

2025

195

Carmen Mayora Cebollero

Machine Learning Techniques in Dynamical Systems: Applications in Excitable Systems

Director/es
Barrio Gil, Roberto
Lozano Rojo, Álvaro

http://zaguan.unizar.es/collection/Tesis

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

cc BY NC

Universidad de Zaragoza
Servicio de Publicaciones

ISSN 2254-7606

Tesis Doctoral

MACHINE LEARNING TECHNIQUES IN DYNAMICAL SYSTEMS: APPLICATIONS IN EXCITABLE SYSTEMS

Autor

Carmen Mayora Cebollero

Director/es

Barrio Gil, Roberto
Lozano Rojo, Álvaro

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

2025

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

Tesis Doctoral

Machine Learning Techniques in Dynamical Systems: Applications in Excitable Systems

Autor

Carmen Mayora Cebollero

Director/es

Roberto Barrio Gil
Álvaro Lozano Rojo

Facultad de Ciencias / Escuela de Doctorado Universidad de Zaragoza

2025

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

Acknowledgments

First of all, I would like to thank my supervisors Roberto Barrio and Álvaro Lozano for their help, valuable advice and support. In addition to learning a lot from you, we have had a great time. I would also want to thank the remaining members of the Computational Dynamics (CoDy) group for the great moments: Jorge A. Jover Galtier, María Ángeles Martínez, Ana Mayora Cebollero, Marcos Rodríguez, Sergio Serrano, and Rubén Vigara. I am sure that there is no better research group. I hope we continue researching and sharing laughs together for a long time.

I would like to thank Flavio Fenton from Georgia Tech (Atlanta, USA) for hosting me during three months. Thanks for giving me the opportunity to learn from you and for allowing me to participate in the experiments. I enjoyed my time in Atlanta with you and all the people in your lab. I would also like to thank Elizabeth Cherry and her group, I really liked sharing time and knowledge with you in the Cardiac Journal Club. I hope we will work together for a long time.

Thanks also to our collaborators Antonio Miguel and Alfonso Ortega from the ViVo Lab group of the University of Zaragoza. It was easier to learn Machine Learning with all your help. I hope we will continue sharing ideas and working together. You are the best people to stay updated on the latest developments in Deep Learning.

I would like to thank all the professors and researchers in the Department of Applied Mathematics, in the IUMA, and in the APEDIF group. It has been great to do my doctoral thesis in this department. I would also want to thank my colleagues in the Faculty of Veterinary and the School of Engineering and Architecture (EINA). It has been wonderful to share my first teaching hours with you. Thank you also to all the PhD students and early career researchers I have met during my PhD studies. I really enjoyed the time with you.

Thanks to my parents Salvador and Ana. You have helped me more than you think. Thank you for drying my tears and brightening my smiles. Thank you for supporting me every day and always being there for me. Thank you for your love and all the moments together. Thanks to you, I am who I am. I love you so much. I would also want to thank my grandparents Jesús, Rosario, Salvador and Carmen. Thank you for believing in me.

I saved the best for the last. Thank you to my twin sister Ana. You are everything to me, my sister, my best friend, my colleague, my other half. It is great to have someone to share everything with. It has been wonderful to work on our doctoral theses side by side. I love you more than you can imagine.

This doctoral thesis is submitted as a compendium of articles that have been published after the start of the doctoral studies of Carmen Mayora Cebollero. This doctoral thesis consists of four articles, three of them are published in journals whose impact factor is included in the Journal of Citation Reports (JCR).

[1] R. Barrio, S. Ibáñez, J.A. Jover-Galtier, Á. Lozano, M.Á. Martínez, A. Mayora-Cebollero, C. Mayora-Cebollero, L. Pérez, S. Serrano, and R. Vigara, "Dynamics of excitable cells: Spike-adding phenomena in action", *SeMA Journal*, vol. 81, no. 1, pp. 113-146, 2024.
`doi:10.1007/s40324-023-00328-2`

[2] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 7, p. 073146, 2023.
`doi:10.1063/5.0143876`

[3] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, "Full Lyapunov exponents spectrum with Deep Learning from single-variable time series", *Physica D: Nonlinear Phenomena*, vol. 472, p. 134510, 2025.
`doi:10.1016/j.physd.2024.134510`

[4] Á. Lozano, R. Vigara, C. Mayora-Cebollero, and R. Barrio, "Dominant patterns in small directed bipartite networks: Ubiquitous generalized tripod gait", *Nonlinear Dynamics*, vol. 112, pp. 15549-15565, 2024.
`doi:10.1007/s11071-024-09830-2`

Moreover, three unpublished works (Preprints) developed during the doctoral studies are included in this doctoral thesis as additional material.

[5] C. Mayora-Cebollero, F.H. Fenton, M. Halprin, C. Herndon, M.J. Toye, and R. Barrio, "Deep Learning for analyzing chaotic dynamics in biological time series: Insights from frog heart signals", *Preprint*, 2024.

[6] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, "Graphical preprocessing techniques to generalize dynamical behavior analysis with Deep Learning", *Preprint*, 2024.

[7] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, "Bregman Proximal Gradient with extrapolation to train a Reservoir Computing network", *Preprint*, 2024.

Table of Contents

Resumen

Los análisis de sistemas dinámicos son de gran relevancia para detectar o explicar ciertos fenómenos. Para ello, se usan técnicas como la continuación numérica de bifurcaciones o el espectro de exponentes de Lyapunov.

En el caso de las células excitables como las neuronas y los cardiomiocitos (células musculares cardiacas), que suelen tener un comportamiento de tipo *bursting* (con una dinámica tipo *fast-slow*), es importante entender fenómenos como el proceso de *spike-adding*, que provoca cambios en las órbitas periódicas y en los atractores caóticos. Para estudiarlos, es necesario utilizar un modelo matemático que sea computacionalmente simple y al mismo tiempo replique adecuadamente el comportamiento básico de una célula excitable. Uno de estos modelos es el modelo de Hindmarsh-Rose, en el que utilizando técnicas de continuación numérica de bifurcaciones, plantillas topológicas y bifurcaciones geométricas se puede explicar el fenómeno de *spike-adding*, su impacto en la estructura de los atractores caóticos, y los cambios drásticos de la dinámica en el espacio de parámetros. Además, con estos y otros métodos se puede estudiar dicho fenómeno de *spike-adding* en el movimiento de insectos y en la dinámica cardiaca a nivel celular a través de modelos más realistas.

Aunque muy útiles y necesarias, algunas de las técnicas utilizadas para estos análisis dinámicos son computacionalmente caras. Por eso, analizar si se pueden realizar dichos estudios con otros métodos como el aprendizaje automático (*Machine Learning*) y, en particular, el aprendizaje profundo (*Deep Learning*), es fundamental.

El aprendizaje automático es el campo de la Inteligencia Artificial (IA) enfocado en programar sistemas que puedan aprender de los datos. Las redes neuronales artificiales (*Artificial Neural Networks*), a las que nos podemos referir como aprendizaje profundo, son un algoritmo de aprendizaje automático diseñado para aprender de datos complejos, es decir, con varios niveles de abstracción. Algunas redes muy conocidas son el perceptrón multicapa (*Multi-Layer Perceptron*), las redes neuronales convolucionales (*Convolutional Neural Networks*), y las redes neuronales recurrentes (*Recurrent Neural Networks*).

El aprendizaje profundo permite realizar estudios de detección de caos de un sistema dinámico en el espacio paramétrico con gran precisión y reduciendo el tiempo computacional respecto al utilizado por las técnicas clásicas. Por ejemplo, nos permite obtener un estudio triparamétrico denso del sistema de Lorenz disminuyendo el tiempo de computación en un 95% aproximadamente respecto a las técnicas estándar. Además, aplicando métodos de preprocesamiento gráfico (*time series imaging*), se puede analizar el comportamiento dinámico en varios modelos discretos utilizando una red neuronal artificial que ha sido entrenada exclusivamente con datos del mapa logístico.

Cuando se trabaja con datos experimentales, son múltiples los inconvenientes a los que tenemos que hacer frente (pequeñas cantidades de datos, grabaciones cortas y ruidosas, etc.) y que dificultan el uso tanto de técnicas clásicas como de aprendizaje profundo. En estos casos, es necesario complementar los métodos con estrategias adecuadas para obtener algoritmos automáticos apropiados. Para realizar detección de caos en series temporales experimentales de un cardiomiocito de rana, el uso de redes neuronales artificiales entrenadas en el mapa logístico y su posterior validación con un modelo matemático de la misma naturaleza que los datos parece dar buenos resultados.

Además de detectar caos y hacer estudios de comportamiento dinámico (tareas de clasificación), el aprendizaje profundo nos permite cuantificar la caoticidad de un sistema. El espectro de exponentes de Lyapunov es una propiedad de los sistemas dinámicos que permite caracterizar su dinámica. Utilizando aprendizaje profundo, podemos obtener el espectro completo de exponentes de Lyapunov usando series temporales de una única variable del sistema. En general, en los métodos estándar se necesitan todas las variables del sistema para obtener todos los exponentes, si únicamente se utiliza una variable sólo se puede aproximar el conocido como máximo exponente de Lyapunov (*maximum Lyapunov exponent*). Otra de las ventajas que nos proporciona el aprendizaje profundo respecto a las técnicas clásicas es la reducción del tiempo de computación en un 90% aproximadamente.

El aprendizaje profundo parece ser un método prometedor para el análisis de sistemas dinámicos al proporcionar buenos resultados y reducir el tiempo de computación. Sin embargo, cambiando el punto de vista, también podemos utilizar las matemáticas, y los sistemas dinámicos en particular, para mejorar las técnicas de aprendizaje profundo.

Uno de los puntos clave en el aprendizaje profundo es el entrenamiento de las redes neuronales artificiales, que involucra un problema de minimización. En general, se suelen utilizar optimizadores derivados del método de gradiente descendiente (*Gradient Descent*) como el Adam. Sin embargo, al existir múltiples familias de optimizadores en la literatura, explorar otras opciones es interesante. En particular, el optimizador conocido como BPGe por sus siglas en inglés (*Bregman Proximal Gradient with extrapolation*) adaptado a una red de tipo recurrente (*Reservoir Computing*) para tareas de clasificación binaria proporciona resultados competitivos y una convergencia más rápida que los optimizadores más utilizados.

Estudios dinámicos realizados en una red de generadores centrales de patrones (*Central Pattern Generator*) utilizada para simular el movimiento de hexápodos han mostrado que el patrón de movimiento dominante es el conocido como *tripod gait* (tres patas en movimiento y tres en reposo). Dado que la red utilizada tiene una conectividad bipartita y el patrón dominante obtenido es bipartito, parece haber una correlación entre la topología de la red y la activación de las neuronas. De hecho, estudiando todas las redes de 6 a 9 neuronas con conectividad bipartita se ha obtenido que el patrón bipartito de cada configuración es siempre dominante. Análisis de este tipo que relacionan la estructura de una red y su dinámica proporcionan un buen punto de partida para el estudio de otro punto clave del aprendizaje profundo: la relación entre la arquitectura de la red neuronal artificial y su funcionamiento.

Abstract

The analyses of dynamical systems are very useful to detect and explain some phenomena. They are carried out using techniques such as numerical continuation of bifurcations and the full Lyapunov exponents spectrum, among others.

In the case of excitable cells such as neurons and cardiomyocytes (cardiac muscle cells), that usually present bursting behavior (following fast-slow dynamics), it is important to understand some phenomena like the spike-adding process, which produces changes in periodic orbits and chaotic attractors. To study such phenomena, it is necessary to use mathematical models that preserve the basic behavior of an excitable cell while being computationally simple. One of these models is the Hindmarsh-Rose model, in which using numerical continuation of bifurcations, topological templates and geometric bifurcations we can explain the spike-adding process, its impact in the structure of the chaotic attractors, and the drastic changes in the dynamics throughout the parameter space. Moreover, with these and other techniques we can study the spike-adding phenomenon in more realistic models related to insect movement and single-cell cardiac dynamics.

Although these methods used for dynamical analyses are very useful, some of them are highly computationally expensive. For this reason, it is necessary to study if these analyses can be carried out using other techniques as Machine Learning and, in particular, Deep Learning.

Machine Learning (ML) is the field of Artificial Intelligence (AI) devoted to programming systems capable of learning from data. Artificial Neural Networks (ANNs), which we can refer to as Deep Learning (DL), are a Machine Learning algorithm focused on learning from complex data, that is, data with several levels of abstraction. Some commonly used Artificial Neural Networks are the Multi-Layer Perceptron (MLP), the Convolutional Neural Networks (CNNs), and the Recurrent Neural Networks (RNNs).

Deep Learning allows performing chaos detection analyses of a dynamical system in the parameter space with high accuracy, while reducing computational time compared to classical techniques. For example, we can carry out a dense triparametric study of the Lorenz system and time savings are approximately 95% relative to standard techniques. In addition, we can analyze the dynamical behavior of several discrete dynamical systems using graphical preprocessing methods (time series imaging) and an Artificial Neural Network only trained with data from the Logistic map model.

When working with experimental data, there are many drawbacks that we have to face (small amount of data, short and noisy recordings, etc.), and that make difficult to apply both classical and Deep Learning techniques. In these situations, it is necessary to use certain strategies to complement the methods and obtain appropriate automatic algorithms. To perform chaos detection in experimental time series of a frog cardiomyocyte, the use of Artificial Neural Networks trained in the paradigmatic Logistic map model and their subsequent validation with a mathematical model of the same nature as the real-world data seems to provide good results.

In addition to detecting chaos and carrying out behavior dynamical analyses (classification tasks), Deep Learning allows quantifying the chaoticity of the systems. The Lyapunov exponents spectrum is a property of dynamical systems that allows characterizing their dynamics. With Deep Learning, we can obtain the full Lyapunov exponents spectrum using just single-variable time series. In general, standard methods need all the system variables to obtain all the Lyapunov exponents, and if only single-variable time series are used, just the maximum Lyapunov exponent is approximated. Another advantage of Deep Learning compared to classical techniques is the time savings, which are approximately 90%.

Deep Learning seems to be a promising technique to study dynamical systems as it provides good results while reducing computational time. However, if we change the point of view, we can also use mathematics, and dynamical systems in particular, to improve Deep Learning.

One of the key points of Artificial Neural Networks is the training process, which involves a minimization problem. In general, optimizers derived from the well-known Gradient Descent method, such as Adam, are used. But, as in the literature there are many other families of optimizers, it is interesting to explore more options. In particular, the Bregman Proximal Gradient with extrapolation (BPGe) method reformulated to perform binary classification tasks with a recurrent-like network (Reservoir Computing) provides competitive results and faster convergence compared to commonly used optimizers.

Dynamical studies of a Central Pattern Generator used to emulate the locomotion of hexapods have shown that the dominant movement pattern is the so-called tripod gait (three legs moving and three at rest). Since the considered network has a bipartite connectivity and the obtained dominant pattern is bipartite, there seems to be a correlation between the topology of the network and the activation of the neurons. In fact, if we study all the network of 6 to 9 neurons with bipartite connectivity, we obtain that the bipartite pattern of each configuration is always the dominant one. This kind of analyses that relates the structure of a network and its dynamics can shed light on the study of another key point of Deep Learning: the relationship between the architecture of the Artificial Neural Network and how it works.

Introduction

The dynamical studies performed using numerical continuation of bifurcations, spike-counting sweeping technique, or Lyapunov exponents spectrum, among others, are of great interest to understand some underlying phenomena of the systems, such as different regimes in the parameter space, multistability, or spike-adding processes. When applied to problems in clinical or biological contexts, they can provide clear insight into what is happening and reveal the core of the phenomena.

The dynamics of the excitable cells, such as neurons and cardiomyocytes (cardiac muscle cells), is mainly described by their membrane potential. Cells are covered by a membrane with various functions such as protecting them from the environment and regulating ionic concentrations through ion channels. The difference in potential between both sides of the membrane (outside and inside the cell) is the so-called membrane potential. The electrical signals generated as a consequence of changes in the membrane potential are known as action potentials. The dynamical analyses of the action potentials of excitable cells can provide insight into their behavior.

The first mathematical model that described the action potential of a neuron is the well-known Hodgkin-Huxley model [8]. Because of the similarities between neurons and cardiac muscle cells, this neuron model has been considered as a paradigmatic example for the development of both neuron and cardiomyocyte models. In fact, the Hodgkin-Huxley model has facilitated the creation of simplified models that, while preserving the basic behavior of an excitable cell, are computationally simple enough to study their dynamics using dynamical systems techniques. One of these simplified systems is the Hindmarsh-Rose model [9], that has been widely studied [10, 11, 12, 13]. It is a neuron model with bursting behavior given by three nonlinear ordinary differential equations. It has 8 parameters, one of them is the so-called small parameter ε that controls the fast-slow dynamics of bursting behavior [14]. Moreover, the spike-adding phenomenon is present in this system.

The spike-adding process in a fast-slow system results in the addition of one extra spike in a bursting orbit. This process is significant because of the changes it can induce in periodic orbits and chaotic attractors [13, 15, 16, 17]. From cardiac dynamics perspective, the spike-adding process corresponds to the creation of Early Afterdepolarizations. An Early Afterdepolarization is an unexpected rise in the membrane potential of a cardiac muscle cell that occurs at a specific phase of the action potential. Under some circumstances, they can give rise to abnormal electrical activity [18], which makes understanding their creation a relevant topic.

In the published article **Dynamics of excitable cells: Spike-adding phenomena in action** (*SeMA Journal*) [1] we study the spike-adding process in neurons and cardiomyocytes using techniques such as spike-counting sweeping, numerical continuation of bifurcations, among others. In particular, we perform a global study of the fast-slow Hindmarsh-Rose model (with explicit parameter ε). The objectives of this analysis (which can be extended to other neuron models) are to highlight the significant role of certain bifurcations in the spike-adding phenomenon [19, 20, 21], to explain using geometric bifurcations [22] the drastic changes in the dynamics when the small parameter ε is varied beyond its standard range [23], and to use topological templates [24] to investigate how the spike-adding process impacts the topological structure of chaotic attractors [25]. Moreover, we consider the aforementioned creation of Early Afterdepolarizations, and the transition between gaits in insects movement (modeled using Central Pattern Generators [26, 27]) to emphasize the importance of the spike-adding phenomenon (and consequently its dynamical analysis) in real processes [28, 29, 30].

Some techniques used to perform dynamical studies, like the ones in [1], provide detailed analyses, but they can be highly computationally expensive. In recent years, Machine Learning (ML), and in particular Deep Learning (DL), has been proposed as an alternative method to deal with dynamical analyses as chaos detection (or similar studies as classification of arrhythmias) [31, 32, 33, 34, 35], approximation of Lyapunov exponents (LEs) [36, 37, 38, 39, 40, 41], forecasting [42, 43, 44, 45], detection and characterization of basins of attraction [46, 47], identification of critical transitions and bifurcations [48, 49, 50, 51], and detection of chimera states [52, 53], among others.

Dynamical behavior analyses in the parameter space have allowed us to study in detail the global dynamics of many systems [54, 55, 56, 57]. In general, these studies are performed using classical techniques for the computation of Lyapunov exponents [58, 59, 60], fast chaos indicators [61], or the 0-1 test for chaos [62]. As already mentioned, Deep Learning has been used to detect chaotic behavior in different dynamical systems [33, 34, 35]. However, one main question arises: Can Deep Learning provide detailed parametric studies while also offering advantages over classical methods? To answer this question is the main objective of the published article **Deep Learning for chaos detection** (*Chaos: An Interdisciplinary Journal of Nonlinear Science*) [2]. In this paper we show how DL networks, once trained in a few one-parameter lines of a dynamical system, are able to determine the behavior of the whole parameter space of such system using just short time series. In particular, a dense triparametric study of the Lorenz system [63] can be obtained with a Long Short-Term Memory (LSTM) network with an accuracy of almost 100% in the detection of regular and chaotic regions and a reduction of approximately 95% in computational time compared to classical techniques [58]. This result highlights the power of Deep Learning for dynamical analyses. Although some previous articles [54, 55] reconstructed the triparametric analysis of the Lorenz system using biparametric planes, to the best of authors' knowledge this is the first time in the literature that a dense triparametric study of the Lorenz system is obtained with DL or any other method.

One of the key points of Deep Learning is to have enough data to train the network. However, in some cases, the available data is limited. This can occur when working with experimental data. It is necessary to analyze real-world data, but sometimes the small number of samples and the short noisy recordings can make its study a challenging task for both classical and Deep Learning techniques [64]. Our objective in the unpublished work **Deep Learning for analyzing chaotic dynamics in biological time series: Insights from frog heart signals** (*Preprint*) [5] is to define an automatic DL-based algorithm to perform chaos analysis of data obtained experimentally from a frog cardiomyocyte. In particular, the experimental dataset consists of short time series whose elements correspond to the duration of the action potentials of membrane potential signals from a single cardiac muscle cell under different pacing rates. Some pathologies such as arrhythmias are related to the well-proven phenomenon of chaos in the heart [65, 66, 67], so the detection of chaos in whatever type of experimental heart time series is crucial. The algorithm [5] described to deal with this chaos analysis in real-world data is based on the universality of the Logistic map [68] and the validation using an appropriate mathematical model. Several LSTM networks are trained to perform the chaos detection task in the Logistic map following the approach of the previous article [2]. Then, all of them carry out chaos analyses in a cardiac mathematical model [69] that emulates, for different pacing rates, the duration of the action potentials in a membrane potential signal, that is, the dynamics of the real-world data. Some restrictive criteria are defined to select the network that has successfully detected the behavior of the cardiac model. Such network is used to obtain the chaos analysis of the experimental dataset. The accuracy of the results is of 90% approximately. This exhibits the utility of Deep Learning combined with other tools (as validity in mathematical models) to carry out studies that because of the nature of the data (limited short noisy time series) would be complicated with standard techniques.

Notice that one of the limitations of the DL approach for chaos detection proposed in the article [2] is that the parametric analyses are performed in the same system in which the DL network has been trained. As demonstrated in the preprint [5], it is not immediate to use a network trained in a particular system to perform dynamical behavior analyses in another system. In the unpublished work **Graphical preprocessing techniques to generalize dynamical behavior analysis with Deep Learning** (*Preprint*) [6], our objective is to deal with this lack of generality. For this purpose, we adapt one of the workflows proposed in [70]. In such article, the authors summarize all the possible combinations of Machine Learning and Recurrence Plots (RPs) [71] that can be useful to perform a classification or a prediction task in a dynamical system. While in [72] the authors perform dynamical classification applying the workflow that uses Recurrence Quantification Analysis (measures provided by the Recurrence Plot) as input information for Machine Learning methods, we propose to consider the procedure that directly uses the Recurrence Plot as input for a Convolutional Neural Network. In [6] we utilize the Recurrence Plot and two time series imaging methods (Gramian Angular Field -GAF- [73, 74], and Markov Transition Field -MTF- [73, 74]) as graphical preprocessing techniques to convert time series into images that embed their dynamics and can be used as input in a Convolutional Neural Network. This allows us to generalize dynamical behavior analysis to different discrete dynamical systems. That is, complete biparametric studies of the Circle map [75] and the Hénon map [76] can be obtained using a DL network trained only with data from the Logistic map [68]. This represents a significant improvement over usual classical techniques of Lyapunov exponents [77] that are used for dynamical classification purposes: while these standard methods particularize the formula for each discrete dynamical system, in our Deep Learning approach just the time series are involved in the studies and no particularization is necessary.

However, the possible behaviors of a dynamical system can be classified into a broader range of dynamical regimes (such as hyperchaos or tori) beyond the regular and chaotic ones detected with a chaos detection strategy [2, 6]. The Lyapunov exponents spectrum of a dynamical system is a property that permits us to characterize its dynamics easily. For instance, if the first two LEs are positive, the system presents hyperchaos, while if both are zero, torus dynamics emerge. Some classical algorithms [58] that use information from all the system variables provide the full Lyapunov exponents spectrum, while other methods [58, 59] approximate just the first Lyapunov exponent (the so-called maximum Lyapunov exponent) using single-variable time series. As indicated previously, Deep Learning techniques have been proposed for LEs approximation. In particular, DL has been used to directly approximate the LE of a one-dimensional discrete dynamical system [40], or as a forecasting or data assimilation tool to obtain the proper information to apply standard techniques to compute LEs [36, 37, 38, 39, 41]. Our objective in the published article **Full Lyapunov exponents spectrum with Deep Learning from single-variable time series** (*Physica D: Nonlinear Phenomena*) [3] is to use Deep Learning techniques to approximate the full Lyapunov exponents spectrum of a dynamical system using just short single-variable time series and no additional dynamical information. Whether training the DL network with a few one-parameter lines or with random data (which provides more dynamical variability), a good estimation of the full Lyapunov exponents spectrum in a whole biparametric plane can be obtained with time savings of more than 90% compared to standard techniques [58]. In fact, an accurate dynamical study (chaos, hyperchaos and tori) of a biparametric plane of the six-dimensional system of two almost-identical coupled Lorenz models (coupling strategy of [78] has been used) is carried out with the DL approximations of the Lyapunov exponents. These results highlight that Deep Learning is a powerful technique for dynamical systems studies. To the best of the authors' knowledge, no other techniques can estimate the full Lyapunov exponents spectrum only using single-variable time series.

As demonstrated, Machine Learning and, in particular, Deep Learning are promising techniques to perform dynamical systems studies. However, we can change the point of view: Can dynamical systems theory and mathematics help improve Machine Learning? Bifurcation analyses, Lyapunov exponents, and graph theory, among others, have been used to provide insights and to improve Artificial Neural Networks [79, 80, 81, 82].

To train the Deep Learning networks used in previous works [2, 5, 6, 3], we have applied the commonly used optimizers Stochastic Gradient Descent (SGD) [83] and Adaptive Moment estimation (Adam) [84]. Both, as well as Root Mean Square Propagation (RMSProp) [85], are variants of the well-known Gradient Descent. Nevertheless, in the literature there are other families of optimizers as those based on Proximal Gradient algorithms [86, 87]. Can any of these optimizers outperform the standard ones? Our objective in the unpublished work **Bregman Proximal Gradient with extrapolation to train a Reservoir Computing network** (*Preprint*) [7] is to adapt the recently introduced Bregman Proximal Gradient with extrapolation (BPGe) optimization algorithm [88] to the supervised training process of a Reservoir Computing network for binary classification tasks, and compare it with the optimizers mentioned above that are widely used (SGD, RMSProp and Adam). BPGe can deal with non-convex and non-smooth minimization problems, which makes it really appropriate to train Artificial Neural Networks. While considering a Reservoir Computing network (in particular, an Echo State Network) facilitates the adaptation of BPGe to supervised DL training, working with a binary classification task and consequently with the Cross-Entropy loss function makes it a bit challenging. Comparing SGD, RMSProp, Adam and BPGe in two binary classification problems of different nature, we can conclude that the new proposed optimizer is highly competitive. In addition to provide better results, Bregman Proximal Gradient with extrapolation highlights because of its convergence speed (with just a few number of epochs good accuracy is achieved). The obtained results make evident that Deep Learning can benefit from the application of new optimization methods.

Central Pattern Generators (CPGs) are small neuron networks driving some rhythmic and coordinated biological processes. In fact, once activated, CPGs generate rhythmic motor patterns even in absence of sensory input. In [28, 89] (and briefly in previous work [1]), the bursting neuron CPG model of Ghigliazza and Holmes [26, 27] with bipartite connectivity for the movement of insects (cockroaches) is studied, concluding that the dominant gait pattern throughout the parameter space seems to be the tripod gait. This pattern corresponds to two disjoint subgroups of synchronized neurons (from the point of view of insect movement, three legs moving and three at rest). As the analyzed network has bipartite connectivity and the tripod gait corresponds to a bipartite pattern, this ubiquity suggests a possible correlation between the patterns of activation of the neurons and the topology of the network. This is an interesting topic since bipartite networks naturally emerge in many situations [90], and the interplay between network topology (underlying graph) and its dynamics [81] can shed light on the relationship between the architecture (number of neurons, connectivity, ...) of an Artificial Neural Network and how it works. The main objective of the published article **Dominant patterns in small directed bipartite networks: Ubiquitous generalized tripod gait** (*Nonlinear Dynamics*) [4] is to analyze if the bipartite pattern is the predominant one in all bipartite networks of 6 to 9 neurons. Using the quasi-Monte Carlo sweeping technique [89], the dominance of bipartite pattern in the studied bipartite networks is exhibited. In addition, in cyclic graph networks with non-bipartite connectivities, the star traveling wave patterns are the dominant ones. These analyses show the correlation between network topology and dominant patterns.

1.1 A Brief Introduction to Machine Learning

Machine Learning (ML) [83, 91] is the field of Artificial Intelligence (AI) focused on programming systems capable of learning from data. It includes algorithms such as Support Vector Machines, Principal Component Analysis, Reinforcement Learning, and Artificial Neural Networks.

Artificial Neural Networks (ANNs) are ML architectures based on their biological counterparts. At the end of the 20th century, Deep Learning (DL) was described as the branch of Machine Learning including ANNs with at least three hidden layers. However, nowadays, as it is common to use a large number of layers, the term Deep Learning is ambiguous and in general it refers to all Machine Learning algorithms and techniques related to Artificial Neural Networks (with any restriction in the number of hidden layers) devoted to learn from complex data (that is, with multiple levels of abstraction).

The first artificial neuron, proposed by McCulloch and Pitts in 1943 [92], was based on propositional logic (binary inputs and a binary output). Later, the Threshold Linear Unit (TLU) with numerical (not necessarily binary) inputs and outputs was the basic unit (artificial neuron) to construct the simplest ANN architecture. It was the Perceptron, a layer of TLUs developed by Rosenblatt in 1958 [93]. To face some limitations, several Perceptrons were stacked creating the Multi-Layer Perceptron architecture.

The Multi-Layer Perceptron (MLP) is a feed-forward fully connected ANN with an input layer, $m-1$ hidden layers and an output layer that computes as

$$y = \mathscr{A}(W^{[m]}\mathscr{A}(W^{[m-1]}(\cdots(\mathscr{A}(W^{[2]}\mathscr{A}(W^{[1]}x + b^{[1]}) + b^{[2]})\cdots) + b^{[m-1]}) + b^{[m]}),$$

where x is the input, y is the output, $\mathscr{A}$ is the (non-linear) activation function, $\{W^{[i]}\}_{i=1}^{m}$ are the weights of the connections between layers $i-1$ and i (layers enumerated from input to output with index 0 for input layer and index m for output layer), and $\{b^{[i]}\}_{i=1}^{m}$ is the bias of each layer. The information flow goes from the input to the output layer, passing through all hidden layers from index 1 to index $m-1$ (feed-forward ANN). The activation of the neurons of each layer is the result of applying a (non-linear) function to the linear combination of the activation of all the neurons of the previous layer (fully connected ANN). Despite its simplicity, we have obtained quite good results using an MLP to perform chaos detection in a dynamical system [2]. See Figure 1.1 for a graphical representation of an MLP.

Notice that an ANN is used to process input data in order to obtain the correct output (or at least a good approximation). However, for this we need an appropriate value for the weights and biases involved in the computations of the network. These weights and biases are the so-called trainable parameters fine-tuned during training. This training process is actually a minimization problem that consists of finding the value of the trainable parameters that minimizes the loss function (error between the network output and the expected output) for the given (training) data. That is, the network learns from the data to fit the weights and biases. Such minimization problem is usually solved using standard optimizers as SGD or Adam, but in [7] we have adapted the Bregman Proximal Gradient with extrapolation algorithm to deal with it. After training, to check that learning has been successful, unseen data (test dataset) is used to test performance.

In addition to the trainable parameters, Artificial Neural Networks have other parameters, such as the number of layers, that are not fine-tuned during training and we have to fix properly in advance. They are known as hyperparameters. In [4] we have seen that the topology of a bursting neuron network is correlated with its behavior. From this, we can infer the importance of setting hyperparameters appropriately in ANNs.

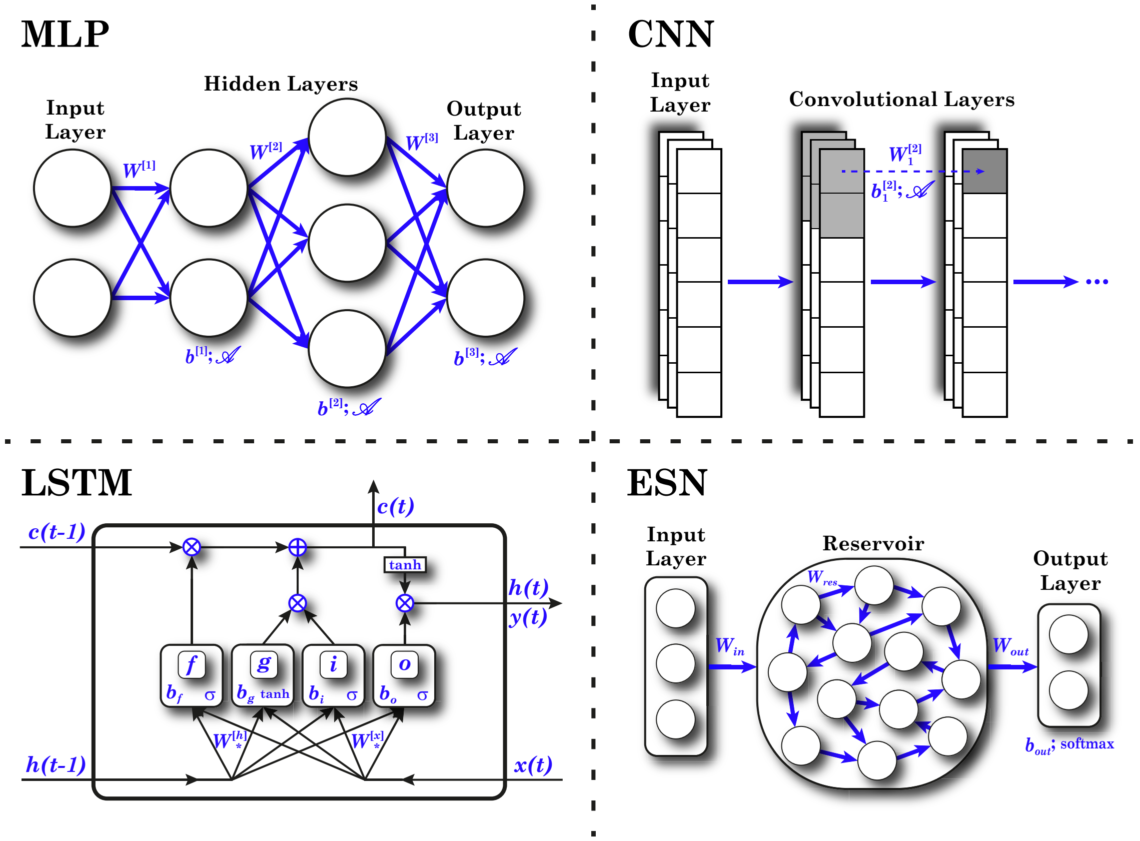


Figure 1.1: Graphical representation of several Artificial Neural Networks: Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) cell, and Echo State Network (ESN).

Several studies of the visual cortex, conducted by Hubel and Wiesel in 1959 [94, 95], showed that many neurons only react to stimuli in a small region of the visual field (local receptive field). Some of them react to different simple patterns, while others with larger receptive fields react to more complex structures that are combinations of the simpler ones. This suggests that neurons reacting to complex patterns work with the information from neurons processing the simpler ones. These ideas led to the development of the neocognitron by Fukushima in 1980 [96]. This architecture evolved to the well-known Convolutional Neural Network [97].

Convolutional Neural Networks (CNNs) are ANNs organized into convolutional layers (with several feature maps) that extract information, and pooling layers that reduce dimensionality. CNNs are not fully connected ANNs, they share weights across multiple neurons, and allow different input formats (vectors, matrices, tensors, etc.). In particular, in a 1D CNN, the activation $x_{i,j}^{[l]}$ of neuron i of feature map j in convolutional layer l is computed as

$$x_{i,j}^{[l]} = \mathscr{A}\left(b_j^{[l]} + \sum_{k=1}^{K^{[l]}} \sum_{f=1}^{F^{[l-1]}} w_{k,f,j}^{[l]} x_{k+(i-1)s^{[l]}+(k-1)(d^{[l]}-1),f}^{[l-1]}\right),$$

where $\mathscr{A}$ is the (non-linear) activation function, $b_j^{[l]}$ is the bias shared by neurons in feature map j of layer l, $W_j^{[l]} = \{w_{k,f,j}^{[l]}\}_{k=1,\ldots,K^{[l]};f=1,\ldots,F^{[l-1]}}$ is the filter or kernel (weight matrix) shared by all neurons in feature map j of layer l, $K^{[l]}$ is the dimension of the receptive field (kernel size) of neurons in layer l, $F^{[l]}$ is the number of feature maps in layer l, $s^{[l]}$ is the stride (distance between the receptive field of two consecutive neurons) of layer l, and $d^{[l]}$ is the dilation (distance between neurons of the same receptive field) of layer l. This type of architecture allows us to deal with several dynamical systems problems as chaos detection [2, 6] and the approximation of Lyapunov exponents [3]. See Figure 1.1 for a graphical representation of a CNN.

Another type of ANN that shares weights, but in time rather than in space like CNNs, is the Recurrent Neural Network (RNN). It is especially useful to process sequential data as, in some sense, it has memory (it retains information). Two examples of RNNs are the Long Short-Term Memory cell network and the Echo State Network.

The Long Short-Term Memory (LSTM) cell, proposed by Hochreiter and Schmidhuber in 1997 [98], is a modification of the basic cell of original RNNs that increases the long-term memory. The information is retained through two states $h(t)$ and $c(t)$ that at time step t are updated as

$$c(t) = f(t) \otimes c(t-1) + i(t) \otimes g(t),\ h(t) = o(t) \otimes \tanh(c(t)),$$

with $\otimes$ the element-wise product, tanh the hyperbolic tangent activation function, and $f(t)$, $g(t)$, $i(t)$ and $o(t)$ given by

$$\begin{gathered} f(t) = \sigma(W_f^{[x]} x(t) + W_f^{[h]} h(t-1) + b_f),\ g(t) = \tanh(W_g^{[x]} x(t) + W_g^{[h]} h(t-1) + b_g), \\ i(t) = \sigma(W_i^{[x]} x(t) + W_i^{[h]} h(t-1) + b_i),\ o(t) = \sigma(W_o^{[x]} x(t) + W_o^{[h]} h(t-1) + b_o). \end{gathered}$$

In these expressions, $x(t)$ is the external input information at time t, σ is the sigmoid activation function, and $W_*^{[\{x,h\}]}$ and b_* ($* \in \{f,g,i,o\}$) are the weights and biases shared through time. It is common to take the output of the cell $y(t)$ equal to the state $h(t)$. Notice that as the sigmoid activation function σ is applied, f, i and o act as gates that filter the information. This ANN has allowed us to analyze chaotic dynamics in both theoretical [2] and experimental data [5]. See Figure 1.1 for a graphical representation of an LSTM cell.

The Echo State Network (ESN), described by Jaeger in 2001 [99], is a type of Reservoir Computing (RC) architecture [100]. In particular, it is a Recurrent Neural Network with three main structures: the input layer, the reservoir (set of connected neurons), and the output layer. Particularizing to a classification task, it works as

$$h(t) = \alpha \tanh(W_{in} x(t) + W_{res} h(t-1)) + (1-\alpha) h(t-1) \text{ for } t \in [1,T], \quad y = \text{softmax}(W_{out} h(T) + b_{out}),$$

where $x(t)$ is the external input at time t, $h(t)$ is the state of the reservoir neurons at time t, $\alpha \in [0,1]$ is the leaking rate parameter, tanh and softmax are the hyperbolic tangent and the softmax activation functions, T is the length of the input sequence, b_{out} is the bias term of the output layer, and W_{in}, W_{res} and W_{out} are the weights of the connections of the input layer with the reservoir, the reservoir with itself, and the reservoir with the output layer, respectively. Just W_{out} and b_{out} have to be fitted during training, the remaining weights are set randomly (following some recommendations [101]). This fact reduces considerably the training process, and has facilitated us the adaptation of Bregman Proximal Gradient with extrapolation algorithm to ANNs [7]. See Figure 1.1 for a graphical representation of an ESN.

Published Articles

This chapter includes the four published articles of the doctoral thesis of Carmen Mayora Cebollero:

[1] R. Barrio, S. Ibáñez, J.A. Jover-Galtier, Á. Lozano, M.Á. Martínez, A. Mayora-Cebollero, C. Mayora-Cebollero, L. Pérez, S. Serrano, and R. Vigara, "Dynamics of excitable cells: Spike-adding phenomena in action", *SeMA Journal*, vol. 81, no. 1, pp. 113-146, 2024.
`doi:10.1007/s40324-023-00328-2`

[2] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 7, p. 073146, 2023.
`doi:10.1063/5.0143876`
Reprinted from R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection", Chaos: An Interdisciplinary Journal of Nonlinear Science, vol. 33, no. 7, p. 073146, 2023. https://doi.org/10.1063/5.0143876, with the permission of AIP Publishing.

[3] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, "Full Lyapunov exponents spectrum with Deep Learning from single-variable time series", *Physica D: Nonlinear Phenomena*, vol. 472, p. 134510, 2025.
`doi:10.1016/j.physd.2024.134510`

[4] Á. Lozano, R. Vigara, C. Mayora-Cebollero, and R. Barrio, "Dominant patterns in small directed bipartite networks: Ubiquitous generalized tripod gait", *Nonlinear Dynamics*, vol. 112, pp. 15549-15565, 2024.
`doi:10.1007/s11071-024-09830-2`

SeMA Journal
https://doi.org/10.1007/s40324-023-00328-2

Dynamics of excitable cells: spike-adding phenomena in action

Roberto Barrio[1] · Santiago Ibáñez[2] · Jorge A. Jover-Galtier[1] · Álvaro Lozano[3] · M. Ángeles Martínez[1] · Ana Mayora-Cebollero[1] · Carmen Mayora-Cebollero[1] · Lucía Pérez[2] · Sergio Serrano[1] · Rubén Vigara[1]

Received: 15 November 2022 / Accepted: 17 April 2023
© The Author(s) 2023

Abstract

We study the dynamics of action potentials of some electrically excitable cells: neurons and cardiac muscle cells. Bursting, following a fast–slow dynamics, is the most characteristic behavior of these dynamical systems, and the number of spikes may change due to spike-adding phenomenon. Using analytical and numerical methods we give, by focusing on the paradigmatic 3D Hindmarsh–Rose neuron model, a review of recent results on the global organization of the parameter space of neuron models with bursting regions occurring between saddle-node and homoclinic bifurcations (fold/hom bursting). We provide a generic overview of the different bursting regimes that appear in the parametric phase space of the model and the bifurcations among them. These techniques are applied in two realistic frameworks: insect movement gait changes and the appearance of Early Afterdepolarizations in cardiac dynamics.

Keywords Excitable cells · Bifurcations · Mathematical neurons · Cardiomyocytes · Dynamical systems · Fast–slow systems

Mathematics Subject Classification 34C23 · 34Cxx · 37G15 · 37G20 · 92C20 · 92B05

1 Introduction

Cells are covered with a membrane that isolates and protects them from the environment. It also acts as a regulator of ionic concentrations in the cellular plasma of important elements for biological processes, such as Ca, K, etc. As a consequence, the ionic concentration of some elements is different on both sides of the membrane. Since the relative permeability of the cell membrane is also different for those elements, there exists a difference in potential between the inside and outside of the cell, which is called the membrane potential. This is particularly important for the study of electrically excitable cells (muscle cells, secretory cells

✉ Roberto Barrio
rbarrio@unizar.es

Extended author information available on the last page of the article

Published online: 17 May 2023

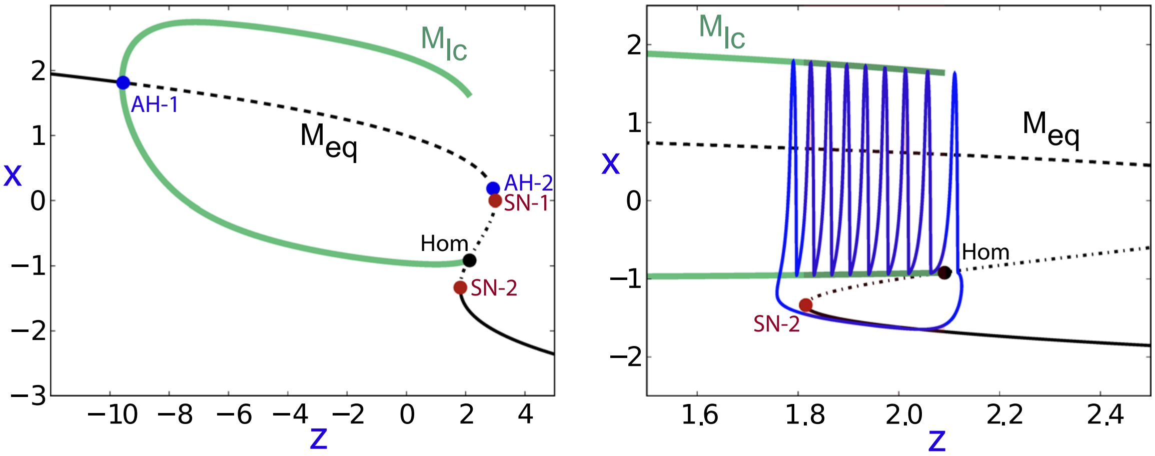


Fig. 1 Left: Fast/slow decomposition that illustrates the bifurcation scenario in which a fold/hom (or square-wave) bursting occurs in the HR model (1). Fixed points occur along the M_{eq} curve (stable when continuous, unstable when discontinuous), and limit cycles occur between both branches of M_{lc}. Andronov–Hopf (AH), saddle-node (SN) and primary homoclinic (Hom) bifurcations are shown. Right: Zoom showing a fold/hom bursting orbit in blue. For more details see [7]

and neurons), which perform their functions by actively changing their membrane potential and thus generating electrical signals, called action potentials (APs). Therefore, to understand the behavior of an excitable cell we should study the dynamics of their APs. The excitable cells that concern us are neurons and cardiac muscle cells.

Hodgkin and Huxley proposed the first neuron (HH) model [1] describing the action potentials in the neuron membrane. It has given rise to several mathematical models for diverse kinds of neurons in different animals. Among them, some simplified models, like the 3D Hindmarsh–Rose (HR) model, have been developed to help in the study of realistic models based on the HH framework [1]. We consider the HR model [2] for a detailed analysis (a similar study can be carried out to other models) given that, even if it is computationally simple, it reproduces quite well the rich firing patterns exhibited by a biological neuron as well as the main behavior in general. The HR model is the following set of three nonlinear ODEs

$$\begin{cases} \dot{x} = y - ax^3 + bx^2 - z + I, \\ \dot{y} = c - dx^2 - y, \\ \dot{z} = \varepsilon[s(x - x_0) - z], \end{cases} \tag{1}$$

where x, y and z are the membrane potential, the fast and the slow gating variables, respectively. Regarding parameters, (a, c, d, s, x_0) are set to the typical values $(1, 1, 5, 4, -1.6)$, (b, I) take values in specific ranges where the bursting or spiking behavior is exhibited and, along the paper, the value of the small parameter ε is taken from different intervals in which it has different magnitude to achieve a global analysis of this fast–slow system.

Bursting is the main behavior present in neuron models, and follows a fast–slow dynamics [3] (it is also common in laser dynamics and chemical reactions, among other practical applications [4, 5]). The fold/hom (or square-wave) bursting, following the Izhikevich [6] classification, is one of the main bursting regimes. In a fold/hom bursting, the active regime begins in a fold (or saddle-node) bifurcation of equilibria and ends at a homoclinic bifurcation in the fast subsystem. In Fig. 1, the conditions for this type of bursting in the HR model and one orbit are shown. Spike-adding bifurcations are a common set of bifurcations in systems that present fast–slow dynamics. Such special bifurcations give rise to the appearance of extra spikes (turns) in the fast manifold region. They are of interest due to the progressive change

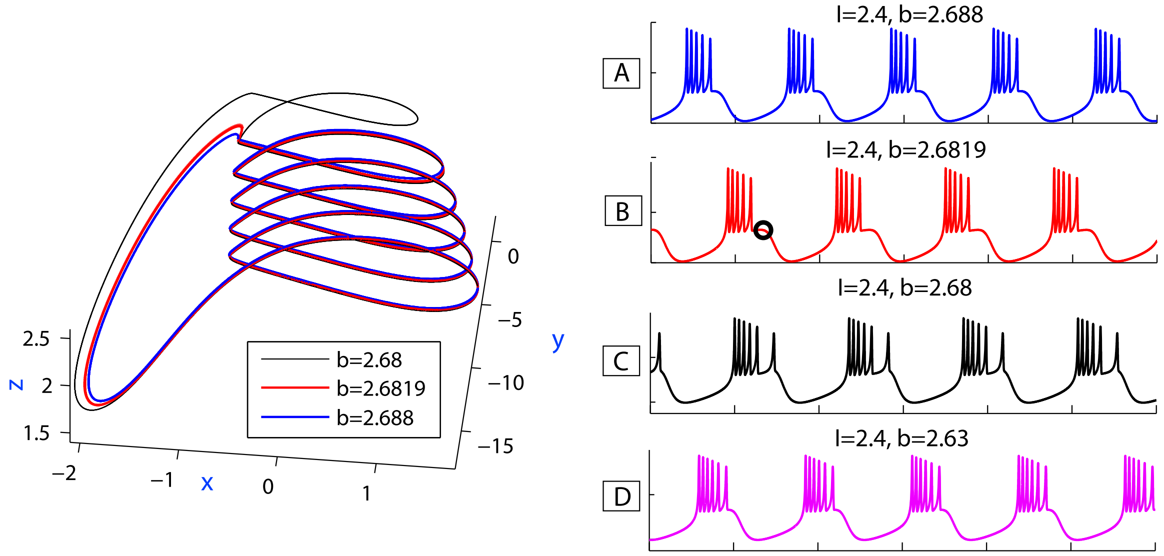


Fig. 2 Taken from [7] (Figure 6). Left: 3D projections of bursting orbits of the HR model (1) for three different values of parameter b. Right: Waveforms of four bursting orbits (three of them shown on the left) of Hindmarsh–Rose model for a fixed value of I and four distinct values of parameter b. A spike-adding process can be appreciated when going from A to D

that they cause in the spectrum of periodic orbits and the structure of chaotic attractors of the system [8–11]. In Fig. 2 several bursting orbits and a spike-adding process in the HR model are represented. In the aforementioned type of bursting, this spike-adding phenomenon is associated with the existence of canard explosions and certain codimension-two homoclinic bifurcations (inclination-flip and orbit-flip) [8, 12–14].

This paper is organized as follows. Section 2 deals with the description of the dynamical properties of the HR model, and is divided into three parts: Sect. 2.1 is devoted to studying slices of the three-parameter space for the HR model, with each slice determined by the values of ε within a standard range of small values; Sect. 2.2 presents the changes observed when we consider the full three-parameter space and the parameter ε is allowed to take larger values, and it also shows the geometric bifurcations that can be found in the HR model; Sect. 2.3 shows the limit behavior when the small parameter $\varepsilon \searrow 0$. Section 3 introduces more realistic models of Hodgkin–Huxley type, one used to analyze insect gait movement changes in Sect. 3.1, and the other to study Early Afterdepolarizations in cardiac dynamics in Subsection 3.2. Section 4 gives some conclusions. Finally, Appendix A provides for completion some theoretical aspects of the homoclinic bifurcations used in the paper.

2 Characterization of the HR neuron model

In this section we provide an exhaustive analysis of the dynamical characteristics of the HR model introduced in (1), and in particular we show how they depend on the values of the small parameter ε. Each of the following subsections deals with a different situation regarding this parameter.

2.1 Dynamics at standard values of the small parameter ε

In this section, we study the behavior of the HR model when the small parameter ε is in a standard range. The main bifurcations of the system are described, thus providing a complete description of the parameter space.

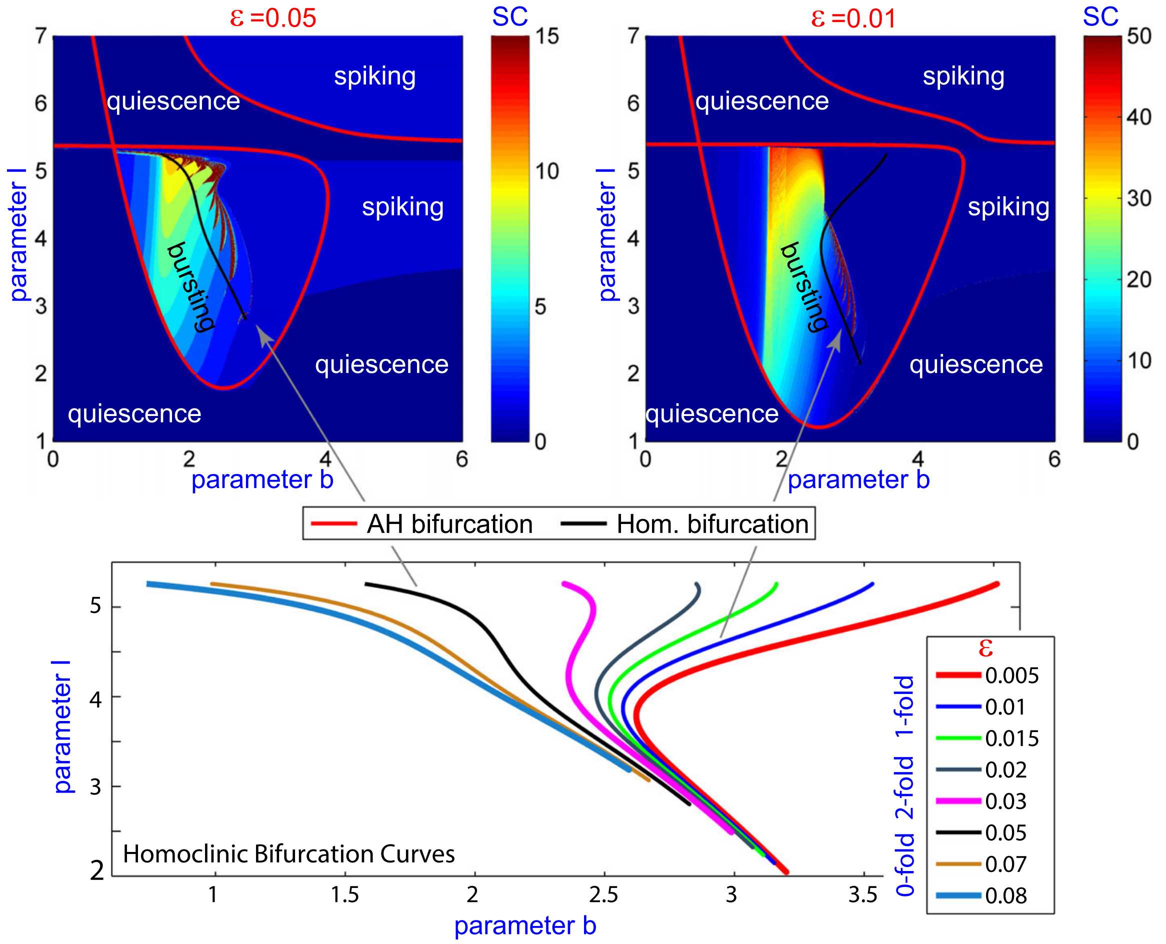


Fig. 3 Taken from [19] with modifications (Figures 1 and 4). Top: Spike-counting (SC) diagram on the (b, I) plane, for $\varepsilon = 0.05$ and $\varepsilon = 0.01$, classifying the AP as quiescent (0 spikes, i.e. fixed point), spiking (1 spike) or bursting (2 or more spikes). Andronov–Hopf (AH) and primary homoclinic (Hom) bifurcations correspond to curves shown in red and black, respectively. Bottom: The first primary homoclinic bifurcations curves are shown for different values of the small parameter ε. The curves have been computed using the continuation software AUTO [17, 18]

In recent years, the HR model has been thoroughly analyzed (see [7, 8, 15, 16]) applying different techniques, such as the spike-counting (SC) method [7, 16]. This technique, applied to a periodic orbit of a dynamical system, determines the number of spikes that a certain variable shows during a period. In the case of excitable cells, the SC technique allows us to compute the number of spikes of the membrane potential during each activation period of the cell. Figure 3 shows the number of spikes of the membrane potential for stable solutions of the HR model, codified with colors, along the (b, I)-plane. This allows us to distinguish between the regions of chaotic bursting, periodic tonic spiking and regular bursting. Stable spiking is shown as APs with a single spike, while quiescence of neurons occur when a fixed point is reached (in SC terms, the AP shows 0 spikes). Ranging from blue to green, we can see the stripes of a spike-adding cascade that corresponds to bursting that becomes chaotic in a chain of onion-like bulbs colored in brown [8]. In this review paper, we briefly focus on the development of a global study of the complete homoclinic structure that gives rise to the complete phenomena. To that goal, a detailed numerical study with continuation techniques is required (we used the well-known software AUTO [17, 18]) as well as with the SC technique.

Andronov–Hopf (AH) bifurcations, in red, and primary homoclinic (Hom) bifurcations, in black, are superimposed in both representations of Fig. 3. These bifurcations, further characterized in [19], correspond to bifurcations shown in Fig. 1. Also in [19], it was studied how

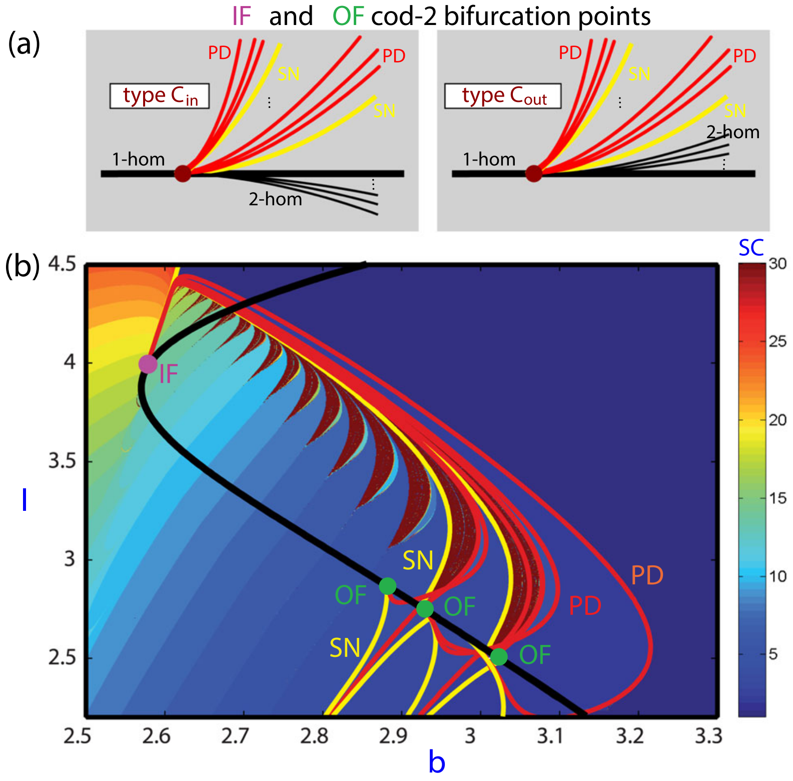


Fig. 4 **a** Theoretical unfolding of the codimension-two IF and OF homoclinic bifurcations of types C_{in} and C_{out} describing the pencils of period-doubling and fold bifurcations. **b** Spike-counting diagram on the (b, I) plane for $\varepsilon = 0.01$. Superimposed, the bifurcation curves (yellow—fold or saddle-node of periodic orbits, red—period-doubling (PD), black—primary homoclinic bifurcations) and some codimension-two bifurcation points (purple—IF and green—OF) (Color figure online)

the first primary homoclinic bifurcation curve modifies its geometry when different values of the small parameter ε are considered and how this phenomenon is related with being in a fast–slow regime or not. At the bottom of Fig. 3, it is shown that the number of "visible" foldings (with respect to parameter b) of the homoclinic curve varies when larger values of ε are considered [19].

A deep use of the SC and continuation techniques also provides a theoretical scheme of the generation of the spike-adding and chaotic regions in fold/hom bursters, related with canard phenomena [20] and codimension-two homoclinic bifurcations [8], respectively. For instance, Fig. 4b shows, for $\varepsilon = 0.01$, a two-parameter plot (plane (b, I)) of the HR-model with the SC technique. Superimposed to it, the figure displays the main bifurcation curves for the spike-adding process (yellow—fold or saddle-node of periodic orbits (SN), red—period-doubling (PD), black—primary homoclinic bifurcations) and the codimension-two bifurcation points (purple—inclination-flip (IF) and green—orbit-flip (OF)) where the pencils of period-doubling and fold bifurcations are generated. In Fig. 4a we depict the theoretical unfolding ([21] and Appendix A) of the "in" and "out" versions of the codimension-two IF and OF homoclinic bifurcations of type C, describing the pencils of period-doubling and fold bifurcations. Although there are more types of IF and OF bifurcations, in HR model only type C is present. See Appendix A for a more detailed mathematical description of homoclinic bifurcations.

All the above results provide us with some global information of the structure of the parameter space of the HR system, but since in most cases only the first homoclinic bifurcation was used, this information is just partial. In Fig. 5 we display the complete scenario that shows the interlaced bifurcation diagram for the n to $n + 1$ spike-adding process when ε is small. Note that the homoclinic curves contain the codimension-two homoclinic bifurcation points

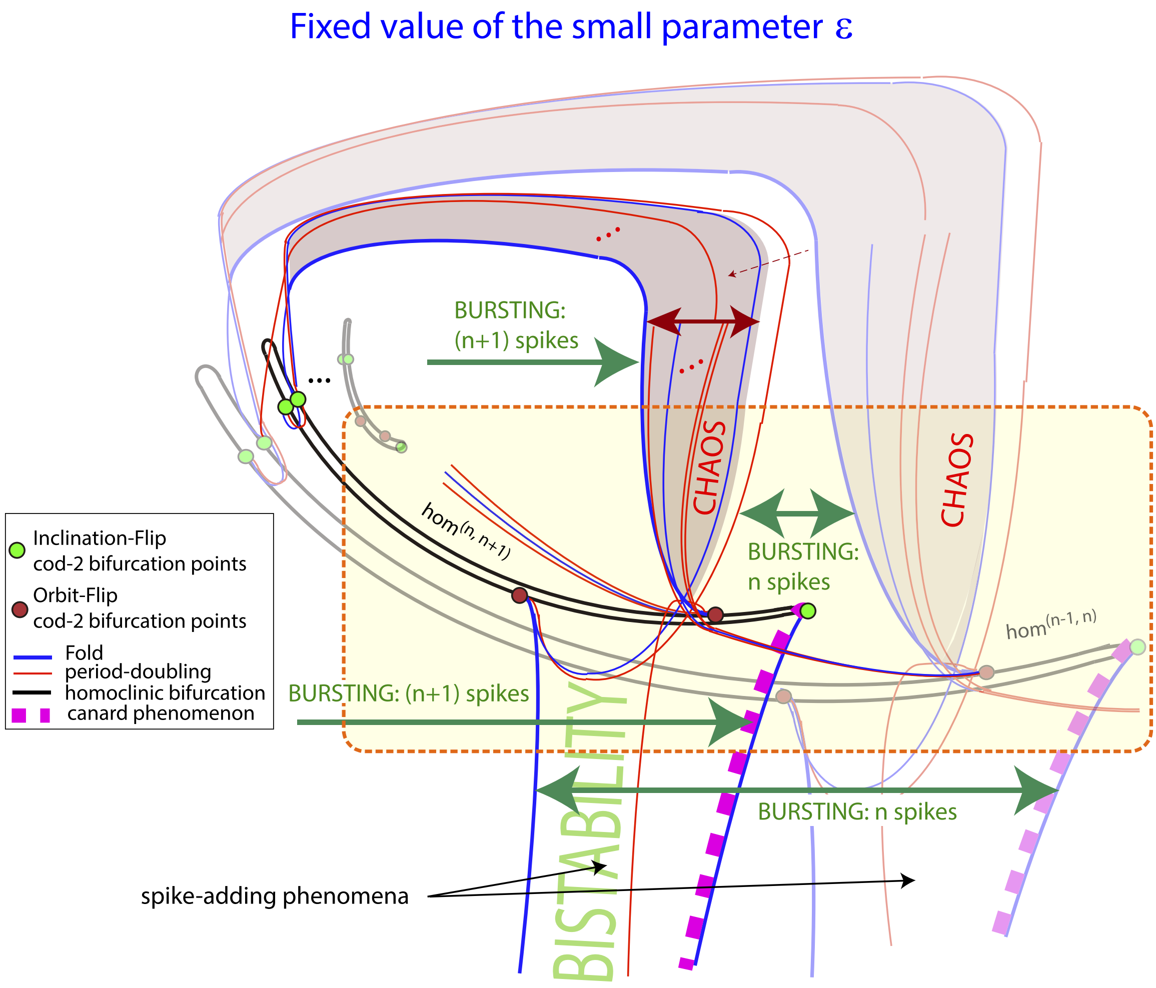


Fig. 5 Taken from [22] with modifications (Figure 6). Generic theoretical scenario (for the case of $n > 2$) that shows the interlaced bifurcation diagram for the n to $n + 1$ spike-adding process when ε is small. The notation $hom^{(n,n+1)}$ is used for the isola of homoclinic bifurcations for which parameters on one side generate homoclinic orbits with n spikes and on the other side with $n + 1$ spikes

that generate the bifurcations leading to the change from n to $n + 1$ spikes, and that they are connected with the other homoclinic curves and with chaotic regions. Moreover, these homoclinic curves are isolas, that is, closed curves in the parameter space. See [22, 23] for a complete description of the theoretical scheme.

The discovery of the relevant role of the homoclinic bifurcations in the fold/hom spike-adding process [8, 12] also allows us to classify the different types of spike-adding processes [14]. See in Fig. 6 how the homoclinic bifurcation acts as a boundary of a continuous case spike-adding involving canard orbits (to the left) and the isola-type one (to the right). The top plot shows the AUTO L_2-norm of the periodic orbits, while the bottom plot displays the Interspike Bifurcation Diagram (IBD), giving the time between two consecutive spikes of the periodic orbits. The bottom plot clearly indicates the number of spikes per orbit, while the top plot indicates the type of spike-adding process (see [14] for more details).

2.2 A global three-parameter analysis: case $\varepsilon \nearrow \mathcal{O}(1)$

The previous subsection showed the behavior of the parameter planes when the small parameter ε was taken in a standard range. In this subsection we focus on studying the global parametric phase space. Just to give an idea of the global parametric panorama, Fig. 7 presents

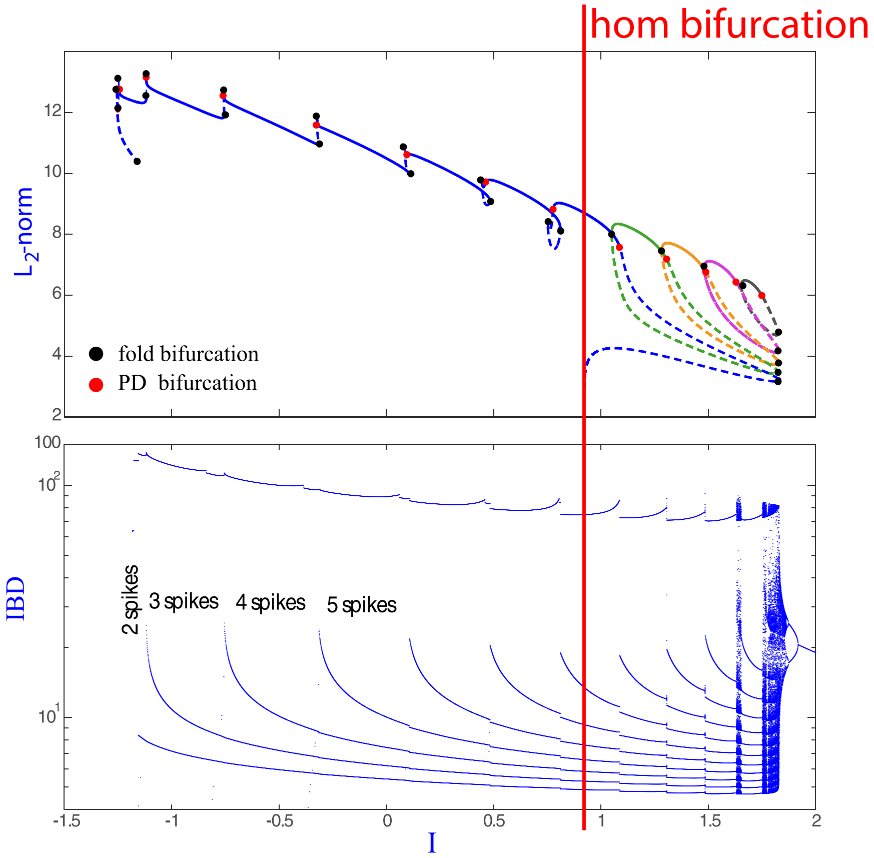


Fig. 6 Taken from [22] with modifications (Figure 11). Top: AUTO L_2-norm of the orbit for $x_0 = -1$, $b = 2.7$ and $\varepsilon = 0.01$ (several fold and period-doubling bifurcations are represented with black and red points, respectively). Bottom: IBD for the same orbit (Color figure online)

a three-parameter plot using the spike-counting technique. As it can be seen from two-parameter plots on the right side of Fig. 7, bifurcation curves of periodic orbits emerge from codimension-two homoclinic bifurcation points, in this case, the HR model exhibits IF, OF and Belyakov bifurcations (see [13] for a complete description). Further details about the bifurcations shown in Fig. 7 will be provided later, but the figure shows us that, by increasing the small parameter ε, some structures change and even disappear. Two-parameter plots show that some color bands and also some codimension-two bifurcation points are present no more when the value of parameter ε is larger than its standard values. This fact leads us to wonder if there are codimension-three bifurcation points or any other mechanism responsible of the disappearance of the codimension-two bifurcation points.

In most fast–slow systems with explicit small parameters, these parameters play a significant role and drastic changes in the global phase space are exhibited under their variation. As commented before, in the HR model, increasing ε leads to numerous changes, and in what follows, we want to show how the HR model exhibits geometric bifurcations [24] with respect to ε that, linked with dynamical bifurcations, explain the observed phenomena. The concept of geometric bifurcations was introduced in [24] to visually classify observed changes in the parameter space when special points "seem" to collide and disappear in points that are not topological bifurcations. These changes are due to the way of observing the dynamics and, although they are not true bifurcations, provide useful insights on the global parametric phase space of the model. Therefore, we add, to the standard codimension of a bifurcation, extra codimensions referred to these geometric bifurcations (see [24] for more details).

We are going to show that codimension–one-plus-one, two-plus-one and one-plus-two geometric bifurcations occur in the HR model. We will see how codimension-one bifurca-

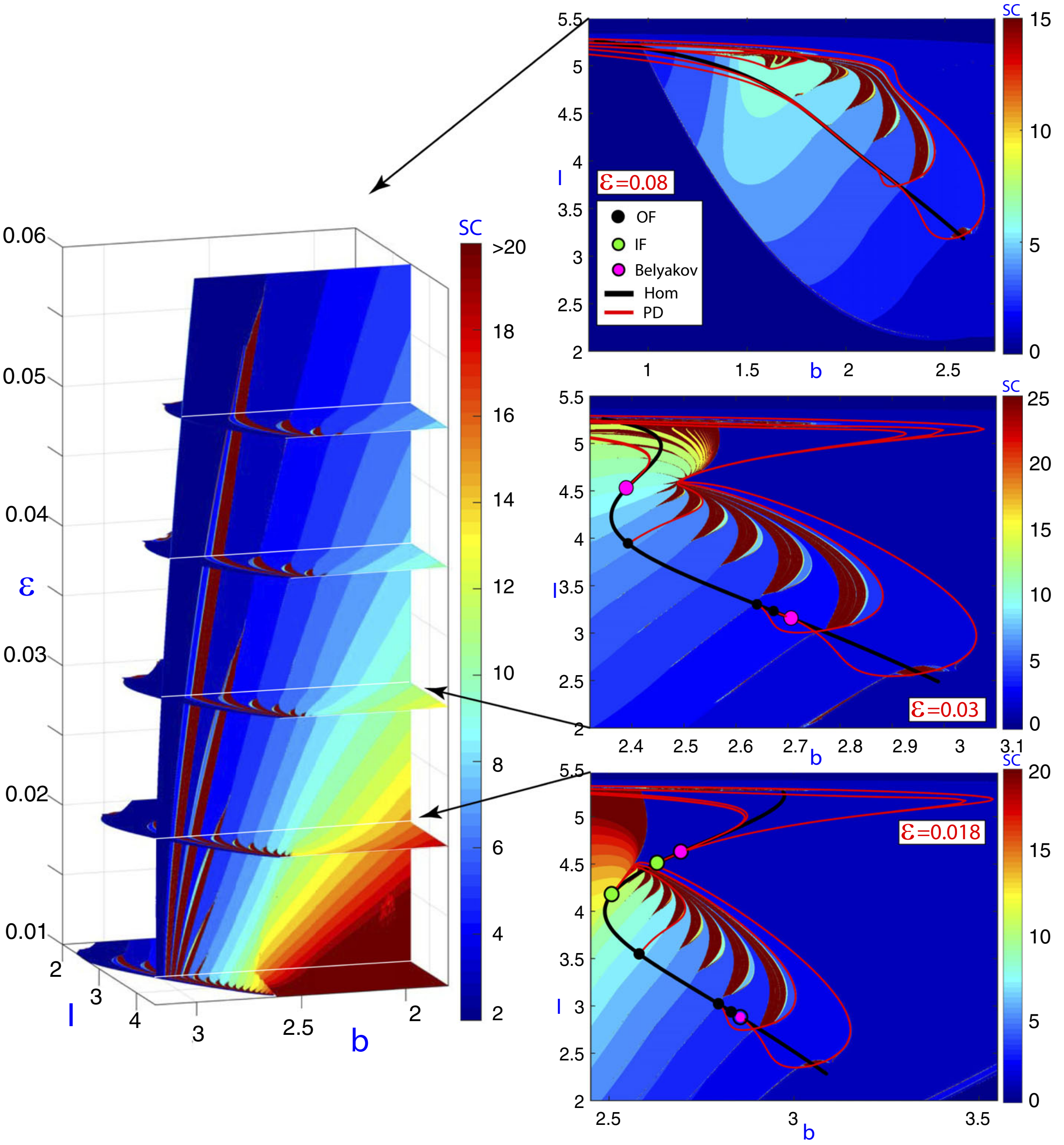


Fig. 7 Left: Three-parameter (b, I, ε) spike-counting diagram of the HR model. Right: Two-parameter plots showing slices for $\varepsilon = 0.018, 0.03$ and 0.08, with bifurcation lines and points: homoclinic and period-doubling codimension-one curves, IF, OF and Belyakov codimension-two bifurcation points

tion surfaces which are unfolded generically from codimension-two bifurcation curves are affected by the existence of codimension–two-plus-one degenerations on these curves.

From the analysis in [13], it follows that in the HR model there exist many codimension-one homoclinic bifurcation surfaces which are exponentially close to each other, and that their number grows to infinity when the small parameter ε tends to zero. The intersection of each surface with horizontal planes produces isolas (closed curves) as these homoclinic surfaces are tubular. Moreover, these isolas exhibit a pair of extremely sharp folds and their width is also exponentially small. Folding points determine two different sides in each isola and also in each bifurcation surface. Typically, the number of spikes of the homoclinic orbit for parameter values on one side of a homoclinic bifurcation isola and on the other side differ in 1. The isola for which parameters on one side generate homoclinic orbits with n spikes and on the other side with $n + 1$ spikes, with $n = 1, 2, \ldots$, will be denoted by $hom^{(n,n+1)}$,

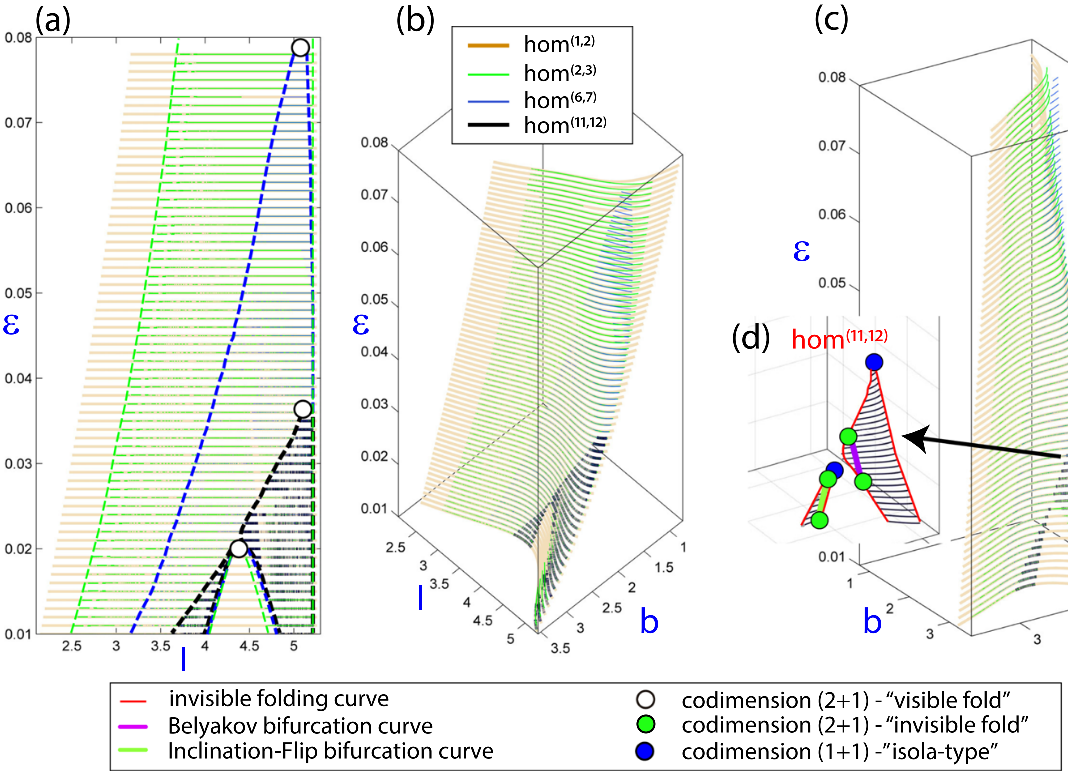


Fig. 8 Three-parameter plot showing codimension-one homoclinic bifurcation surfaces. In plots **a–c** the homoclinic surfaces $hom^{(1,2)}$, $hom^{(2,3)}$, $hom^{(6,7)}$ and $hom^{(11,12)}$ are shown together using three points of view to illustrate how close are one to each other and to compare their respective sizes. In plot **d** it is presented the $hom^{(11,12)}$ homoclinic bifurcation surface with some codimension-two homoclinic bifurcation curves and some geometric bifurcations

and so will be the corresponding surface. In Fig. 8 we show three-parameter plots with some codimension-one homoclinic bifurcation surfaces and codimension-two homoclinic bifurcation curves computed using the AUTO continuation software [17, 18]. In plots (a), (b) and (c) the $hom^{(1,2)}$, $hom^{(2,3)}$, $hom^{(6,7)}$ and $hom^{(11,12)}$ surfaces are given. In plot (d) we include Belyakov and IF bifurcation curves. This partial bifurcation diagram allows us to illustrate a collection of geometric bifurcations, some of which are quite evident, while others not so much. First, we observe the existence of some codimension–two-plus-one geometric bifurcations that correspond to folds, with respect to ε, of curves of codimension-two homoclinic bifurcations. Nevertheless, we will distinguish between "visible" and "invisible" folds (see plot (d)) depending on whether the fold is clearly visible or another kind of analysis is needed to ensure its presence (this happens in the very sharp folds of the homoclinic isolas where it is needed a plot of the bifurcation curve showing the AUTO L_2-norm of the homoclinic orbit as in Fig. 9). Recall that in all 2D-manifolds of codimension-one homoclinic bifurcations we distinguish two leaves which are exponentially close (in fact the two leaves that form the isolas) and therefore they are indistinguishable in the visualization of our numerical results. They glue together along curves of sharp folding marked with a red line in plot (d) of Fig. 8. If a curve of codimension-two homoclinic bifurcation folds inside one of the leaves, that folding is said a *codimension–two-plus-one visible fold*. If the fold is along one of the folding curves of the whole surface (that is, going from one leaf to the other), and hence it is hidden for visualization, we refer to the bifurcation as a *codimension–two-plus-one invisible fold*. In any case, both types of folding are codimension–two-plus-one geometric bifurcations. Note that visible folds appear in pairs, one on each leaf, but invisible folds are unique points. In [24] it is shown that on the surface $hom^{(1,2)}$, both the Belyakov and the IF bifurcation curves

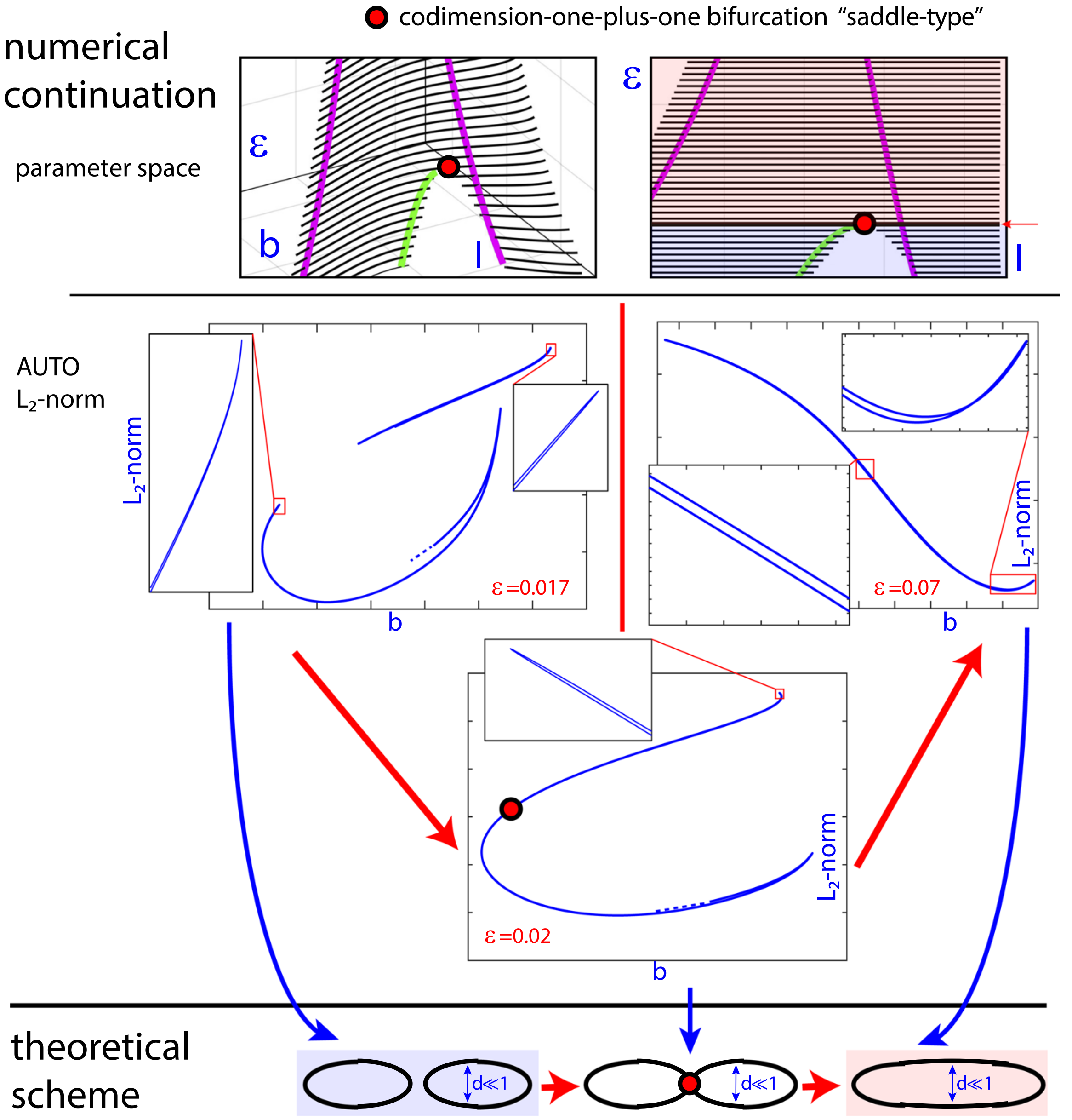


Fig. 9 Saddle-type codimension–one-plus-one geometric bifurcation located on the homoclinic bifurcation surface *hom*$^{(2,3)}$. Bottom line: theoretical scheme showing the connection of two folds of two disconnected isolas giving rise to just one isola. Middle-top lines: numerical continuation illustrating the saddle-type bifurcation in parameter space, showing the connection of the isolas and also a continuation in the AUTO L_2-norm, to show that really we have two or just one isola

exhibit "visible" folds, and that, on the surface *hom*$^{(2,3)}$, the Belyakov bifurcation curve also undergoes a "visible" fold, but the IF curve presents an "invisible" fold. Note that, as shown in Fig. 8, on the surface *hom*$^{(11,12)}$, both curves exhibit "invisible" folds.

In plots (a), (b) and (c) of Fig. 8 the homoclinic surfaces *hom*$^{(1,2)}$, *hom*$^{(2,3)}$, *hom*$^{(6,7)}$ and *hom*$^{(11,12)}$ are shown together using three different points of view to show how close they are to each other and to compare their respective sizes. The *hom*$^{(1,2)}$ homoclinic surface is the biggest one and so over this surface mainly "visible" folds are observed. On the contrary, over the other surfaces, as the size of the homoclinic surface is smaller, and some bifurcations are located in a small region, there are some "invisible" folds because the codimension-one curves reach the fold of the homoclinic isola and go to the other side of the homoclinic surface via an "invisible" fold.

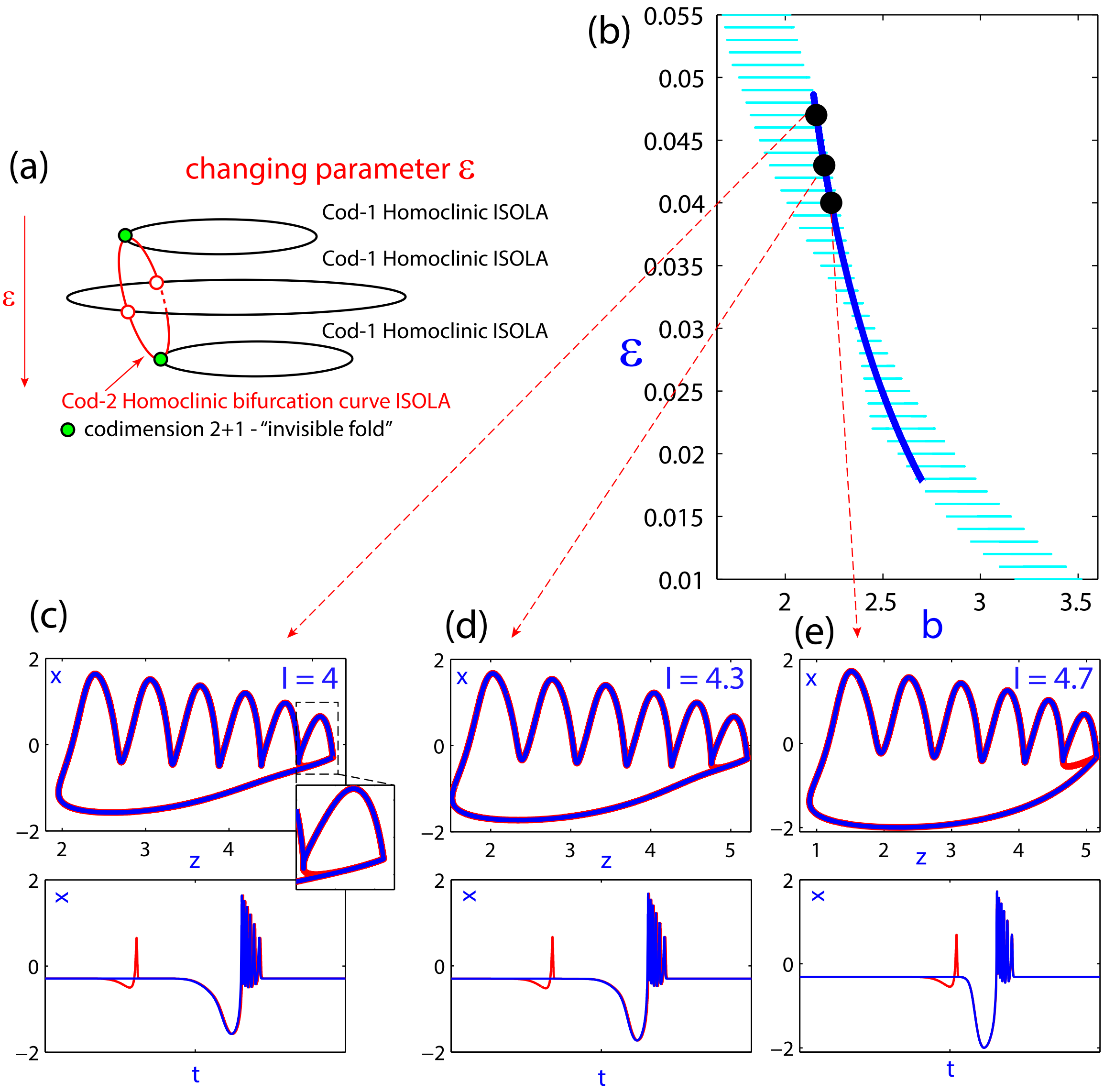


Fig. 10 "Invisible" fold phenomenon giving rise to a codimension-two Belyakov homoclinic bifurcation isola on the surface $hom^{(6,7)}$. **a** Theoretical scheme of the process. **b** Projection of the homoclinic surface on the (b, ε) parameter plane and location of the Belyakov isola. **c–e** Homoclinic orbits on the Belyakov isola on both sides of the curve for the indicated values of I. One side has one extra spike as shown both on the time series (t, x) and (z, x) pictures for three values of the parameter b

Now, looking for codimension–one-plus-one geometric bifurcations, we pay attention to the bifurcation surfaces themselves. In Fig. 8d we see how $hom^{(11,12)}$ splits into two disconnected components and each of them has a maximum (with respect to ε). These two points are isola-type codimension–one-plus-one geometric bifurcations with respect to ε. Also, a saddle-type codimension–one-plus-one geometric bifurcation is detected on the bifurcation surfaces $hom^{(2,3)}$ and $hom^{(6,7)}$ (see Fig. 9). Saddle-type codimension–one-plus-one bifurcations do not appear on $hom^{(1,2)}$ for the studied values of ε, but they are present in $hom^{(n,n+1)}$, when $1 < n < n_s$, for certain small n_s. Nevertheless, for $n \geq n_s$, the surface $hom^{(n,n+1)}$ splits into two pieces (see Fig. 8d) and hence saddle-type codimension–one-plus-one bifurcations are no longer present. Most likely, isola-type codimension–one-plus-one geometric bifurcations are present in all homoclinic surfaces if the small parameter ε is large enough. This fact explains why, as the small parameter grows, fewer color bands appear in the 3D graph of Fig. 7 indicating that the bursting orbits have fewer spikes. Note that this structure provides a complete explanation of the maximum number of spikes in models once the parameter ε is no longer small.

In order to show more clearly a saddle-type codimension–one-plus-one bifurcation we show in Fig. 9 the theoretical scheme of the connection of two folds of two disconnected isolas giving rise to an isola, and therefore giving rise to just one tubular surface from two previous tubular surfaces. On the upper plots, several numerical continuation results permit to illustrate the saddle-type bifurcation in parameter space showing the connection of the isolas. AUTO L_2-norm is used to show that really we have isolas of homoclinic bifurcations. Note that the numerical software AUTO stops the continuation process at the dotted areas of the blue curves because of numerical limitations. The magnifications illustrate the very sharp folds of the isolas, only visible with zooms on the AUTO L_2-norm plots up to some parameter values due to the numerical precision of the program.

The phenomenon of the "invisible" folds is a direct consequence of the existence of very squashed tubular structures (and therefore isolas when a two-parameter section is considered) where the two leaves are infinitesimally close to each other. Thus, if a codimension-two homoclinic bifurcation curve reaches the homoclinic surface folding curve, then it continues to the other side because the conditions in the phase space that are required to have the codimension-two bifurcation are still satisfied on the other side. So, the bifurcation curve has experimented an "invisible" fold. Figure 10 details a complete process of creation of "invisible" folds on a codimension-two Belyakov homoclinic bifurcation curve on the surface $hom^{(6,7)}$. The plot (a) shows the theoretical scheme of the process and illustrates how one side of the Belyakov isola is on one leaf of the tubular surface and the other side on the other leaf, and so we have "invisible" folds due to the very small distance between the leaves. On the plot (b) the projection of the homoclinic surface on the $(b,\ \varepsilon)$ parameter plane and the location of the Belyakov isola are shown. In order to illustrate that indeed the Belyakov curve forms an isola on both leaves of the homoclinic surface, we present on plots (c), (d) and (e) the time series (t, x) and (z, x) pictures of two homoclinic orbits, one on each side of the surface, for three values of parameter b: $b = 2.1597$, 2.1999 and 2.2478, respectively. Note that on one side the homoclinic orbit has one extra spike (7 spikes versus 6 spikes).

Figure 7 shows how some codimension-two points disappear. Now we have enough elements to describe the process and also to explain that a subsidiary effect is the fact that different curves of periodic bifurcations may connect together. We can connect all these phenomena with some geometric bifurcations illustrated in Fig. 8, in particular "visible" or "invisible" folds depending on the size of the homoclinic isolas. As ε increases, the first geometric bifurcation shown in Fig. 8 that explains what happens in Fig. 7 is a codimension–two-plus-one bifurcation on the IF homoclinic bifurcation curve. For the $hom^{(1,2)}$ case, as the homoclinic surface is big enough, what happens is a maximum of each of the IF curves (one on each leaf of the tubular surface), and so we have a "visible" fold and we observe the geometric "collision" of two pairs of IF points, one pair on each of the leaves of the homoclinic surface (the white points on the figure). On the contrary, for the rest of homoclinic surfaces, the IF bifurcation curves have "invisible" folds, as they are smaller and the surfaces are composed of isolas disconnected for small values of the parameter. Moreover, when n grows enough $hom^{(n,n+1)}$ has only one branch of the IF curve, that is, IF points are only present on one of the two disconnected components of the isolas for a small value of ε. In these cases, the IF point seems to disappear on the limit of a homoclinic curve, and what really happens is that geometrically "collides" with the corresponding point of the another leaf of the surface that we cannot see. The evolution of the Belyakov curve is similar but for higher values of the small parameter ε.

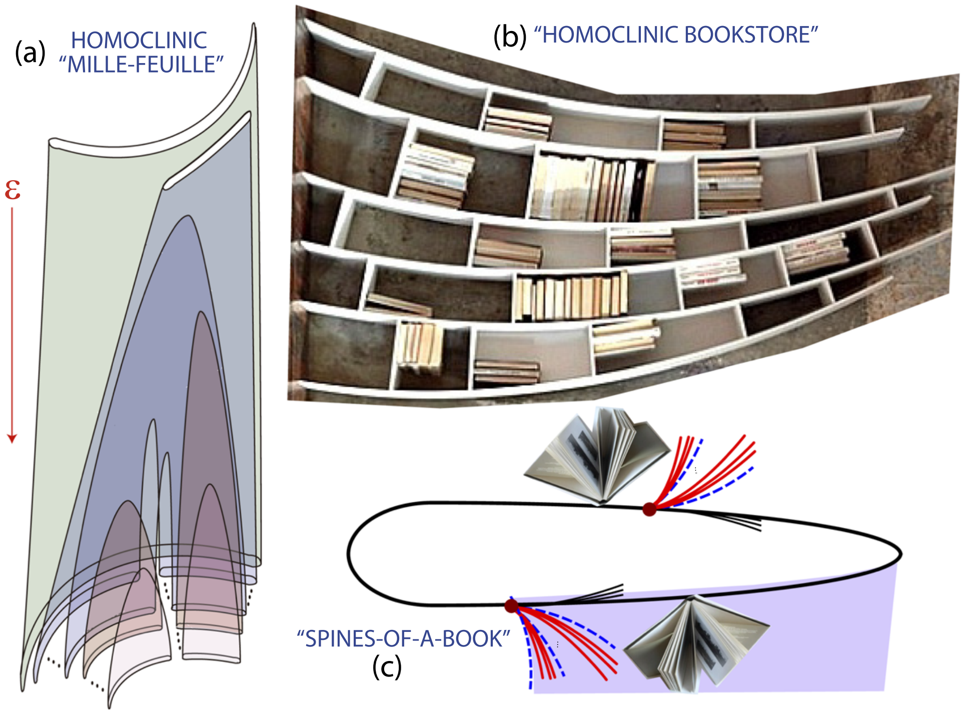


Fig. 11 Left: Fold/hom homoclinic "mille-feuille" organization. Right: Simile of the structure on the left. The behavior of the curves of codimension-two bifurcations is compared to the spine-of-a-book placed on a bookstore that corresponds with the homoclinic surface. The pages of such books consist of surfaces of bifurcations of periodic orbits and secondary homoclinic bifurcations involved in the spike-adding process

Finally, Fig. 11a shows the schematic distribution of the homoclinic bifurcation surfaces giving rise to what we have called fold/hom homoclinic "mille-feuille" organization [13]. On plot (b) we show a simile comparing the structure with a bookstore, where each homoclinic surface is compared with a bookstore shelf, and where each book will take the role of the codimension-two homoclinic bifurcations on the homoclinic surface. They will give rise to the spine-of-a-book structure (plot (c)) with the countable set of bifurcations emanating from the codimension-two points (the "sheets of the book"). Note that this structure provides a complete explanation of the global homoclinic bifurcation in the HR model and in other fold/hom bursters.

2.3 Topological structure of chaotic attractors: limit case $\varepsilon \searrow 0$

In addition to studying the different bifurcations that periodic orbits undergo, we can analyze the topological structure of chaotic attractors and their evolution when the parameter values vary. One of the fundamental tools in such analysis for dissipative three-dimensional dynamical systems are topological templates, which are constructed as a version of the system in the limit case of infinite dissipation. That is, a projection of its flow is made by squeezing it along its stable direction, so that the three-dimensional flow becomes a two-dimensional branched manifold (see Fig. 12). The projection preserves the relative positions of the periodic orbits, so that the crossings between them are neither created nor destroyed in the projection. The template is therefore a useful tool for analyzing the stretching and squeezing processes that shape the chaotic attractor and how the unstable periodic orbits (UPOs) are organized in these processes. To determine the topological template, the intertwining of the low-period periodic orbits must be studied through their topological periods (number of spikes) and linking numbers (number of turns that one orbit makes around another). With that information,

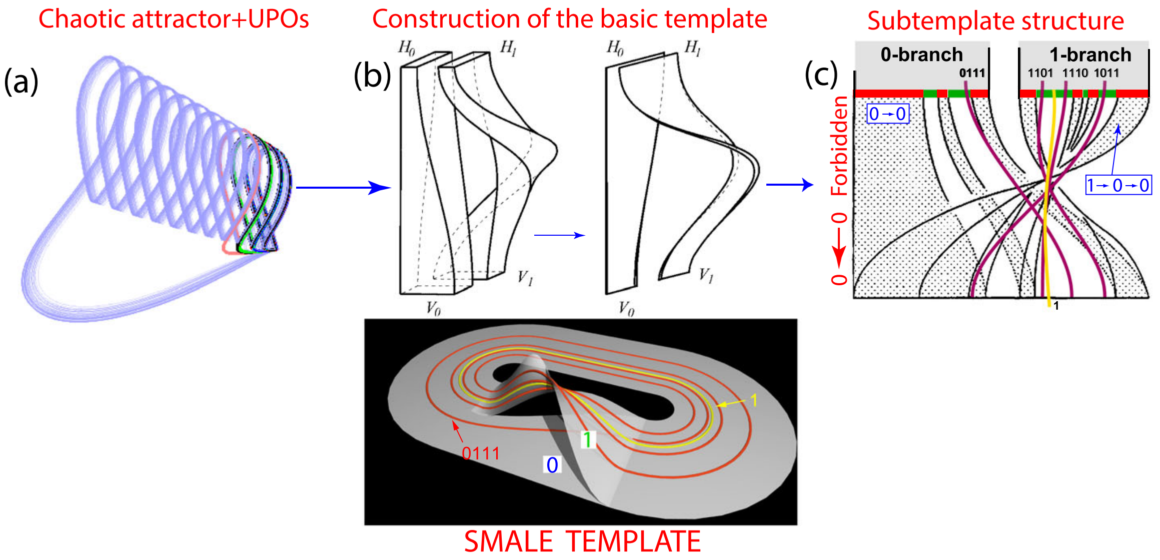


Fig. 12 Top: Basic elements of topological template analysis. Bottom: Global topological template obtained for HR model, the complete Smale horseshoe template. See [25] for the analysis of the templates and the description of allowed and forbidden symbolic sequences associated to periodic orbits

the corresponding topological template will be the simplest one that has the characteristics detected in the UPOs.

When the chaotic attractor is hyperbolic, there is a biunivocal correspondence between the periodic orbits of the template and those which foliate the attractor. In the case of the HR system, the attractor is not hyperbolic, which implies that some orbits in the template do not appear in the attractor. However, those that do exist in the attractor have the same internal organization as in the template. This results in the concrete template of the attractor being a subtemplate of the global template in which certain paths of the template are closed (those corresponding to the orbits that are not in the attractor). This gives a Cantor structure in the subtemplate depending on the closed paths and corresponding to the structure of the chaotic attractor, where the "holes" correspond to the periodic orbits laying on those close paths. See [25] for a full analysis of these templates, making use of symbolic dynamics, topological templates and first return maps (FRMs) [26, 27]. In that study, it was found that the topological template of each chaotic attractor corresponds to a subtemplate of the complete Smale horseshoe template where there are paths that are closed, and how they relate with certain forbidden symbolic sequences (Fig. 12).

Following a line crossing different chaotic regions in the parameter space, we can study how the chaotic attractor changes in the different regions. To do this, we can take, for example, $\varepsilon = 0.01$ and $I = (100 - 26.5b)/6.91$ (see Fig. 4). The upper part of Fig. 13 shows how the shape of these attractors changes along that line. Now, using the symbolic dynamics theory, we get that those chaotic invariants with more foliated periodic orbits have more symbolic chains allowed in their topological template than those with a smaller number of them. That is, depending on the number of periodic orbits that foliate (or not) the chaotic attractor, its generic topological template will have certain open (or closed) paths forming a subtemplate with holes.

According to the classification of the different types of spike-adding processes provided in [14], we have a chaos-induced discontinuous spike-adding (Terman [11]) case. This case is generated by a type C OF homoclinic bifurcation (see Sect. 2.1). This codimension-two point generates a countable set of saddle-node and period-doubling bifurcation pencils. Moreover, in [25] it was seen that a symbolic-flip bifurcation occurs before each period-doubling bifurcation. This bifurcation does not change the stability of the periodic orbit or any of its

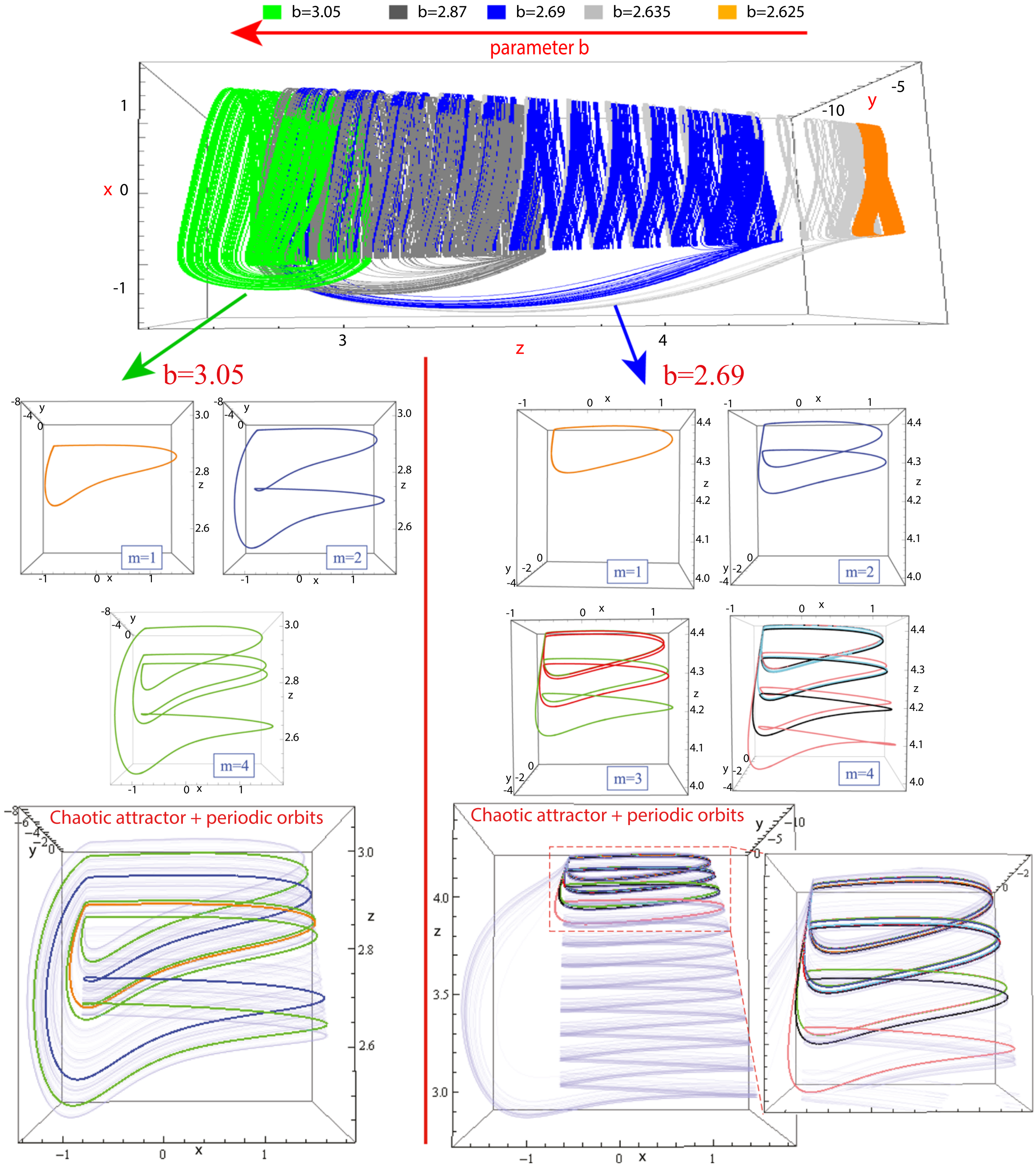


Fig. 13 Top: Chaotic attractors for different values of b in the line $I = (100 - 26.5b)/6.91$ (with $\varepsilon = 0.01$). Middle-bottom: Two of these chaotic attractors with all their UPOs with multiplicity m from 1 to 4. We can check how, when parameter b decreases, the number of orbits foliated to the attractor increases

topological properties, but it does modify the symbolic sequence of the periodic orbit and it allows generating new symbolic chains. That is why, as more spike-adding pencils are traversed, more symbolic chains are allowed. Figure 14 plots the number of spikes reached by the periodic attractor as a function of the values of b and ε. The number of spikes grows exponentially as ε decreases to zero.

Thus, we can elaborate the hypothesis represented schematically in Fig. 15. The template of any chaotic attractor in the system is a subtemplate of the complete Smale template with closed paths corresponding to the forbidden symbolic chains in the symbolic dynamics of that chaotic attractor. However, when ε tends to zero, choosing the right path to go through all the different spike-adding pencils as they occur indefinitely, the subtemplate is being filled in and the Smale horseshoe template is completed as the limit case.

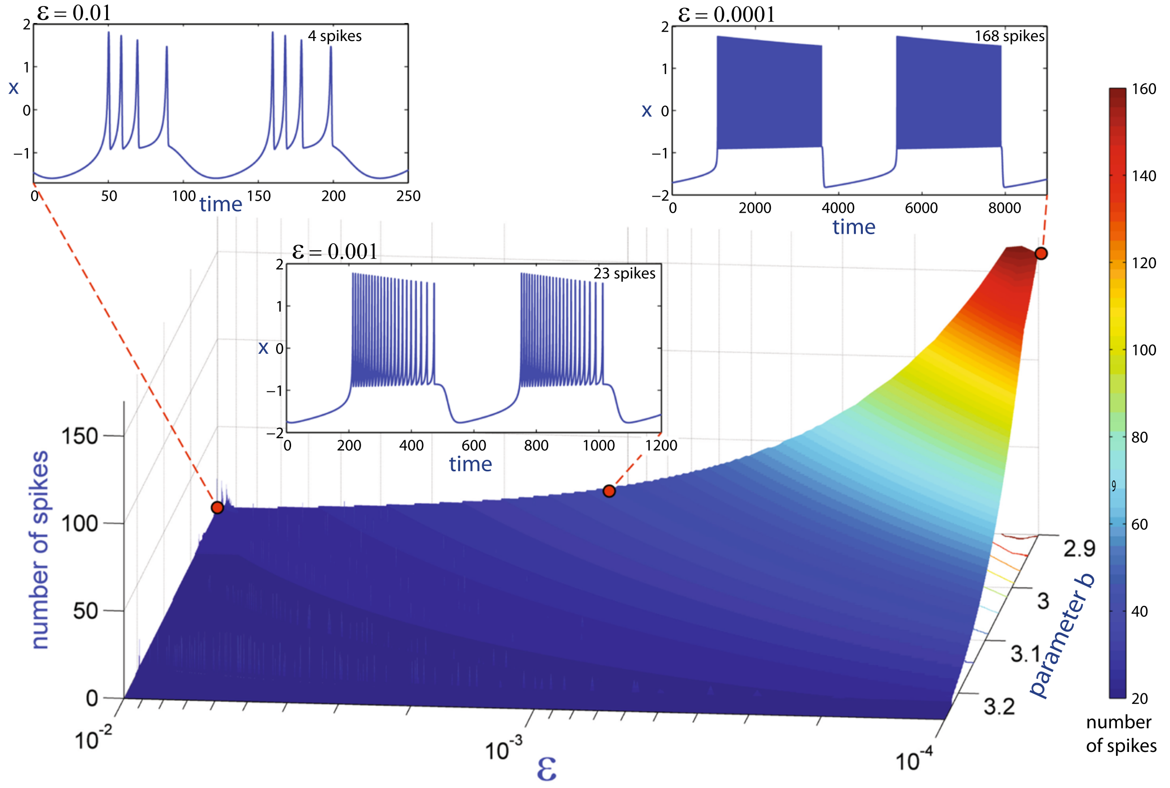


Fig. 14 Spike-counting in the bursting attractor along the line $I = 14.4 - 4b$ for $\varepsilon \in [10^{-4}, 10^{-2}]$ (in logarithmic scale to show exponential increment in the number of spikes) and $b \in [2.9, 3.25]$. The time series for three particular examples are shown too

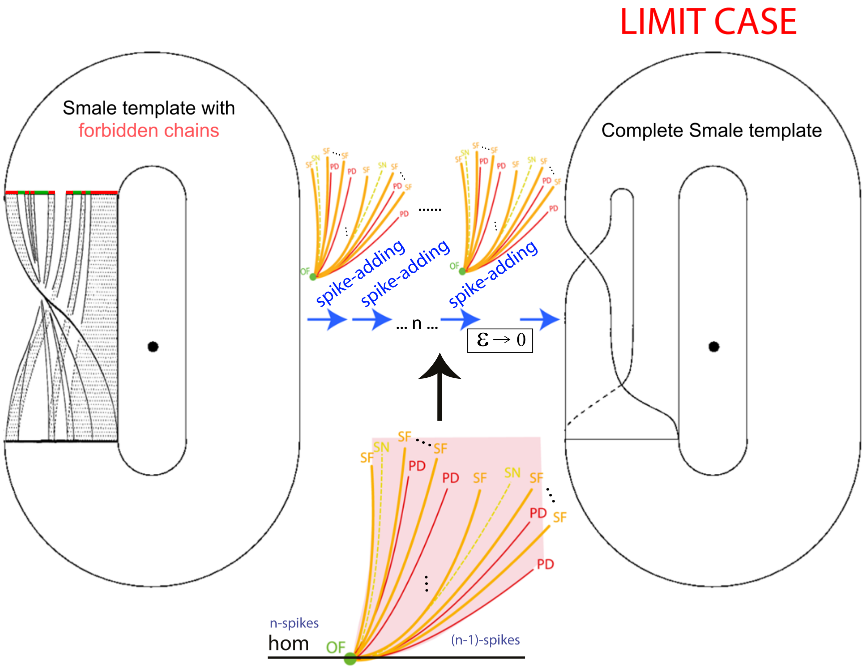


Fig. 15 Theoretical scheme showing how, through infinite spike-adding pencils, the topological template of the chaotic attractors fills more and more gaps and tends (when ε tends to zero) to the complete Smale template

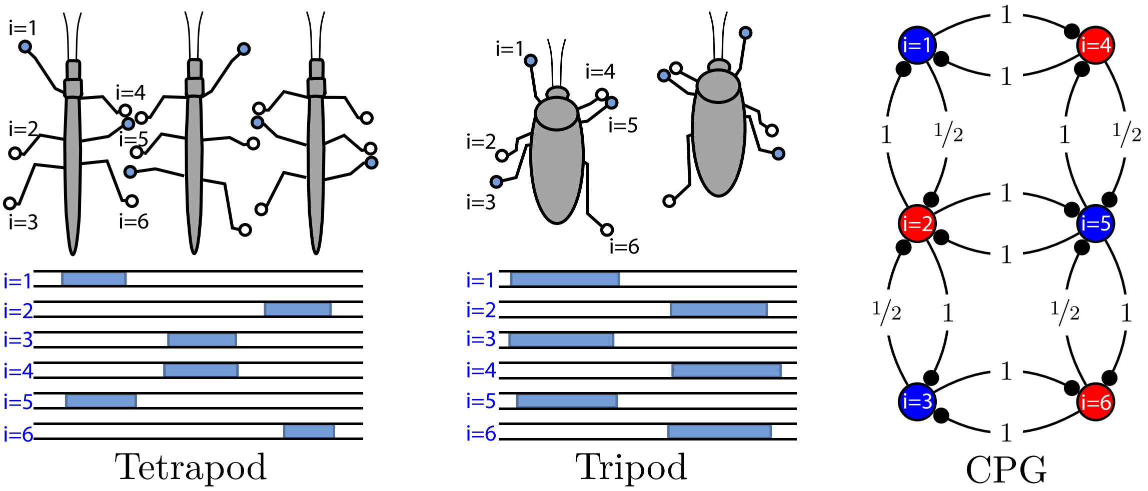


Fig. 16 Left and center: Illustration of the tetrapod and tripod insect movement gaits. Each leg is associated to a neuron, whose activation (bursting) sets the leg into motion. Bottom figures show the temporal bursting pattern, from left to right, of each neuron, with shaded regions indicating the activation of the corresponding neuron. Right: Network of neurons used as a model for hexapods, with the corresponding connections. Each connection from neuron j to neuron k has an associated non-symmetric relative inhibitory strength $c_{j,k}$, with the indicated values: $c_{1,2} = \frac{1}{2}$, $c_{2,1} = 1$, and so on

3 Realistic models applications

In this section we illustrate how spike-adding phenomena are involved in several real processes, such as changes in movement gaits and the creation of Early Afterdepolarizations in cardiac dynamics.

3.1 Insect gait movement changes

The generation of rhythmic or coordinated behaviors in different organisms, such as heartbeat, respiration, swimming or walking, is an important research topic in neuroscience with applications to other fields like biological-inspired robotics.

Many animals, including humans, present Central Pattern Generators (CPGs) to produce the basics of such rhythms (see [28] and the references therein). A CPG is a small biological neuron circuit, a small network of interconnected neurons, that produces a rhythmic output without needing a rhythmic input, while also adapting it when an input is present. Hence, the organism can modify the rhythms according to the environment as needed. Its dynamics depends on intracellular, synaptic, and network level phenomena.

Classic examples of CPGs driven rhythmic behavior are direct-reverse flow of circulatory system on leeches [29, 30] and locomotive patterns [31–33]. Of special interest, on both insect movement and robotics research, is the case of the movement of hexapods. In Nature, hexapod insects can present different gaits, most common (and idealized) ones are shown in Fig. 16. In the tetrapod gait there are four legs in the ground and the other two are moving, while in tripod gait, usually present when the insect wants to go faster, three legs are moving while the other three are at rest.

Researchers on both mentioned areas have proposed simplified CPGs to describe those movements, usually including six motoneurons [34–40]. There are more complex models which include biomechanical information [41–43], but those simplified CPGs are rich enough to represent the usual patterns and exhibit some of the changes one may observe in the movement of real insects.

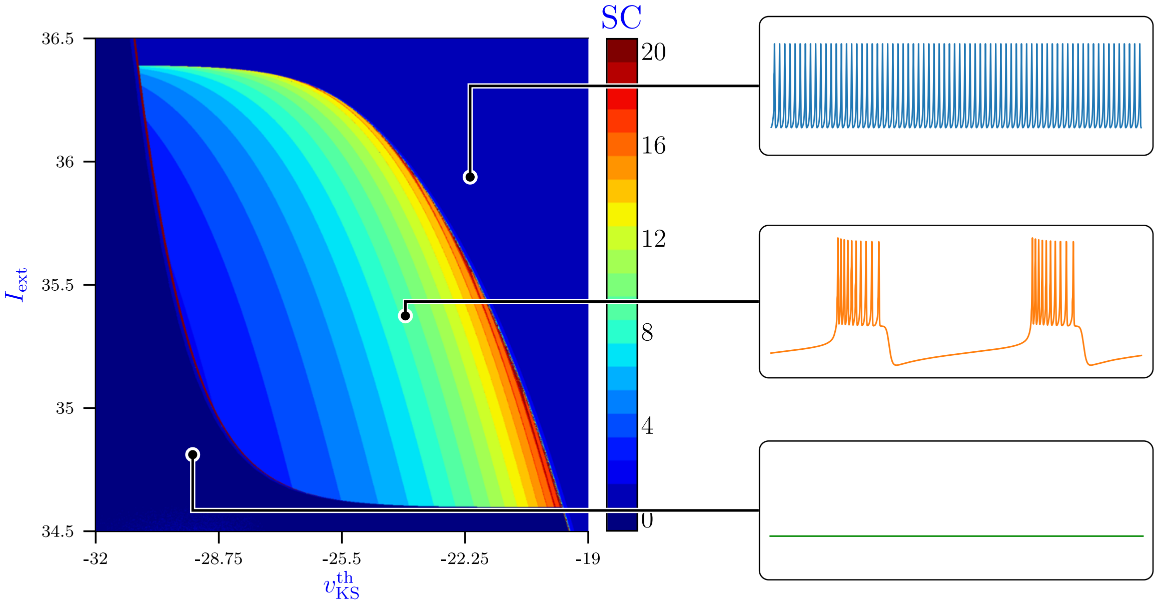


Fig. 17 The spike-counting diagram for an isolated neuron varying the parameters I_{ext} and v_{KS}^{th} of (2). On the right there are three selected membrane potential time series showing the usual behavior in the three different regions [37, 38, 44]

Ghigliazza and Holmes [37, 38] introduced one of those simple models for the movement of cockroaches, and examined it analytically using some reductions, which unfortunately hide some non-symmetric patterns that may occur. In this subsection we want to explore its dynamics numerically without resorting to such reductions.

The dynamics of one isolated neuron is described in terms of a fast nonlinear calcium current I_{Ca}, a slow potassium current I_K, a very slow potassium current I_{KS} [37], a linear leakage current I_L and an external current I_{ext}. The ODEs describing the dynamics of the membrane potential v, the potassium gate variable m, and the slow potassium gating variable w are

$$\begin{cases} C\dot{v} = -(I_{Ca} + I_K + I_{KS} + I_L) + I_{ext}, \\ \dot{m} = \dfrac{\epsilon}{\tau_m(v)}[m_\infty(v) - m], \\ \dot{w} = \dfrac{\delta}{\tau_w(v)}[w_\infty(v) - w], \end{cases} \tag{2}$$

with the auxiliary ionic current functions defined by

$$\begin{aligned} I_{Ca} &= g_{Ca}\, n_\infty(v)\,(v - E_{Ca}), \qquad & I_K &= g_K\, m\,(v - E_K), \\ I_L &= g_L\,(v - E_L), & I_{KS} &= g_{KS}\, w\,(v - E_K), \end{aligned}$$

and where the different time scales and steady state gating variables are

$$\begin{aligned} \tau_m(v) &= \operatorname{sech}\big(k_{0_K}\,(v - v_K^{th})/2\big), \qquad & \tau_w(v) &= \operatorname{sech}\big(k_{0_{KS}}\,(v - v_{KS}^{th})/2\big), \\ m_\infty(v) &= \Big(1 + e^{-2k_{0_K}\,(v - v_K^{th})}\Big)^{-1}, & w_\infty(v) &= \Big(1 + e^{-2k_{0_{KS}}\,(v - v_{KS}^{th})}\Big)^{-1}, \\ n_\infty(v) &= \Big(1 + e^{-2k_{0_{Ca}}\,(v - v_{Ca}^{th})}\Big)^{-1}. \end{aligned}$$

The parameters C, ϵ and δ determine the time scales of v, m and w, E_X are the Nernst potentials, g_X are maximal conductances, k_{0_X} is the steepness of the transition happening at threshold potential v_X^{th}, where X denotes each of the considered ions. See [37, 38, 44] for the parameters and the exact values used in the simulations.

Figure 17 shows the behavior of the membrane potential for the stable periodic orbit when varying parameters v_{KS}^{th} and I_{ext}. Within the colorful droplet-shaped region we can observe the characteristic neuronal bursting of the model. Outside of it we have two opposite behaviors: quiescence (an attracting equilibrium) and spiking (a small periodic orbit with one spike). The global picture is similar for any other set of parameters [44].

When the six neurons (each driving the movement of a leg of the hexapod) are coupled, they are arranged in a network shown in the right part of Fig. 16. We assume that the inhibitory coupling is achieved via synapses that produce negative postsynaptic currents (we obtain this by subtraction of positive postsynaptic currents instead of adding negative terms as in the original model). Therefore, the first equation of (2) is modified in each neuron i from 1 to 6 to include the postsynaptic current $I_{s,i}$:

$$C\dot{v}_i = -(I_{Ca,i} + I_{K,i} + I_{L,i} + I_{KS,i}) + I_{ext} - I_{s,i}, \;\; I_{s,i} = g_s(v_i - E_s^{post}) \sum_{j \in N_i} c_{j,i} s_j,$$

with N_i the set of neighbors of neuron i as indicated in Fig. 16, $c_{j,i}$ the network parameters shown in the same picture, and a new synapse variable s_i associated to each neuron, whose dynamics is described by

$$\dot{s}_i = \alpha s_\infty(v_i)(1 - s_i) - \beta s_i, \quad s_\infty(v_i) = \frac{T_{max}}{1 + e^{-k_s(v_i - E_s^{pre})}},$$

where α and β are voltage-independent forward and backward rate constants, and g_s, T_{max}, E_s^{pre} and E_s^{post} are parameters describing the synapses (see [44] for details).

Having a roadmap (Fig. 17) for the dynamics of a single neuron allows us to explore the connected case, focusing on the relevant regions of the parameter space. To exemplify this, let us consider the straight line $I_{ext} = 35.5$ in the plane depicted in Fig. 17. We perform a *quasi-Monte Carlo sweeping*, that is, we sweep for values of v_{KS}^{th} between -30 to -22.5 and we compute for each parameter set the orbits for 200 initial conditions selected using a low discrepancy sequence to cover nicely the region of reasonable initial conditions (remember that the phase space is 24-dimensional). For each of these orbits we compute the periodic orbit towards which it converges after a large transient of 10^5 ms and automatically deduce the patterns (as in Fig. 16) it describes. We are not interested in the number of spikes in this case, but only in knowing if a neuron is active (bursting) or not. Figure 18 shows the results of this numerical experiment. At the bottom of the figure, we can see the main obtained patterns.

The histogram represents the perceptual amount of initial conditions that converges to a particular pattern. Patterns related by symmetry (interchanging either left and right or front and back neurons) are considered identical. Below the histogram, we have the spike-counting diagram on the line $I_{ext} = 35.5$. We can observe that the spike-adding areas affect the smoothness of the transition among patterns, which helps us to locate the regions where most changes are performed. Those changes are related with bifurcations on the system that are explained in [45].

The main observation about our computations is the fact that the tripod gait is ubiquitous: it is a possible movement pattern for any choice of the parameters, although it is unstable for some parameter values. However, when it becomes stable, for $v_{KS}^{th} > -25$, it quickly becomes dominant. This result is general, in the sense that we can observe the same dominance for other sets of parameters. Those results are quite stable under small perturbations [45]. An open problem regarding the different existing gaits and their stability is the interesting question of the dynamic change between them when parameters are varied. While animals are able to change from one gait to another one without tripping, it yet remains to be solved how these smooth transitions can be incorporated in the model.

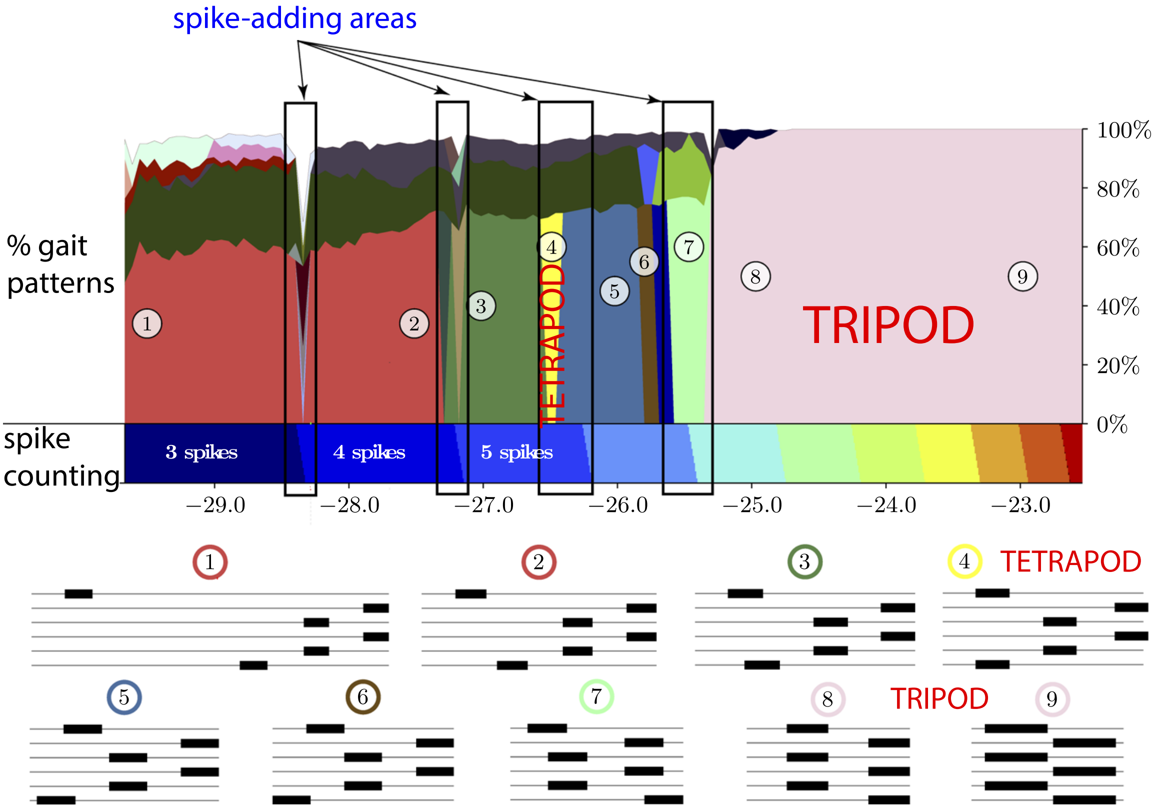


Fig. 18 The histogram with the percentage of the different gaits corresponding to the parametric line $I_{ext} = 35.5$ for variable v_{KS}^{th} using 200 initial conditions for each parametric set. The main patterns are depicted at the bottom. The thick black lines describe the bursting time of each neuron along the periodic orbit; for each pattern, time goes from left to right, and neurons $i = 1$ to $i = 6$ from top to bottom. See [44] for details

3.2 Early afterdepolarizations in cardiac dynamics

In previous sections we have centered our study in neurons, either isolated or connected. Now we are going to study other excitable cells, the myocytes (or muscle cells), in particular, heart myocytes. We want to show how the spike-adding phenomenon is relevant in cardiac dynamics, but now with the name of Early Afterdepolarization (EAD).

Although the Hodkin–Huxley model [1] was initially proposed to simulate the behavior of a neuron, this model established a useful mathematical formalism to describe the ionic kinetics of the membrane which has been used as a basis for the development of mathematical models for other excitable cells. In this way, Denis Noble [46] published the first mathematical model of a cardiac cell based on the adaptation of the equations of the Hodkin–Huxley model. After some improvement by McAllister et al. [47], Beeler and Reuter [48] developed the first ventricular myocyte model in 1977, later reformulated by Luo and Rudy in 1991 [49] and 1994 [50, 51].

In the left plot of Fig. 19a we can see an example of an AP for a cardiomyocyte. It has several phases. The first one, after being stimulated, is an increment in their transmembrane voltage (depolarization or phase 0). This increment is followed by a small partial voltage decrease (transient repolarization or phase 1) and a prolonged phase where voltage remains approximately constant (plateau phase or phase 2). In the final part of the AP, transmembrane voltage decreases (repolarization or phase 3) while returning to the resting potential level, which is maintained until receiving the next stimulus. The time elapsed between two stimuli applied to the cell is called Pacing Cycle Length (PCL). Under some circumstances, transmembrane potential can experience an unexpected rise during AP phase 2 or phase

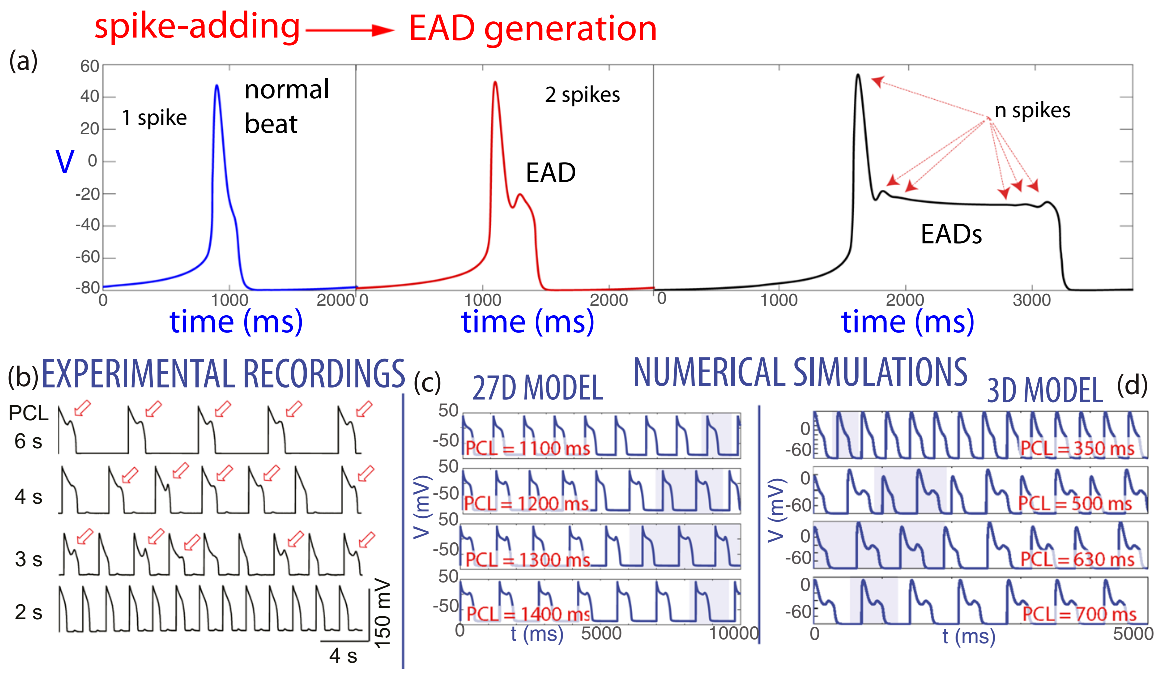


Fig. 19 In the top panel (**a**) we can see different APs without (left) with one (middle) and with several (right) EADs depending on the PCL value (taken from Fig. 1 of [52]). In lower panel (**b**) we show some experimental recordings where for different values of the PCL we detect some APs with one EAD (taken from Fig. 1A of [53]). This is also observed in our numerical simulation (taken from Fig. 1c and d of [54]) for a biophysical realistic model (**c**) and for a simpler one (**d**). In all cases, as we increase the PCL, more and more APs show an EAD

3, which is termed Early Afterdepolarization (EAD). If EADs at cellular level are of large enough magnitude and occur over a substantial tissue area, they can lead to triggered activity and arrhythmias [55] which makes the study of EADs highly relevant. Recall that, in the mathematical neuroscience notation, the creation of EADs is denoted as a spike-adding process.

For our study we have used two mathematical models, the biophysically realistic Sato model [53] (with 27 variables) and a simpler one (the 3D Luo–Rudy model [49, 56] with just three variables). As we can see in Fig. 19c, d both models show the EADs observed in experimental recordings (as illustrated in Fig. 19b).

The high-dimensional model, updated by Sato et al. [53] to properly reproduce EADs, simulates a rabbit ventricular myocyte. The total ionic current is the sum of nine ionic currents including the L-type Ca^{2+} current, the fast sodium current (I_{Na}), five components of the potassium current and two pump currents (I_{NaK} transporting potassium into the cell in exchange for sodium out of the cell and I_{NaCa} transporting sodium in and calcium out of the cell). An additional stimulus current I_{stim} is included, which in this study is delivered at a constant stimulus period defined by the PCL. The complete description of the ionic currents involves 27 ordinary differential equations for the 27 state variables, in which 177 model parameters are used (see [53, 57, 58] for full details and equations).

In Fig. 20 we show some results obtained for the Sato model [59]. In Fig. 20a we show the biparametric bifurcation diagram taking into account the PCL and the K_{mNao} parameter which is the extracellular sodium dissociation constant used to calculate the I_{NaCa} current and, in turn, update sodium and calcium concentrations. For each configuration we integrate until we find the periodic orbit and we count the number of peaks and the number of APs. The ratio between both quantities indicates the progressive appearance of EADs: a ratio of 1 indicates that no EADs are present, while a ratio of 2 shows that every AP has an EAD; a ratio

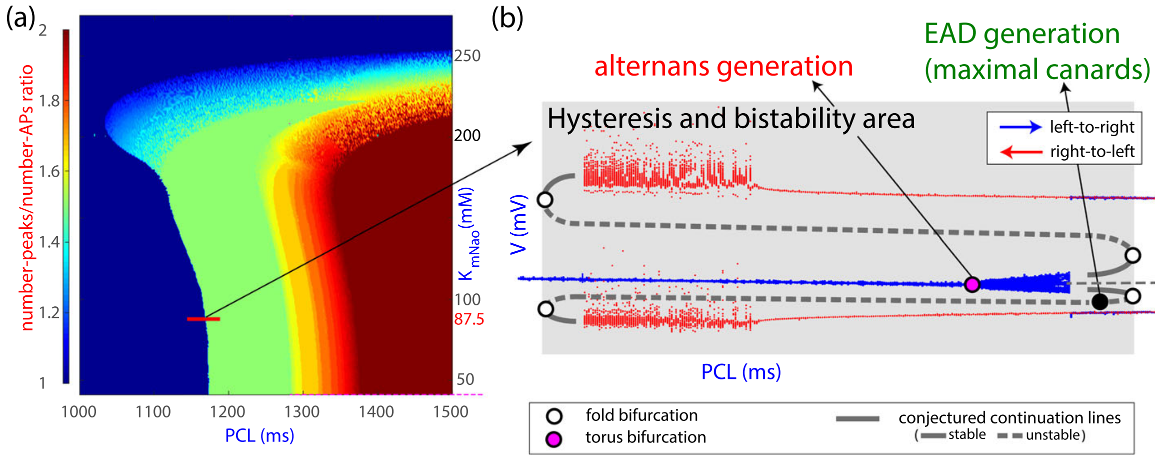


Fig. 20 Results for the EAD transition region using the high dimensional Sato model. In plot **a** we show the biparametric bifurcation diagram taking into account the PCL and the K_{mNao}. In **b** one-parameter bifurcation diagram varying PCL. The points correspond to the transmembrane potential of the AP-peaks. See text for explanation

of 1.5 would thus indicate that an EAD is present in one out of two APs, and so on. This ratio is plotted following a color code. The red bar in the diagram corresponds to the transition region from no EAD to EAD for the standard value of the K_{mNao}. The one-parameter bifurcation diagram varying PCL is plotted in Fig. 20b. The points correspond to the peaks of the AP. On the left (where only the blue line is visible) the ratio is 1 and all APs have a single peak, i.e. present no EADs. On the right (where two red lines are present) the orbit will show two APs: one with EAD and another one without it (both with different AP-peak). Blue points are obtained for increasing PCL values (from left to right) and red points for decreasing PCL values (from right to left). It can be seen that for some PCLs there is coexistence phenomenon, it means that for the same configuration but different initial conditions we obtain different periodic orbits (one with EAD and another one without EADs). Based on our simulations [59] using the 27D Sato model we can conjecture one possible theoretical scheme of the creation of the EAD. First, a torus bifurcation occurs, which allows the creation of alternans and an orbit with different APs. Moving on the bifurcation line, some of the APs of the orbit start to create an EAD and the orbit goes from the configuration without EAD (ratio 1, blue region) towards the configuration with EAD (ratio 1.5, green region). We plot white points and gray lines in Fig. 20b to explain this conjecture. To prove this conjecture we need a simpler model to do a theoretical study.

As with the Hodkin–Huxley model, where a more simplified model was used (the three-dimensional HR model), also for the myocyte we use a more simplified model than Sato, the modified Luo–Rudy 3D model. This model meets, like the HR model, the two basic conditions of being computationally simple but at the same time capable of reproducing the behavior we want to study: the EADs.

The Luo–Rudy 3D model [49, 56] is a simplification of the original Luo–Rudy model [49] following the approach of [56]. This simplified model contains an intermediate time-scale Ca current and a slow K current. The fast Na current was discarded due to its little effects on EAD generation, as it is activated mostly during the AP upstroke but it is practically null during the plateau and repolarization AP phases where the EAD appears. We include, as in Sato model, an additional stimulus current I_{stim} at a constant period defined by the PCL. The three variables of this model correspond to the inactivation gating variable of the Ca current (f), the activation gating variable of the K current (x) and the transmembrane potential (V). After checking that both models experience similar behavior [54], we study

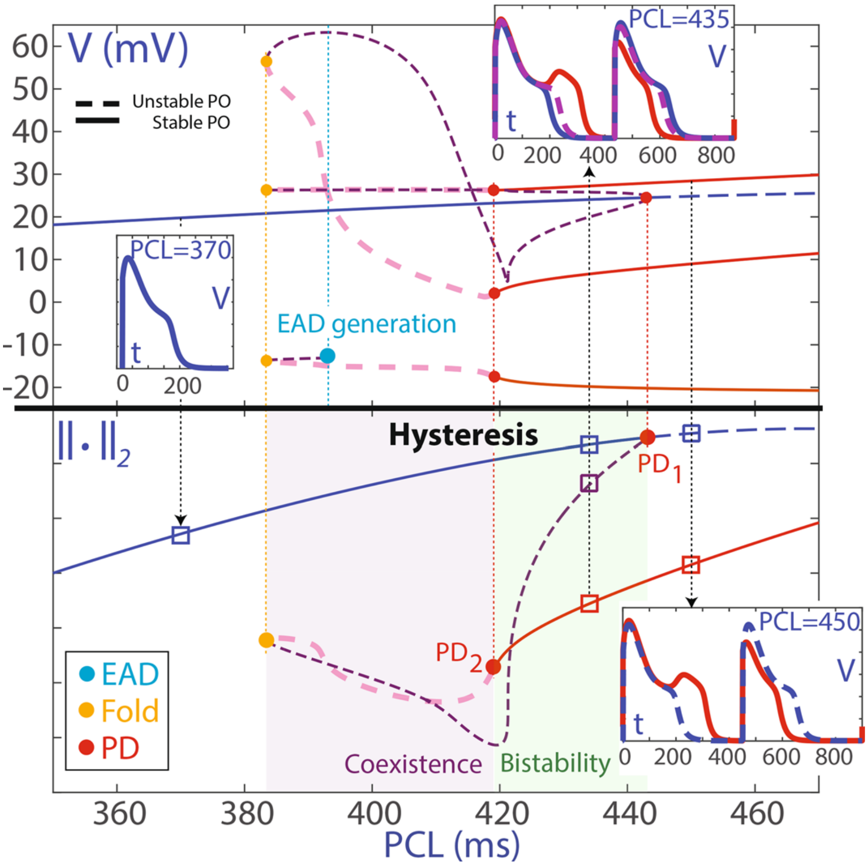


Fig. 21 Taken from [54] with modifications (Figure 6). The top panel shows the EAD transition region between periodic orbits (PO) without and with EAD using the transmembrane potential of the AP-peaks for the low dimensional Luo–Rudy 3D model. The lower panel shows the above bifurcation diagram, but with the AUTO L_2-norm ($\| \cdot \|_2$) of the periodic orbits. We can also see some time series of the existing attractors for three different values of the PCL. See explanation on the text

the same transition from non-EAD to EAD obtaining the complete families of periodic orbits (stable and unstable ones) using the AUTO continuation software [17, 18].

We can see the results in Fig. 21. Top panel shows the EAD transition region between periodic orbits without and with EAD for the Luo–Rudy 3D model. This bifurcation diagram has been completed with the unstable branches calculated by the AUTO continuation software. This gives us the opportunity to visualize the complete evolution of the different periodic orbits [54]. The main bifurcations (period-doubling (PD, in red), and fold of limit cycles (in yellow)) that influence the stability of the periodic orbits are marked with dots of different colors. We distinguish between PD_1, responsible of the alternans generation, and PD_2, responsible of destabilization of the red family and creation of some transient behaviors. In the fold point the periodic orbit family turns generating hysteresis phenomena with different coexistence orbits (with or without bistability). We are also able to detect the EAD generation point (blue dot) which is located in the top plot as the point where a low voltage peak is created in the unstable branch. This point is not visible in the lower panel where we represent the AUTO L_2-norm of the periodic orbits but with this representation is more evident the hysteresis phenomena since it can be easily seen how both stable branches connect through an unstable branch that evolves continuously from one to the other as we conjectured with Sato model in Fig. 20. The bistability and coexistence regions have been marked in green and purple, respectively.

This theoretical scheme explains what we observe in the more biophysical realistic model, i.e., a possible mechanism in the generation of the first EAD. This mechanism is not unique, alternans generation can be caused by another type of bifurcations, but it seems that the generation of alternans is a necessary previous step for the subsequent appearance of the first EADs.

In summary, we have observed, in both the Sato model and the Luo–Rudy 3D model, how the periodic orbit without EAD experiences a bifurcation (torus bifurcation or period-doubling bifurcation) that generates the appearance of alternans. Later, the family with alternans evolves and some of them begin to experience EADs. This evolution occurs in the unstable branch, so it can only be detected by continuing that branch, but its effects are evident in the stable branch.

4 Conclusions

Simplified neuron models can be used to better understand more complex models, once we have ensured that they faithfully reproduce the main dynamic patterns of the complex ones. Low-dimensional fold/hom bursting models can be studied in detail using analytical and numerical methods in order to obtain a full portrait of the different dynamical regimes and bifurcations that appear in their parametric phase space. We have exemplified this for the 3D Hindmarsh–Rose neuron model. We have provided an account of recent results about the behavior of this model when its parameters ε, b and I take values along some specific ranges. Our analysis can be reproduced for other models as well.

For fixed values of the small parameter ε, spike-counting methods give snapshots of the different bursting regions into which the (b, I) parameter plane is divided and the bifurcation lines between them. Adding continuation techniques provides a deeper understanding of these snapshots. Homoclinic bifurcations appear as a key element in spike-adding processes that allows to classify them. We have given a complete theoretical scenario of the interlaced bifurcation diagram for the n to $n + 1$ spike-adding process.

If we let the parameter ε vary, new phenomena are observed. For each number of spikes there appears a homoclinic codimension-one bifurcation surface having a spike-adding role (the model has different number of spikes on different sides of the surface). These surfaces are exponentially close to each other, and their number grows as the small parameter ε tends to zero. Using continuation techniques we have been able to give a detailed description of these surfaces through the concept of geometric bifurcations. Some of these geometric bifurcations are “visible”: they can be easily detected from the plots; but some others (“invisible”) require further analysis in order to be detected.

The topological structure of chaotic attractors changes with the parameters of the system. We have studied these changes using symbolic dynamics and topological templates. The topological templates of the attractors appear as subtemplates of the complete Smale horseshoe template. Moreover, the complete Smale template is obtained as a limit case when the parameter ε tends to zero.

These types of analysis can be applied to other models and in different realistic situations. We have illustrated this with two examples. Once we have tools for studying the dynamics of a single neuron in the 3D Hindmarsh–Rose neuron model, we have studied the dynamics of sets of neurons that are coupled via synapses having inhibitory effect. In particular, we have studied a network of six neurons modeling a CPG for hexapod insects movement. Different values of the parameters for the neuron model produce different gait movements of the insect, and we have shown that some of the main gait movements observed in Nature (tripod and tetrapod gaits) appear as dominant gaits for some regions of the parameter space. Finally, we have studied a different type of electrically excitable cells: cardiomyocytes (cardiac muscle cells). In this case, spike-adding processes are related with Early Afterdepolarization (EAD), a phenomenon of high clinical interest. Numerical simulations on high-dimensional Sato model of the APs of these cells allowed us to conjecture a theoretical scheme for the creation of EADs. As we did in the neuron case, we have driven a more detailed study on a simplified model (Luo–Rudy 3D model), and this study has proved the conjecture about EAD generation that we stated for the high-dimensional model.

Summarizing, spike-adding phenomena are present in numerous theoretical and realistic excitable cells as key ingredients in important changes of the cells.

Funding RB, AL, AM, CM, MAM and SS have been supported by the Spanish Research projects PGC2018-096026-B-I00 and PID2021-122961NB-I00. RB, MAM and SS have been supported by the European Regional Development Fund and Diputación General de Aragón (E24-20R and LMP94-21). CM has been supported by Ministerio de Universidades of Spain with an FPU grant (FPU20/04039). JJ and RV have been supported by the Spanish Research project PID2021-122961NB-I00. SI and LP have been supported by the Spanish Research projects MTM2017-87697-P and PID2020-113052GB-I00. AL has been supported by Diputación General de Aragón project E22-20R.

Data Availability Data available on request from the authors.

Declarations

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Appendix A: Short survey on homoclinic bifurcations

Since this paper deals with homoclinic structures arising in the HR model, we include, for the convenience of the reader, an overview about homoclinic bifurcations, with focus on those detected in the HR model. We mainly follow [21, 60].

Consider a smooth family of vector fields X_μ on $\mathbb{R}^3$ depending on a parameter $\mu \in \mathbb{R}^k$ and suppose that there exist $\mu_0 \in \mathbb{R}^k$ and $p_0 \in \mathbb{R}^3$ such that p_0 is a saddle type hyperbolic equilibrium point of X_{μ_0}. Without loss of generality we can assume $\mu_0 = 0$ and $p_0 = 0$. In fact we can assume that $X_\mu(0) = 0$ for all $\mu \in \mathbb{R}^k$. Because this is the case in the HR model, we only pay attention to saddles with stability index 1 (note that the discussion in the case of stability index equals 2 is identical, but reversing time); hence the vector field has a one-dimensional stable manifold $W^s(0)$ and a 2-dimensional unstable manifold $W^u(0)$. Suppose now that Γ_0 is a homoclinic orbit of X_0 asymptotic to 0. We distinguish two cases:

- All eigenvalues of $DX_0(0)$ are real.
- Unstable eigenvalues of $DX_0(0)$ are complex conjugate.

Moreover, we assume that the connection splits generically when $\mu = 0$.

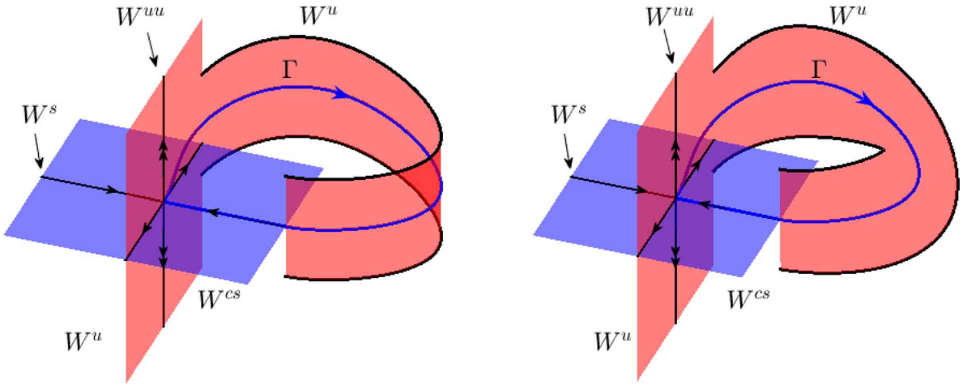


Fig. 22 Unstable manifold, when followed backwards along the homoclinic orbit, can change its orientation

Remark 1 A distance function $\Delta(\mu)$ between $W^s(0)$ and $W^u(0)$ is well defined (see [21]) and we say that the splitting of the connection is generic when $D_\mu \Delta(0)$ does not vanish. Roughly speaking, using Fig. 22 as reference, genericity of the splitting means that $W^s(0)$ splits inwards or outwards, that is, to the left or to the right of W^u, according to the sign of μ.

A.1 Codimension-one homoclinic bifurcations

A.1.1 All eigenvalues are real

In this case, $DX_0(0)$ has real eigenvalues λ_s, λ_u and λ_{uu} such that $\lambda_s < 0 < \lambda_u < \lambda_{uu}$ and we define the so called saddle quantity

$$\sigma = \lambda_s + \lambda_u. \tag{3}$$

A negative (resp. positive) saddle quantity means, geometrically, that the local forward (resp. backward) flow contracts area.

Codimension 1 homoclinic orbits are characterized by the following conditions:

(H1) $\sigma \neq 0$.
(H2) $\Gamma_0 \not\subset W^{uu}(0)$.
(H3) $W^{cs}(0)$ intersects $W^u(0)$ transversally along Γ_0.

Here $W^{uu}(0)$ denotes the one-dimensional strong unstable manifold at 0, whose tangent space at 0 is given by the eigenspace associated with the eigenvalue λ_{uu} (the strong unstable direction), and $W^{cs}(0)$ denotes the 2-dimensional center-stable manifold at 0, whose tangent space at 0 is given by the eigenspace associated with eigenvalues λ_u and λ_s (see Fig. 22).

Condition (**H1**) is a non-resonance condition and (**H2**) means that Γ_0 is tangent to the weak unstable direction, that is, the direction given by the eigenspace associated with the weak unstable eigenvalue λ_u. Condition (**H3**) is a non-inclination property and it requires that the 2-dimensional unstable invariant manifold $W^u(0)$, when followed by the backward flow along Γ, returns along the strong unstable manifold W^{uu} or, equivalently, a transverse intersection between $W^u(0)$ and $W^{cs}(0)$ along the homoclinic connection.

When $\sigma > 0$, a single unstable (repelling) periodic orbit is born from the homoclinic loop when $\mu > 0$ and there is no periodic orbit when $\mu \leq 0$ (see [60, Theorem 13.6]). For positive saddle quantities (**H2**) and (**H3**) play no role. Nevertheless, when $\sigma < 0$ those hypotheses are required and moreover one needs to pay attention to orientability (see Fig. 22) of W^u when followed by the backward flow (see [60, Theorem 13.7]). In the orientable (resp. non-orientable) case a saddle periodic orbit with orientable (resp. non-orientable) stable invariant manifold emerges when $\mu < 0$ (resp. $\mu > 0$).

A.1.2 Complex eigenvalues

In this case the linearization at the equilibrium point has a pair of complex unstable eigenvalues $\rho_u \pm \omega_u \mathrm{i}$ and a real stable eigenvalue λ_s, and we define again the saddle quantity as $\sigma = \rho_u + \lambda_s$. When $\sigma > 0$, the result [[60], Theorem 13.6] remains valid and we know that there exists an unstable periodic orbit when $\mu > 0$. When $\sigma < 0$ it follows from [[60], Theorem 13.8] the existence of infinitely many saddle periodic orbits in any neighborhood of the homoclinic orbit. In fact, as argued in [61], there exist infinitely many horseshoes in any neighborhood of the homoclinic orbit Γ_0. When $\mu \neq 0$ and the connection splits, finitely many of the horseshoes persist and hence it follows the existence of an infinite number of periodic solutions for any value of μ small enough. In fact, each time that a horseshoe is created or destroyed more complex dynamics emerge.

A.2 Codimension-two homoclinic bifurcations

Attending to the description of codimension-one homoclinic bifurcations contained in the previous section, it follows that there may appear the following codimension-two cases:

- Resonant bifurcation: $\sigma = 0$.
- Orbit-flip (OF) bifurcation: $\sigma < 0$ and $\Gamma_0 \subset W^{uu}(0)$.
- Inclination flip (IF) bifurcation: $\sigma < 0$ and the intersection between $W^{cs}(0)$ and $W^u(0)$ is not transversal along Γ_0.

Since no resonant bifurcation is detected in the HR model, we do not discuss details of its unfolding. Nevertheless, orbit- and inclination-flips do appear and seem to be essential ingredients in the dynamics of the system.

A.2.1 Inclination-flips

The essential feature of an inclination-flip is that the intersection between $W^{cs}(0)$ and $W^u(0)$ is not transversal along Γ_0. As a first consequence, the biparametric unfolding includes a curve of homoclinic bifurcations but the orientation is reversed (see Fig. 22) at the IF point.

We introduce the following ratios between eigenvalues

$$\alpha = -\frac{\lambda_{uu}}{\lambda_s}, \qquad \beta = -\frac{\lambda_u}{\lambda_s}, \tag{4}$$

and note that $\alpha > \beta$. We distinguish the following three cases (see Fig. 23):

Case A: $\beta > 1$.
Case B: $\alpha > 1$ and $\frac{1}{2} < \beta < 1$.
Case C: $\alpha < 1$ or $\beta < \frac{1}{2}$.

Each case has its own bifurcation diagram, but we only pay attention to Case C (see Fig. 4).

Hypothesis 1 *Type C inclination-flips are codimension-two bifurcations characterized by the following generic assumptions:*

(I1) $\sigma < 0$.

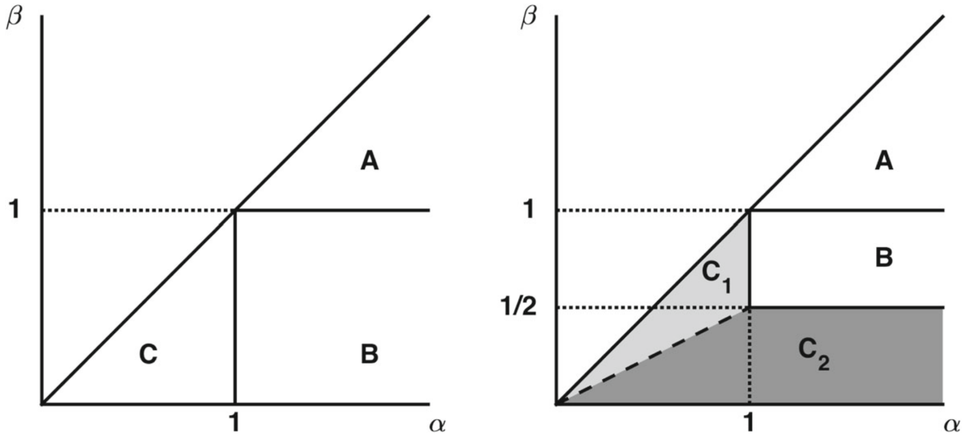


Fig. 23 Regions in the (α, β)-plane corresponding to different cases of orbit-flip (left) and inclination-flip (right)

> ***(I2)*** $\beta \neq \frac{1}{2}\alpha$.
>
> ***(I3)*** *If* $\beta > \frac{1}{2}\alpha$, *the homoclinic orbit does not lie in the unique smooth leading unstable manifold.*
>
> ***(I4)*** *If* $\beta < \frac{1}{2}\alpha$, *there is a quadratic tangency between* $W^{cs}(0)$ *and* $W^u(0)$ *along the homoclinic orbit.*

Hypothesis (I3) makes sense in the region C_1 depicted in Fig. 23 (right). In this case there exists a unique leading unstable manifold which is smooth. Hypothesis (I4) makes sense in the region C_2 depicted in Fig. 23 (right). In this case, when followed by the backward flow, $W^u(0)$ strikes out a parabola-shaped curve on the local center-stable (see lower part of Figure 2 in [62]). Moreover, when $\beta < \frac{1}{2}\alpha$, $W^{cs}(0)$ is a $\mathcal{C}^2$ manifold and hence a quadratic tangency is feasible and both branches (the two branches separated by the connection) converge to the same branch of the local strong unstable manifold. Note that in the region C_1 the tangency is not assumed to be quadratic.

Theorem 2 ([21, 60]) *Assume Hypothesis 1 is satisfied. Depending on a global condition on the stable and unstable manifolds, the bifurcation diagram is given by one of the two cases shown in Fig. 4a. In particular, infinitely many one-sided curves of* N*-homoclinics emerge for each* $N \geq 2$ *from the inclination-flip point at* $\mu = 0$ *on the branch of primary homoclinic orbits.*

To explain the global condition stated in Theorem 2 we need to consider the Poincaré map around the homoclinic orbit. As before, let Γ_0 denote the homoclinic orbit and take a cross-section Σ transversely intersecting Γ_0 at a point p_0. It is easy to verify that the Poincaré map Π is defined on a cusp-shaped domain $D \subset \Sigma$ bounded by curves a, b and c as in Fig. 24 and such that the point p_0 is the cusp point but it is not in D. This domain is mapped by Π to a region $\Pi(D)$ limited by curves a', b', c', images of a, b, and c, respectively, and a curve $l^u = W^u \cap \Sigma$. There are two possibilities, either a' is above l^u (left panels in Fig. 24), or a' is below l^u (right panels in Fig. 24). The first (resp. second) case is called inward (resp. outward) twist and leads to bifurcation diagrams as that depicted in the left (resp. right) panel of Fig. 4 and labeled as type C_{in} (resp. C_{out}).

Let $l^{cs} = W^{cs} \cap \Sigma$ and note that Σ is split into two connected components by l^{cs}. When $\beta > \frac{1}{2}\alpha$, to assume that, as in hypothesis (I3), the homoclinic orbit does not lie in the unique smooth leading unstable manifold means, geometrically, that D is included in one of such connected components (see top panels in Fig. 24). Note that in these cases the tangency

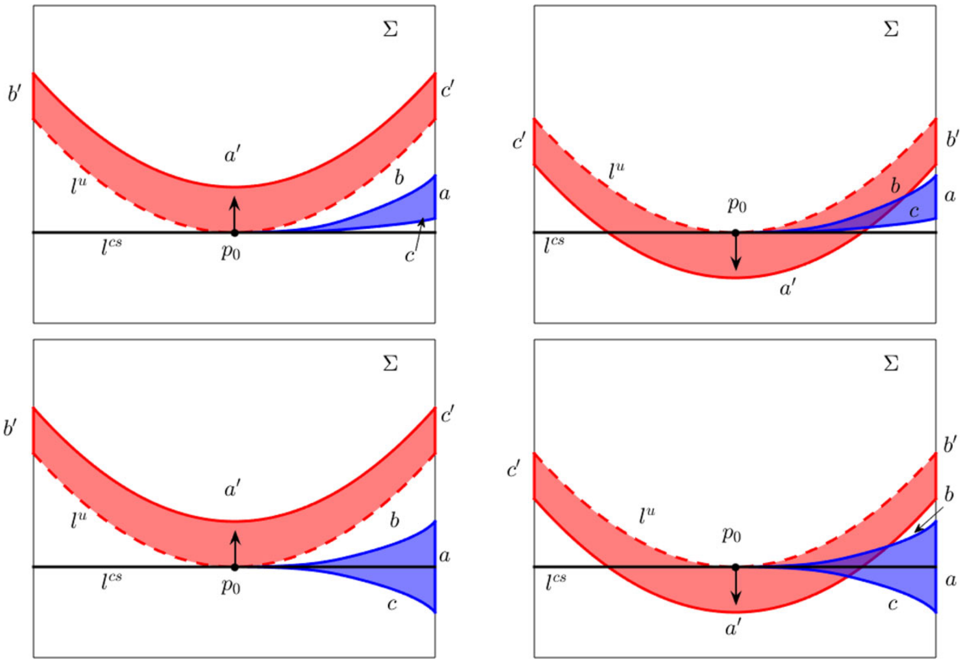


Fig. 24 Blue colored regions represent the domain D where the Poincaré map Π is defined. Red colored regions correspond to $\Pi(D)$. Top (resp. bottom) panels correspond to the case $\beta > \frac{\alpha}{2}$ (resp. $\beta < \frac{\alpha}{2}$). Left (resp. right) panels correspond to the inward (resp. outer) twist

between l^u and l^{cs} is not assumed to be quadratic. On the other hand, hypothesis (I4) means that l^u and l^{cs} have a quadratic contact at p_0 (see bottom panels in Fig. 24).

The distinction between regions C_1 and C_2 corresponds to different ways that a curve which is tangent to the local center stable manifold evolves by local transitions. If $(\alpha, \beta) \in C_1$ then, by a local transition map, such curve will leave a neighborhood of the equilibrium through a weak unstable direction, otherwise, if $(\alpha, \beta) \in C_2$, it will do it through a strong unstable direction (see details in [62]). Nonetheless, this local property, which only depends on eigenvalues, does not lead to differences in the bifurcation diagrams. Another distinction, which leads to different unfoldings and does not depend on eigenvalues, has to do with the behavior of a normal vector when followed by the non-local flow in between two local cross-sections along the homoclinic orbit. This leads to the cases C_{in} and C_{out}.

It should be noticed that following a homoclinic orbit through a primary homoclinic bifurcation curve, an inclination (or orbit) flip bifurcation point drives a change in the orientation of the unstable manifold (see Fig. 22). In both cases C_{in} and C_{out} a region of parameters corresponding to a horseshoe dynamics exists. For inclination-flips of type C_{in} the closure of such region contains one of the branches of primary homoclinic bifurcations; namely, the branch of orientable homoclinic orbits. This is not the case for inclination-flips of type C_{out}; the closure of the region corresponding to shift dynamics does not contain any of the branches of primary homoclinic bifurcations. On the other hand, in both unfoldings a homoclinic doubling cascade appears but, in case C_{in} one of the primary branches of homoclinic orbits separates the cascade from the region of horseshoe dynamics while, in case C_{out}, one of such branches is placed at the beginning of the cascade. As shown in Fig. 4 the region of horseshoe dynamics is placed in between a homoclinic doubling cascade, as already mentioned, and a complicated structure of period-doubling cascades and saddle-node bifurcations of periodic orbits.

A.2.2 Orbit-flips

Bifurcation diagrams at orbit-flips closely resemble those at inclination-flips. The essential feature of an orbit-flip is that the homoclinic orbit approaches the equilibrium along the strong unstable manifold as $t \to -\infty$. With α and β as defined in (4), we distinguish the following three cases (see Fig. 23):

Case A: $\beta > 1$.
Case B: $\beta < 1$ and $\alpha > 1$.
Case C: $\alpha < 1$.

As we do for inclination-flips, we only pay attention to Case C because none of the others arises in the HR model. Additionally, the following nondegeneracy conditions are required:

(**O1**) $\sigma < 0$.
(**O2**) $W^{cs}(0)$ intersects $W^u(0)$ transversally along Γ_0.

Theorem 3 ([21, 60]) *Assume that the nondegeneracy conditions (O1) and (O2) are satisfied. Depending on a global condition on the stable and unstable manifolds, the bifurcation diagram is given by one of the two cases shown in Fig. 4a. In particular, infinitely many one-sided curves of N-homoclinics emerge for each $N \geq 2$ from the orbit-flip point at $\mu = 0$ on the branch of primary homoclinic orbits.*

A.2.3 Belyakov points

The last codimension-two homoclinic bifurcation that we need for our purposes is the Belyakov bifurcation. Original analysis is due to Belyakov [63] but we follow [64] and [21].

Main feature of a Belyakov homoclinic bifurcation is that $DX_0(0)$ has a simple negative real eigenvalue λ_s and a double positive real eigenvalue λ_u with geometric multiplicity one.

We recall that the imaginary part of the unstable eigenvalues at 0 is nonzero if the discriminant $\Delta(\mu)$ of $DX_\mu(0)$ restricted to the unstable generalized eigenspace is negative. Considering $\mu = (\mu_1, \mu_2)$ we can assume, without loss of generality, that $\partial\Delta/\partial\mu_1(0) \neq 0$.

We assume that $\lambda_u < -\lambda_s$. Otherwise, the bifurcation set is simple, a unique unstable cycle bifurcates from the homoclinic orbit for $\mu_2 > 0$. In both cases, crossing the line $\mu_2 = 0$ when $\mu_1 < 0$ results in the appearance of a single limit cycle.

As in the previous configurations, the splitting of the homoclinic connections is supposed to be generic, and hence the two-parameter unfolding includes a curve of homoclinic bifurcation through $\mu = 0$ and crossing transversely the line $\mu_1 = 0$.

We also assume conditions which play the role of *non orbit-flip* and *non inclination-flip* properties. We assume that the homoclinic orbit does not belong to the strong unstable manifold, that is, $\|h(t)\| \approx Kte^{\lambda_u t}$ as $t \to -\infty$, where $h(t)$ denotes the homoclinic orbit. Moreover following the unstable manifold backwards along the homoclinic orbit, the manifold tends toward the strong unstable direction.

Hence we have the following result:

Theorem 4 ([21, 60]) *We may assume that the primary homoclinic orbit exists for $\mu_2 = 0$. Hence the homoclinic orbits are of saddle-focus type for $\mu_1 > 0$ and of saddle-node type for $\mu_2 < 0$. There are then infinitely many one-sided curves of 2-homoclinic orbits in the half plane $\mu_1 > 0$ that emerge from $\mu = 0$, they are tangent to $\mu_2 = 0$ at $\mu = (0, 0)$ and accumulate onto $\mu_2 = 0$ from one side. Furthermore, there are infinitely many one-sided* *curves of saddle-node and period-doubling bifurcation of periodic orbits in $\mu_2 > 0$ that emerge from $\mu = 0$. They are tangent to $\mu_2 = 0$ at $\mu = (0, 0)$, and accumulate onto $\mu_2 = 0$ from both sides.*

Bifurcation diagrams are similar to those in Fig. 4a (right panel) but saddle-node and period-doubling bifurcations curves arise at both sides of the homoclinic bifurcation curve.

Remark 2 It can also be proved that N-homoclinic orbits bifurcate for each $N \geq 2$.

References

1. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. **117**(4), 500–544 (1952)
2. Hindmarsh, J.L., Rose, R.M.: A model of neuronal bursting using three coupled first order differential equations. Proc. R. Soc. Lond. **B221**, 87–102 (1984)
3. Ermentrout, G.B., Terman, D.H.: Mathematical Foundations of Neuroscience. Interdisciplinary Applied Mathematics, vol. 35. Springer, New York (2010)
4. Broens, M., Bar-Eli, K.: Canard explosion and excitation in a model of the Belousov–Zhabotinskii reaction. J. Phys. Chem. **95**(22), 8706–8713 (1991)
5. Wieczorek, S., Krauskopf, B., Lenstra, D.: Multipulse excitability in a semiconductor laser with optical injection. Phys. Rev. Lett. **88**(6), 063901 (2002)
6. Izhikevich, E.M.: Dynamical Systems in Neuroscience. The Geometry of Excitability and Bursting. MIT Press, Cambridge (2007)
7. Barrio, R., Shilnikov, A.: Parameter-sweeping techniques for temporal dynamics of neuronal systems: case study of Hindmarsh–Rose model. J. Math. Neurosci. **1**(1), 1–22 (2011)
8. Barrio, R., Martínez, M.A., Serrano, S., Shilnikov, A.: Macro- and micro-chaotic structures in the Hindmarsh–Rose model of bursting neurons. Chaos **24**(2), 023128 (2014)
9. Hirata, Y., Oku, M., Aihara, K.: Chaos in neurons and its application: perspective of chaos engineering. Chaos **22**(4), 047511 (2012)
10. Korn, H., Faure, P.: Is there chaos in the brain? II. Experimental evidence and related models. C. R. Biologies **326**(9), 787–840 (2003)
11. Terman, D.: Chaotic spikes arising from a model of bursting in excitable membranes. SIAM J. Appl. Math. **51**(5), 1418–1450 (1991)
12. Linaro, D., Champneys, A., Desroches, M., Storace, M.: Codimension-two homoclinic bifurcations underlying spike adding in the Hindmarsh–Rose burster. SIAM J. Appl. Dyn. Syst. **11**(3), 939–962 (2012)
13. Barrio, R., Ibáñez, S., Pérez, L.: Homoclinic organization in the Hindmarsh–Rose model: a three parameter study. Chaos **30**(5), 053132–20 (2020)
14. Barrio, R., Ibáñez, S., Pérez, L., Serrano, S.: Classification of fold/hom and fold/Hopf spike-adding phenomena. Chaos **31**(4), 043120–14 (2021)
15. Shilnikov, A., Kolomiets, M.: Methods of the qualitative theory for the Hindmarsh–Rose model: a case study. A tutorial. Int. J. Bifurc. Chaos **18**(8), 2141–2168 (2008)
16. Storace, M., Linaro, D., de Lange, E.: The Hindmarsh–Rose neuron model: bifurcation analysis and piecewise-linear approximations. Chaos **18**(3), 033128 (2008)
17. Doedel, E.: AUTO: a program for the automatic bifurcation analysis of autonomous systems. In: Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing, vol. I (Winnipeg, Man., 1980), vol. 30, pp. 265–284 (1981)
18. Doedel, E.J., Paffenroth, R.C., Champneys, A.R., Fairgrieve, T.F., Kuznetsov, Y.A., Oldeman, B.E., Sandstede, B., Wang, X.J.: Auto2000. http://cmvl.cs.concordia.ca/auto
19. Barrio, R., Ibáñez, S., Pérez, L.: Hindmarsh–Rose model: close and far to the singular limit. Phys. Lett. A **381**(6), 597–603 (2017)
20. Desroches, M., Kaper, T.J., Krupa, M.: Mixed-mode bursting oscillations: dynamics created by a slow passage through spike-adding canard explosion in a square-wave burster. Chaos **23**(4), 046106 (2013)
21. Homburg, A.J., Sandstede, B.: Homoclinic and heteroclinic bifurcations in vector fields. Handb. Dyn. Syst. **3**, 379–524 (2010)
22. Barrio, R., Ibáñez, S., Pérez, L., Serrano, S.: Spike-adding structure in fold/hom bursters. Commun. Nonlinear Sci. Numer. Simul. **83**, 105100 (2020)
23. Barrio, R., Ibáñez, S., Pérez, L.: Homoclinic organization in fold/hom bursters: the Hindmarsh–Rose model. Chaos **30**(5), 053132 (2019)

24. Barrio, R., Ibáñez, S., Pérez, L.: Geometry of bifurcation sets: exploring the parameter space. Preprint (2022)
25. Serrano, S., Martínez, M.A., Barrio, R.: Order in chaos: structure of chaotic invariant sets of square-wave neuron models. Chaos **31**(4), 043108 (2021)
26. Gilmore, R., Lefranc, M.: The Topology of Chaos: Alice in Stretch and Squeezeland. WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim (2011)
27. Hao, B., Zheng, W.: Applied Symbolic Dynamics and Chaos. World Scientific, Singapore (2018)
28. Bucher, D., Haspel, G., Golowasch, J., Nadim, F.: Central Pattern Generators, pp. 1–12. Wiley, New Jersey (2015)
29. Lamb, D.G., Calabrese, R.L.: Neural circuits controlling behavior and autonomic functions in medicinal leeches. Neural Syst. Circuits **1**(1), 1–10 (2011)
30. Calabrese, R.L., Norris, B.J., Wenning, A., Wright, T.M.: Coping with variability in small neuronal networks. Integr. Comp. Biol. **51**(6), 845–855 (2011)
31. Kristan, W.B., Calabrese, R.L.: Rhythmic swimming activity in neurones of the isolated nerve cord of the leech. J. Exp. Biol. **65**(3), 643–668 (1976)
32. Masino, M.A., Calabrese, R.L.: Phase relationships between segmentally organized oscillators in the leech heartbeat pattern generating network. J. Neurophysiol. **87**(3), 1572–1585 (2002)
33. Masino, M.A., Calabrese, R.L.: Period differences between segmental oscillators produce intersegmental phase differences in the leech heartbeat timing network. J. Neurophysiol. **87**(3), 1603–1615 (2002)
34. Ayali, A., Borgmann, A., Büschges, A., Couzin-Fuchs, E., Daun-Gruhn, S., Holmes, P.: The comparative investigation of the stick insect and cockroach models in the study of insect locomotion. Curr. Opin. Insect Sci. **12**, 1–10 (2015)
35. Fujiki, S., Aoi, S., Funato, T., Tomita, N., Senda, K., Tsuchiya, K.: Hysteresis in the metachronal-tripod gait transition of insects: a modeling study. Phys. Rev. E **88**(1), 012717 (2013)
36. Ritzmann, R., Zill, S.N.: Neuroethology of insect walking. Scholarpedia **8**(9), 30879 (2013)
37. Ghigliazza, R.M., Holmes, P.: Minimal models of bursting neurons: how multiple currents, conductances, and timescales affect bifurcation diagrams. SIAM J. Appl. Dyn. Syst. **3**(4), 636–670 (2004)
38. Ghigliazza, R.M., Holmes, P.: A minimal model of a central pattern generator and motoneurons for insect locomotion. SIAM J. Appl. Dyn. Syst. **3**(4), 671–700 (2004)
39. Tedeschi, F., Carbone, G.: Design issues for hexapod walking robots. Robotics **3**(2), 181–206 (2014)
40. Campos, R., Matos, V., Santos, C.: Hexapod locomotion: a nonlinear dynamical systems approach. In: IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society, pp. 1546–1551 (2010)
41. Toth, T.I., Schmidt, J., Büschges, A., Daun-Gruhn, S.: A neuro-mechanical model of a single leg joint highlighting the basic physiological role of fast and slow muscle fibres of an insect muscle system. PLoS One **8**(11), 78247 (2013)
42. Mantziaris, C., Bockemühl, T., Holmes, P., Borgmann, A., Daun, S., Büschges, A.: Intra-and intersegmental influences among central pattern generating networks in the walking system of the stick insect. J. Neurophysiol. **118**(4), 2296–2310 (2017)
43. Tytell, E.D., Holmes, P., Cohen, A.H.: Spikes alone do not behavior make: why neuroscience needs biomechanics. Curr. Opin. Neurobiol. **21**(5), 816–822 (2011)
44. Barrio, R., Lozano, A., Rodríguez, M., Serrano, S.: Numerical detection of patterns in CPGs: gait patterns in insect movement. Commun. Nonlinear Sci. Numer. Simul. **82**, 105047 (2020)
45. Barrio, R., Lozano, A., Martínez, M.A., Rodríguez, M., Serrano, S.: Routes to tripod gait movement in hexapods. Neurocomputing **461**, 679–695 (2021)
46. Noble, D.: A modification of the Hodgkin-Huxley equations applicable to Purkinje fibre action and pace-maker potentials. J. Physiol. **160**(2), 317–352 (1962)
47. McAllister, R.E., Noble, D., Tsien, R.W.: Reconstruction of the electrical activity of cardiac Purkinje fibres. J. Physiol. **251**(1), 1–59 (1975)
48. Beeler, G.W., Reuter, H.: Reconstruction of the action potential of ventricular myocardial fibres. J. Physiol. **268**(1), 177–210 (1977)
49. Luo, C., Rudy, Y.: A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interaction. Circ. Res. **68**(6), 1501–1526 (1991)
50. Luo, C., Rudy, Y.: A dynamic model of the cardiac ventricular action potential. I. Simulations of ionic currents and concentration changes. Circ. Res. **74**(6), 1071–1096 (1994)
51. Luo, C., Rudy, Y.: A dynamic model of the cardiac ventricular action potential. II. Afterdepolarizations, triggered activity, and potentiation. Circ. Res. **74**(6), 1097–1113 (1994)
52. Barrio, R., Martínez, M.A., Pérez, L., Pueyo, E.: Bifurcations and slow-fast analysis in a cardiac cell model for investigation of early afterdepolarizations. Mathematics **8**(6), 880 (2020)

53. Sato, D., Xie, L.H., Sovari, A.A., Tran, D.X., Morita, N., Xie, F., Karagueuzian, H., Garfinkel, A., Weiss, J.N., Qu, Z.: Synchronization of chaotic early afterdepolarizations in the genesis of cardiac arrhythmias. Proc. Natl. Acad. Sci. **106**(9), 2983–2988 (2009)
54. Barrio, R., Martínez, M.A., Serrano, S., Pueyo, E.: Dynamical mechanism for generation of arrhythmogenic early afterdepolarizations in cardiac myocytes: insights from in silico electrophysiological models. Phys. Rev. E **106**(2), 024402 (2022)
55. Weiss, J.N., Garfinkel, A., Karagueuzian, H.S., Chen, P.S., Qu, Z.: Early afterdepolarizations and cardiac arrhythmias. Heart Rhythm **7**(12), 1891–1899 (2010)
56. Sato, D., Xie, L.H., Nguyen, T.P., Weiss, J.N., Qu, Z.: Irregularly appearing early afterdepolarizations in cardiac myocytes: random fluctuations or dynamical chaos? Biophys. J. **99**(3), 765–773 (2010)
57. Mahajan, A., Shiferaw, Y., Sato, D., Baher, A., Olcese, R., Xie, L.H., Yang, M.J., Chen, P.S., Restrepo, J.G., Karma, A., Garfinkel, A., Qu, Z., Weiss, J.N.: A rabbit ventricular action potential model replicating cardiac dynamics at rapid heart rates. Biophys. J. **94**(2), 392–410 (2008)
58. Otte, S., Berg, S., Luther, S., Parlitz, U.: Bifurcations, chaos, and sensitivity to parameter variations in the Sato cardiac cell model. Commun. Nonlinear Sci. Numer. Simul. **37**, 265–281 (2016)
59. Barrio, R., Martínez, M.A., Pueyo, E., Serrano, S.: Dynamical analysis of early afterdepolarization patterns in a biophysically detailed cardiac model. Chaos **31**(7), 073137 (2021)
60. Shilnikov, L.P., Shilnikov, A.L., Turaev, D.V., Chua, L.O.: Methods of Qualitative Theory in Nonlinear Dynamics. Part II. World Scientific, Singapore (2001)
61. Tresser, C.: About some theorems by L. P. Shilnikov. Ann. Inst. H. Poincaré Phys. Théor. **40**(4), 441–461 (1984)
62. Homburg, A.J., Krauskopf, B.: Resonant homoclinic flip bifurcations. J. Dyn. Differ. Equ. **12**(4), 807–850 (2000)
63. Belyakov, L.A.: Bifurcation set in a system with homoclinic saddle curve. Mat. Zametki **28**(6), 911–922 (1980)
64. Kuznetsov, Y.A., De Feo, O., Rinaldi, S.: Belyakov homoclinic bifurcations in a tritrophic food chain model. SIAM J. Appl. Math. **62**(2), 462–487 (2001)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Roberto Barrio[1] · Santiago Ibáñez[2] · Jorge A. Jover-Galtier[1] · Álvaro Lozano[3] · M. Ángeles Martínez[1] · Ana Mayora-Cebollero[1] · Carmen Mayora-Cebollero[1] · Lucía Pérez[2] · Sergio Serrano[1] · Rubén Vigara[1]

Santiago Ibáñez
mesa@uniovi.es

Jorge A. Jover-Galtier
jorgejover@unizar.es

Álvaro Lozano
alozano@unizar.es

M. Ángeles Martínez
gelimc@unizar.es

Ana Mayora-Cebollero
amayora@unizar.es

Carmen Mayora-Cebollero
cmayora@unizar.es

Lucía Pérez
perezplucia@uniovi.es

Sergio Serrano
sserrano@unizar.es

Rubén Vigara
rvigara@unizar.es

[1] Departamento de Matemática Aplicada and IUMA, Computational Dynamics Group (http://cody.unizar.es/), Universidad de Zaragoza, Zaragoza, Spain

[2] Departamento de Matemáticas, Universidad de Oviedo, Oviedo, Spain

[3] Departamento de Matemáticas and IUMA, Computational Dynamics Group, Universidad de Zaragoza, Zaragoza, Spain

Chaos
An Interdisciplinary Journal of Nonlinear Science

RESEARCH ARTICLE | JULY 21 2023

Deep Learning for chaos detection

Roberto Barrio ✉ ; Álvaro Lozano ; Ana Mayora-Cebollero ; Carmen Mayora-Cebollero ; Antonio Miguel ; Alfonso Ortega ; Sergio Serrano ; Rubén Vigara

Check for updates

Chaos 33, 073146 (2023)
https://doi.org/10.1063/5.0143876

View Online

Export Citation

CrossMark

21 July 2023 16:30:37

Reprinted from R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection", Chaos: An Interdisciplinary Journal of Nonlinear Science, vol. 33, no. 7, p. 073146, 2023. https://doi.org/10.1063/5.0143876, with the permission of AIP Publishing.

AIP Publishing

Chaos ARTICLE pubs.aip.org/aip/cha

Deep Learning for chaos detection

Cite as: Chaos **33**, 073146 (2023); doi: 10.1063/5.0143876
Submitted: 27 January 2023 · Accepted: 30 June 2023 ·
Published Online: 21 July 2023

View Online Export Citation CrossMark

Roberto Barrio,[1,a)] Álvaro Lozano,[2,b)] Ana Mayora-Cebollero,[1,c)] Carmen Mayora-Cebollero,[1,d)] Antonio Miguel,[3,e)] Alfonso Ortega,[3,f)] Sergio Serrano,[1,g)] and Rubén Vigara[1,h)]

AFFILIATIONS

[1]Departamento de Matemática Aplicada and IUMA, Computational Dynamics group, Universidad de Zaragoza, Zaragoza E-50009, Spain
[2]Departamento de Matemáticas and IUMA, Computational Dynamics group, Universidad de Zaragoza, Zaragoza E-50009, Spain
[3]Departamento de Ingeniería Electrónica y Comunicaciones, ViVoLab, Aragón Institute for Engineering Research (I3A), University of Zaragoza, Zaragoza E-50018, Spain

[a)]**Author to whom correspondence should be addressed:** rbarrio@unizar.es
[b)]**Electronic mail:** alozano@unizar.es
[c)]**Electronic mail:** amayora@unizar.es
[d)]**Electronic mail:** cmayora@unizar.es
[e)]**Electronic mail:** amiguel@unizar.es
[f)]**Electronic mail:** ortega@unizar.es
[g)]**Electronic mail:** sserrano@unizar.es
[h)]**Electronic mail:** rvigara@unizar.es

21 July 2023 16:30:37

ABSTRACT

In this article, we study how a chaos detection problem can be solved using Deep Learning techniques. We consider two classical test examples: the Logistic map as a discrete dynamical system and the Lorenz system as a continuous dynamical system. We train three types of artificial neural networks (multi-layer perceptron, convolutional neural network, and long short-term memory cell) to classify time series from the mentioned systems into *regular* or *chaotic*. This approach allows us to study biparametric and triparametric regions in the Lorenz system due to their low computational cost compared to traditional techniques.

Published under an exclusive license by AIP Publishing. https://doi.org/10.1063/5.0143876

Deep Learning techniques have recently been introduced in the area of dynamical systems. These new tools can speed up studies and permit us to go deeper into simulations. Of all the problems in which these methodologies can help us, we focus on the problem of detecting chaos, showing how Deep Learning allows, in a fast way, us to handle large amounts of data, such as 2D and 3D parametric phase space studies, and therefore, they can be powerful techniques in the global analysis.

I. INTRODUCTION

One of the main topics in Dynamical Systems is the detection of chaotic regions in the parameter space. The classical technique to detect chaos is the use of Lyapunov exponents (LEs).[1,2,4] Recently, some authors have applied Deep Learning (DL) techniques in dynamical systems to handle different tasks, mainly for forecasting problems,[7,19] but also for chaos detection.[3,6,14] Here, we are interested in the latter of these tasks.

In this paper, we choose three common DL architectures (multi-layer perceptron, convolutional neural network, and long short-term memory cell) for chaos detection in time series from a dynamical system. We provide a detailed analysis of the learning process of the networks, and we are able to use the trained networks to reproduce 1D, 2D, and 3D parametric plots. Remarkably, we are able to obtain the behavior of a dynamical system in regions of the parameter space where the DL techniques have not been trained. Moreover, as far as we know, this is the first time in the literature that a dense 3D parametric plot of a continuous dynamical system, such as the Lorenz system, is represented.

This paper is organized as follows. In Sec. II, we describe the chosen DL networks giving a brief introduction of each of them. In Secs. III and IV, we perform the chaos detection task in the Logistic

Published under an exclusive license by AIP Publishing

map and the Lorenz system, respectively. We give a detailed description of the datasets and network architectures and comment on the obtained results. In Sec. IV E, we present 2D and 3D parametric diagrams of the Lorenz system computed with trained networks. Finally, we draw some conclusions in Sec. V.

All the DL experiments in this paper have been performed using PyTorch.[17] The code was executed on a Linux box with dual Xeon ES2697 with 128 Gb of DDR4-2133 memory with a RTX2080Ti GPU.

II. INTRODUCTION TO THE DEEP LEARNING ARCHITECTURES USED TO DETECT CHAOS

Deep Learning[9,11] is the branch of machine learning that uses deep artificial neural networks to learn from data with several levels of abstraction. Artificial Neural Networks (ANNs) are formed by artificial neurons (loosely inspired by their biological counterparts) organized in layers.

Of all the DL architectures found in the literature, the multilayer perceptron (MLP) is the simplest one and it is widely used for this reason. Convolutional neural networks (CNNs) and long short-term memory networks (LSTMs) have been previously used to analyze time series data,[3,6,14] as in our chaos detection experiments. We have tested these three well-known ANN architectures to detect chaos.

For the MLP, we start with a similar architecture to that used in Ref. 3. For the CNN and LSTM, we use a not very complicated structure, and we do not perform hyperparameter optimization. A more detailed study of network architectures may improve our results, and it is part of our future research.

Remark II.1. Although our mathematical problem is called *chaos detection*, from the point of view of DL, it is a *binary classification task* instead of a detection task: our networks classify the input vectors (time series corresponding to an orbit of a dynamical system) into two disjoint categories (*regular* vs *chaotic*).

A. Multi-layer perceptron

One of the fundamental Deep Learning architectures is multilayer perceptron.[9] It operates by taking a linear combination of inputs in each layer, followed by a non-linear activation function. In Fig. 1(a), we have an example of an MLP whose output y is given by

$$y = W^{[3]}\mathscr{A}\left(W^{[2]}\mathscr{A}\left(W^{[1]}x + b^{[1]}\right) + b^{[2]}\right) + b^{[3]},$$

where x is the input, $W^{[l]}$ is the matrix of weights of the connections between layer $l-1$ and layer l (layers are enumerated from 0 to 3 from left to right), $b^{[l]}$ represents the bias vector of layer l, and $\mathscr{A}$ is a non-linear activation function such as the rectified linear unit $\mathrm{ReLU}(x) = \max(0, x)$, the sigmoid $\sigma(x) = (1 + e^{-x})^{-1}$, or the hyperbolic tangent $\tanh(x) = 2\sigma(2x) - 1$.

B. Convolutional neural network

Convolutional neural networks were originally developed for image recognition tasks[13] and are organized into convolutional and pooling layers to capture features and reduce dimensions, respectively. One of the key features of CNNs is that they share weights across multiple neurons[5] for more efficient processing. They handle different input formats such as vectors, matrices, or 3D tensors, depending on the type of convolution used. In this paper, the input data are in the vector form, and therefore, we focus on the use of 1D CNNs, as depicted in Fig. 1(b).

To exemplify how a CNN works, we show how to compute the value of the shaded neuron in the second layer of the network in Fig. 1(b), which is given by

$$x_{0,0}^{[1]} = \mathscr{A}\left(b_0^{[1]} + \sum_{j=0}^{1}\sum_{k=0}^{2} w_{j,k,0}^{[1]}\, x_{j,k}^{[0]}\right),$$

where $x_{j,k}^{[l]}$ is the activation of neuron j of channel k at layer l (the first index for the neurons, the channels, and the layers is 0), $W_0^{[1]} = (w_{j,k,0}^{[1]})_{j=0,1;\,k=0,1,2}$ is the weight matrix, $b_0^{[1]}$ is the bias vector, and $\mathscr{A}$ is the activation function. We could have more complex CNN architectures[10,18] if we consider stride, dilation, residual connections, etc.

C. Long short-term memory

Recurrent neural networks (RNNs) are commonly used for sequential processing since they retain some information from past inputs. Long short-term memory cells[12] represent a specific type of RNN architecture. Among the distinctive elements of an LSTM are the hidden state h and the cell state c, which are the elements keeping information from previous steps. Computations performed by such type of memory cells [represented in Fig. 1(c)] are as follows:

$$\begin{aligned}
f(t) &= \sigma\left(W_f^{[x]}x(t) + W_f^{[h]}h(t-1) + b_f\right),\\
g(t) &= \tanh\left(W_g^{[x]}x(t) + W_g^{[h]}h(t-1) + b_g\right),\\
i(t) &= \sigma\left(W_i^{[x]}x(t) + W_i^{[h]}h(t-1) + b_i\right),\\
c(t) &= f(t)\otimes c(t-1) + i(t)\otimes g(t),\\
o(t) &= \sigma\left(W_o^{[x]}x(t) + W_o^{[h]}h(t-1) + b_o\right),\\
y(t) &= h(t) = o(t)\otimes \tanh(c(t)),
\end{aligned} \tag{1}$$

where $x(t)$ is the external input, $y(t)$ is the usual output, $W_*^{[x,h]}$ and b_* represent the matrix of weights and the bias term for the external input or the hidden state, $\otimes$ is the element-wise product, and σ and tanh are activation functions. Roughly speaking, as a consequence of the application of the sigmoid activation function in left formulas in (1), f, i, and o are deciding which information is kept and which is removed in the output and hidden and cell states.

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

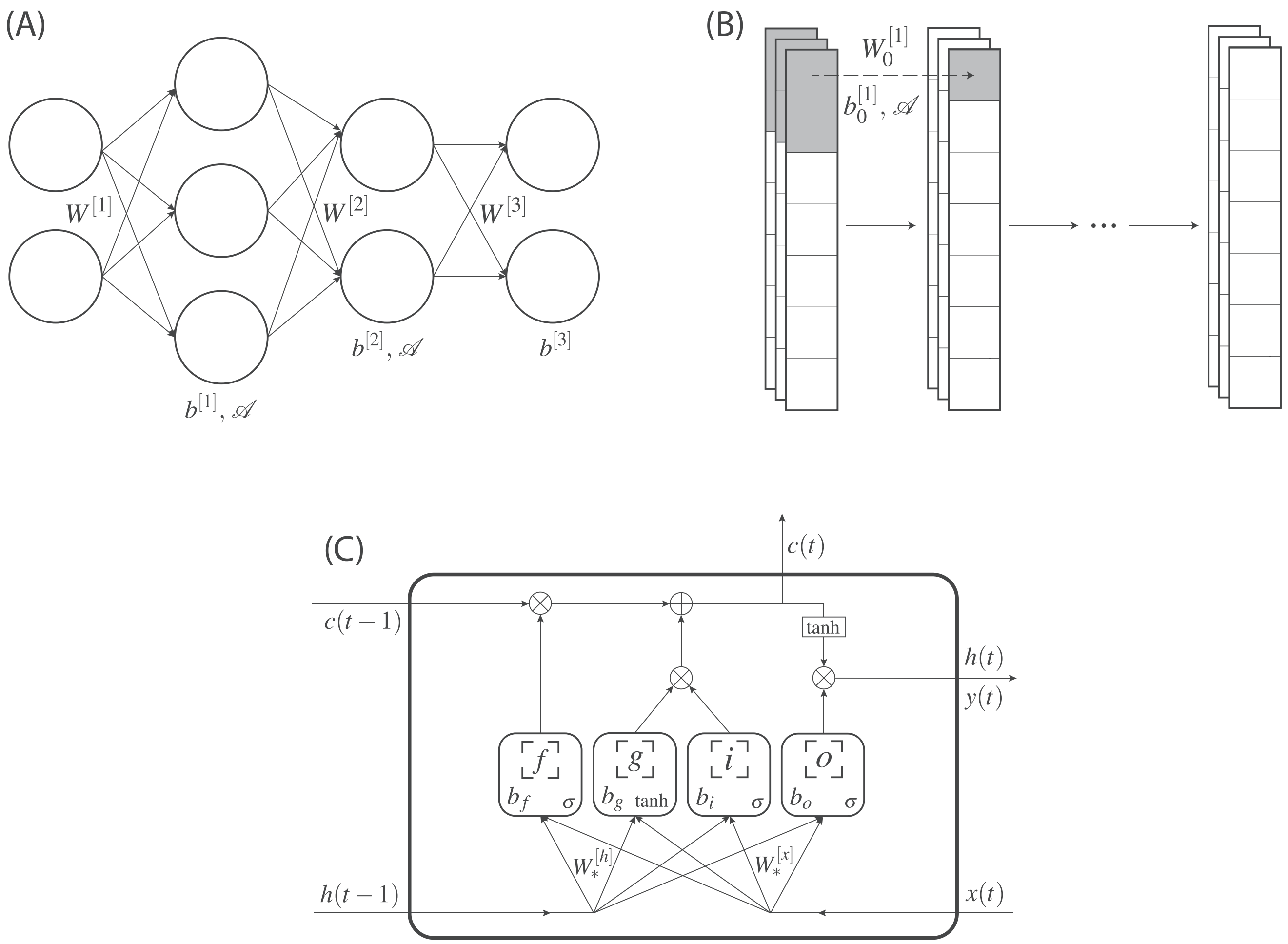


FIG. 1. Simple graphic representations of the architectures of (a) an MLP with two hidden layers, (b) a 1D CNN with three channels in the represented convolutional layers, and (c) a generic LSTM cell.

III. TEST EXAMPLE OF DISCRETE DYNAMICAL SYSTEMS: THE LOGISTIC MAP

The Logistic map[16] is a well-studied model that describes the dynamics of animal populations. It is given by the equation

$$x_{n+1} = \alpha x_n(1 - x_n), \tag{2}$$

where x_n is the variable in the nth iteration and α is the bifurcation parameter. Note that since (2) is symmetric with respect to $x = 0.5$, the evolution of the initial condition x_0 is exactly the same as the one of $1 - x_0$. The Lyapunov exponent for this map is

$$\text{LE} = \lim_{k\to+\infty} \frac{1}{k} \sum_{n=0}^{k-1} \log|\alpha(1 - 2x_n)|, \tag{3}$$

where log is the natural logarithm.

A. Dataset

To prepare the training, test, and validation sets for the Logistic map, we generate multiple raw datasets of time series samples, which are subsequently screened. Each raw dataset comprises time series with a fixed length of 1000. The initial condition x_0 is fixed for all the time series in each raw dataset ($x_0 \in \{0.5, 0.9\}$ for training dataset, $x_0 = 0.75$ for validation, and $x_0 = 0.8$ for test), and the bifurcation parameter α takes 12 000 equidistant values in the interval $[0, 4)$. The time series of the raw datasets are the last 1000 time steps obtained applying the iterative formula (2) for 12 000 time steps. The LE of each sample is approximated applying formula (3) with the last 11 000 time steps of the time series (the first 1000 time steps are considered as transient time).

From the union of the raw datasets with $x_0 = 0.5$ and $x_0 = 0.9$, we obtain the training dataset. In particular, we split the joint dataset into regular and chaotic samples (chaotic if the approximated LE is

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

TABLE I. Loss and accuracy (%) of training, validation, and test datasets for the Logistic map test problem. Results in this and similar tables or figures are given as mean±standard deviation for 10 different trials (see Remark III.1).

	MLP		CNN		LSTM	
	Loss	Accuracy (%)	Loss	Accuracy (%)	Loss	Accuracy (%)
Training	0.026 ± 0.001	99.302 ± 0.161	0.018 ± 0.007	99.383 ± 0.335	0.059 ± 0.020	97.979 ± 0.584
Validation	0.107 ± 0.003	96.775 ± 0.068	0.037 ± 0.004	98.925 ± 0.203	0.104 ± 0.048	96.605 ± 1.337
Test	0.046 ± 0.002	97.865 ± 0.098	0.018 ± 0.004	99.410 ± 0.239	0.045 ± 0.013	98.565 ± 0.279

greater than 0.1, to reduce the transient behavior as done in several studies; and regular otherwise), obtaining 21 791 regular and 2209 chaotic time series. We delete the samples that are similar (we consider that two samples are similar if their distance in infinity norm is less than 10^{-4}) to avoid repeated time series. In this process, the number of chaotic samples is not reduced, but the number of regular samples decreases to 6402. After shuffling these datasets (to have time series from different space regions in the subsequent selection), we choose 2000 chaotic and 2000 regular samples for the training set.

Validation dataset is obtained from raw dataset with $x_0 = 0.75$ (it contains 10 896 regular and 1104 chaotic samples). The training, test, and validation datasets have to be pairwise disjoint. So, we drop any validation sample similar in the previous sense to a training sample. After this process, we have 1268 regular and 1104 chaotic time series. We build the validation dataset with 1000 regular and 1000 chaotic samples randomly taken.

For test dataset, we use raw dataset with $x_0 = 0.8$ (it has 10 896 regular and 1104 chaotic time series). After data selection, the number of chaotic samples does not change, but the number of regular ones decreases to 5705. After shuffling, we build the test dataset with 1000 regular and 1000 chaotic samples.

To perform a supervised DL training as the one that concerns us, we need the network inputs (that we have already created) and the expected labels, i.e., whether the inputs are regular or chaotic. We label a times series with 0 if it is regular and with 1 if it is chaotic. To input the data into networks, we split each dataset into different batches. The batch sizes of the training, validation, and test sets are 128, 100, and 100, respectively (in the case of the training set, the last incomplete batch is deleted).

B. Multi-layer perceptron

Our architecture is inspired by the one in Ref. 3, with some changes as the overfitting technique (instead of dropout, we perform early stopping and L^2-regularization). It has five layers: an input layer with 1000 neurons (the length of the input), three hidden layers with 500 neurons each one, and an output layer with 2 neurons (regular vs chaotic). The ReLU activation function is applied in the hidden layers and the softmax function in the output layer. This network is trained for 2000 epochs, saving the first model with the lowest value of the loss function for the validation dataset (early stopping technique). Here, we use the cross-entropy loss with weight decay of 10^{-5} for the L^2-penalty, optimized with the stochastic gradient descent with learning rate of 10^{-3}.

TABLE II. Accuracy (%) of regular and chaotic samples in the test set for the Logistic map test problem.

	MLP	CNN	LSTM
Regular	99.860 ± 0.049	99.710 ± 0.094	99.030 ± 0.827
Chaotic	95.870 ± 0.200	99.110 ± 0.509	98.100 ± 0.775

C. Convolutional neural network

Our CNN architecture has two 1D convolutional layers with 5 and 10 channels, kernel size 10 and 5, dilation equal to 2 and 4, respectively, and stride 1 for both. Each convolutional layer adds a bias term and applies the ReLU activation function. We use zero-padding and cropping to ensure that the length of the output sequence remains the same as the input since the stride is equal to one. A global average pooling layer is applied after the last convolutional layer. A binary classification layer with two neurons is stacked at the end. A bias term is added and the softmax activation function is applied. The CNN is trained for 2000 epochs using the early stopping technique. The loss function is the cross-entropy loss with a weight decay of 10^{-5} for L^2-regularization. For this architecture, the optimizer is Adam with a learning rate of 0.008.

D. Long short-term memory

Our LSTM architecture consists of two stacked layers followed by a linear layer with two output neurons for classification. Both LSTM layers are unidirectional, their states have dimension 4, and bias terms are considered. The input of the linear layer is the last hidden state of both LSTM cells. The network is trained for 2000 epochs with early stopping. The loss function and the optimizer are the same as the ones used for the CNN in Sec. III C.

E. Results

The accuracy is the measure used to determine the performance of all the networks. It is computed with the following formula:

$$\text{Accuracy (\%)} = \frac{T_R + T_C}{T_R + T_C + F_R + F_C} \cdot 100,$$

where T_R and T_C represent the number of regular and chaotic samples, respectively, that the network has classified correctly and F_R and F_C are the number of chaotic and regular samples, respectively, that have been wrongly classified. To assure that the network has learnt correctly, that is, the percentage of correct classifications of

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

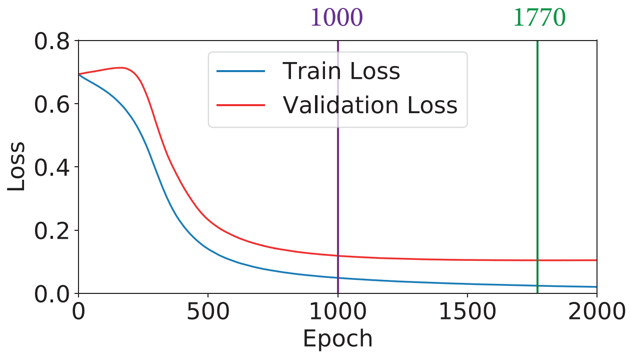


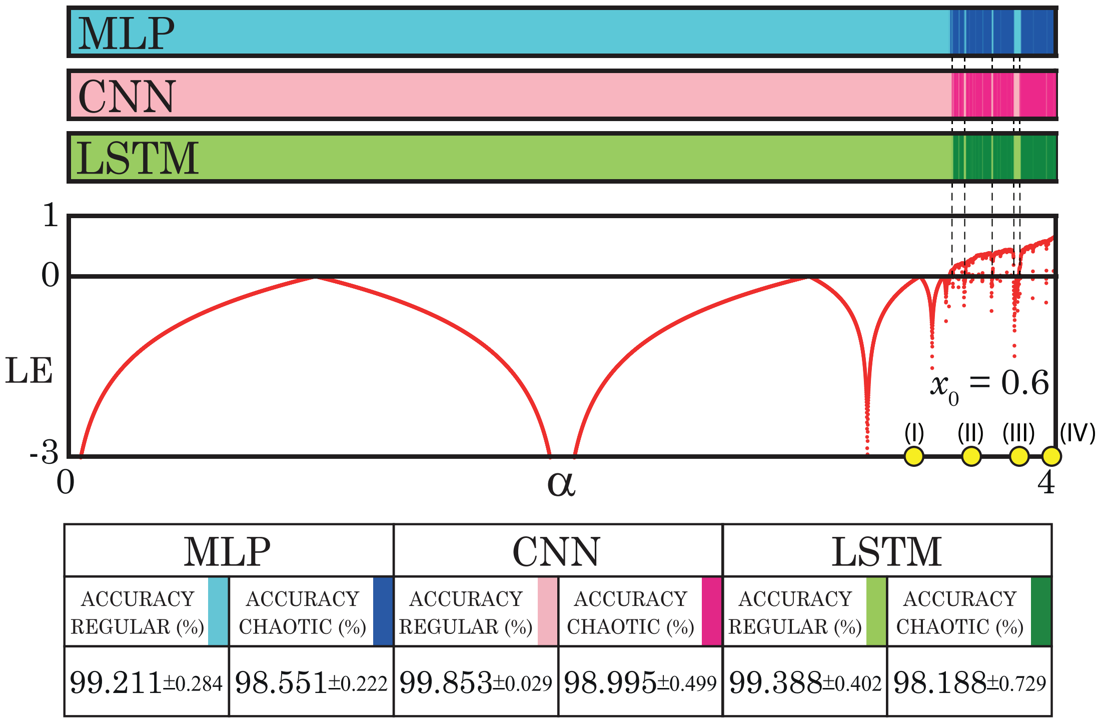
FIG. 2. Evolution along 2000 epochs of the loss (left panel) and accuracy (right panel) in training (blue) and validation (red) datasets for one MLP trained in the Logistic map test problem. Green line corresponds to the best epoch (1770) and purple line to the recommended epoch (1000).

both classes is balanced, the following magnitudes are computed:

$$\text{Accuracy Regular (\%)} = \frac{T_R}{T_R + F_C} \cdot 100 \text{ and}$$

$$\text{Accuracy Chaotic (\%)} = \frac{T_C}{T_C + F_R} \cdot 100.$$

Remark III.1. The performance results of the networks indicated in this subsection and Secs. IV C and IV D are the mean and the standard deviation of the binary classification of 10 trained networks randomly initialized by PyTorch (the graphic results correspond to a unique network). The experiments in Sec. IV E are carried out using one of the trained networks.



MLP		CNN		LSTM	
ACCURACY REGULAR (%) ■	ACCURACY CHAOTIC (%) ■	ACCURACY REGULAR (%) ■	ACCURACY CHAOTIC (%) ■	ACCURACY REGULAR (%) ■	ACCURACY CHAOTIC (%) ■
99.211±0.284	98.551±0.222	99.853±0.029	98.995±0.499	99.388±0.402	98.188±0.729

FIG. 3. Chaos detection task with the MLP, the CNN, and the LSTM in an α-parametric line ($x_0 = 0.6$) of the Logistic map test problem. From top to bottom, three color bars with the results (light for regular, dark for chaotic) of the trained networks (MLP in blue, CNN in pink and LSTM in green), approximated LEs, and table with the accuracy of regular and chaotic samples for the three types of architectures. In yellow, α values of the orbits in Fig. 5.

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

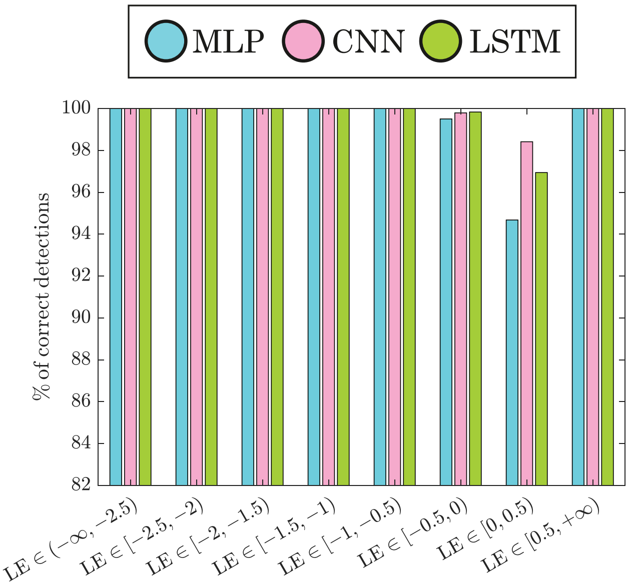


FIG. 4. Histogram of the percentage of correct detections according to the value of the approximated LE (blue for the MLP, pink for the CNN, and green for the LSTM) for the α-parametric line with $x_0 = 0.6$ (see Fig. 3) of the Logistic map test problem.

We show in Table I the loss and accuracy of training, validation and test datasets for the three ANNs (MLP, CNN, and LSTM). Notice that for all the networks, the training and test losses are quite small, so we can consider that the ANNs have been able to learn from data correctly. If we compare the results for the test data, we can see that our CNN seems to give better results than the other two networks. For completion, we can compute the accuracy in the regular and chaotic samples of the test set, obtaining the results in Table II. As we can see, the percentages of regular and chaotic samples are quite similar (and always close to 100%), so our ANNs have learnt correctly the main features of both types of dynamical regimes.

We can also analyze the evolution of the loss and accuracy of the training and validation datasets along the 2000 epochs to study the learning process of the networks. To exemplify it, we choose one of the trained MLPs (for all the trained networks, a similar situation occurs). In Fig. 2, we have drawn the evolution of the loss and accuracy of training (blue) and validation (red) datasets along 2000 epochs. Notice that from epoch 1000 (approximately), the loss and the accuracy of the training dataset do not change significantly, so although we can see that the mathematical optimum of the loss function for the validation dataset has been obtained at epoch 1770 (best epoch, see green line in Fig. 2), we could say that the computational optimum has been achieved in epoch 1000 approximately (see purple line in Fig. 2). We can conclude that we could train the network for less number of epochs, and the obtained minimum of the loss function would be similar and still acceptable (moreover, the training time would be less).

Once we have successfully trained our networks, we carry out the chaos detection task in one parametric line of the Logistic map: we fix $x_0 = 0.6$ (we cannot take $x_0 \in \{0.1, 0.2, 0.25, 0.5, 0.75, 0.8, 0.9\}$ as these are the values used to create datasets and their symmetric cases with respect to $x = 0.5$), we take 12 000 values of $\alpha \in [0, 4)$, and we use each network to detect the behavior of the system. In Fig. 3, we have a table with the accuracy of the networks for each dynamical regime. We have also drawn the approximated LEs in red

21 July 2023 16:30:37

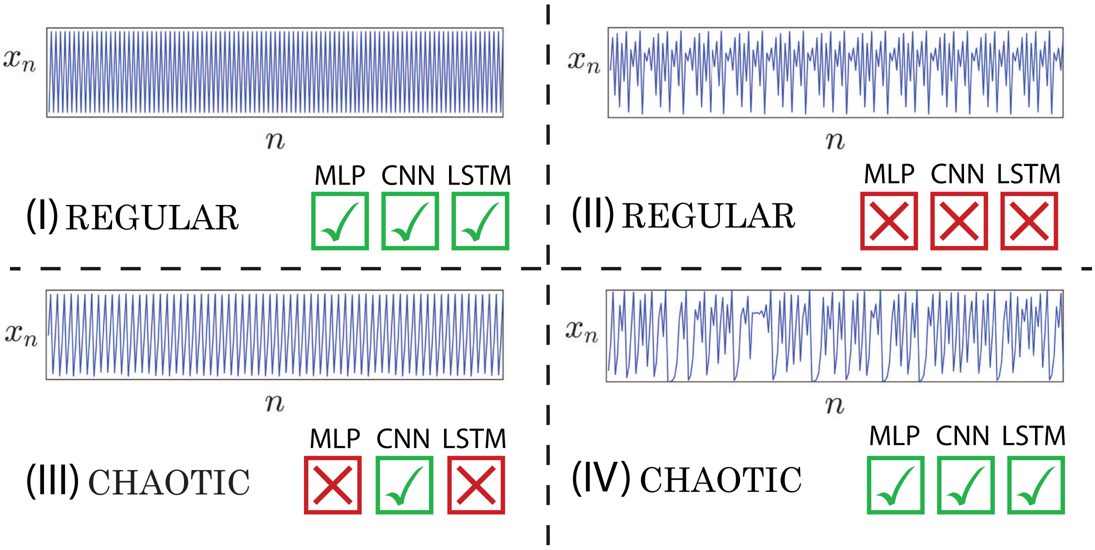


FIG. 5. Some orbits of the Logistic map, their true dynamical behavior, and the success or error of the classification made by different networks. For a correct visualization, only the first 200 steps of the Logistic map have been drawn. In all cases $x_0 = 0.6$, (I) $\alpha = 3.432\,333\,230\,972\,29$, (II) $\alpha = 3.655\,999\,898\,910\,522\,5$, (III) $\alpha = 3.856\,666\,549\,414\,062$, and (IV) $\alpha = 3.999\,666\,690\,826\,416$ (see yellow points in Fig. 3).

Published under an exclusive license by AIP Publishing

TABLE III. Loss and accuracy (%) of training, validation, and test datasets for the Lorenz system test problem.

	MLP		CNN		LSTM	
	Loss	Accuracy (%)	Loss	Accuracy (%)	Loss	Accuracy (%)
Training	0.072 ± 0.002	98.844 ± 0.074	0.046 ± 0.011	98.194 ± 0.464	0.045 ± 0.006	98.125 ± 0.271
Validation	0.194 ± 0.002	94.125 ± 0.189	0.063 ± 0.006	97.720 ± 0.205	0.042 ± 0.005	98.120 ± 0.200
Test	0.179 ± 0.002	93.350 ± 0.613	0.085 ± 0.071	97.575 ± 0.990	0.051 ± 0.030	97.870 ± 1.326

(that have been computed with a classical technique as explained in Sec. III A) and in three color bars we have the results given by the networks. Light colors are used for regular regimes and dark ones for chaotic behavior. Blue is the color chosen for the MLP, pink for the CNN, and green for the LSTM. As we can see in the aforementioned table, all the networks are able to perform the chaos detection task: the accuracy is close to 100% for all the architectures and dynamical regimes. Furthermore, in three color bars, we can see that all the networks are able to detect quite well the boundaries between both dynamical regimes (see the first dotted line from left to right), and they have detected the regular windows in the chaotic region (see the remaining dotted lines in the image).

In Fig. 4, we have a histogram with the percentage of correctly classified time series in the α-parametric line according to the value of the approximated LE (100% means that all the samples whose LE is in the corresponding interval have been correctly detected). Blue bars correspond to the MLP, pink ones to the CNN, and green ones to the LSTM network. Notice that when the LE is far from 0, the three networks perform the task perfectly. Otherwise, the percentage of correct detections is lower but still larger than 90%, with the MLP giving the worst results.

It is interesting to analyze some correct and incorrect classifications of the DL networks (see yellow points in Fig. 3). In case I of Fig. 5 we show a periodic orbit (regular behavior) time series that all the networks have been able to classify correctly. The sample of case II is also regular, but in this case, it is misclassified by all the models. In case III, the orbit is chaotic and only the CNN detects it correctly. Finally, in case IV, the sample is chaotic too, and all the networks classify it correctly. Case II illustrates a periodic orbit with many oscillations, being extremely difficult to recognize as it has a similar behavior to some chaotic samples in the dataset. Case III illustrates another common dynamical situation, a chaotic behavior but where the chaos is weak, that is, the "irregularity" is small, being quite similar to a periodic orbit, and so, the DL techniques can also be wrong. Note that these two cases with wrong detections are also complicated cases for standard techniques. In these cases, expert researchers would use their background to classify the behavior. In any case, these boundary cases are just a small percentage of the tests, and most of the data are correctly classified as regular or chaotic.

TABLE IV. Accuracy (%) of regular and chaotic samples in the test set for the Lorenz system test problem.

	MLP	CNN	LSTM
Regular	92.680 ± 0.481	97.710 ± 1.724	97.770 ± 2.759
Chaotic	94.020 ± 0.306	97.440 ± 0.709	97.970 ± 0.438

IV. TEST EXAMPLE OF CONTINUOUS DYNAMICAL SYSTEMS: THE LORENZ SYSTEM

The Lorenz system[15] is a very simple model representing cellular convection. It is given by the following system of equations:

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

where x, y, and z are the variables, σ is the Prandtl number, r is the relative Rayleigh number, and b is a positive constant. The Lyapunov exponents of a continuous dynamical system are computed with the algorithm in Ref. 20.

A. Dataset

As in the case of the Logistic map, we create several raw datasets (with time series of the Lorenz system), and then we screen them to obtain the three sets needed to train, validate, and test the networks. Each raw dataset satisfies that the length of the time series is 1000 (in Sec. IV D, we justify our choice), and the initial condition is fixed to $(x_0, y_0, z_0) = (1, 1, 1)$ (the initial value used by other authors[3] in this chaos detection task). Bifurcation parameter σ is fixed to the classical value 10, the relative Rayleigh number r moves equidistantly in the interval $(0, 300]$ to obtain 5999 values, and the positive constant b is fixed along all the samples of each raw dataset ($b \in \{2, 8/3\}$ to build the training dataset, $b = 2.4$ for validation, and $b = 2.8$ for test). A transient interval is performed until time $t = 100\,000$ with time step 0.01 using DOPRI5 (Runge–Kutta integrator of order 5); the LEs are computed using 10 001 more unit times with a time step of 0.001 and the same integrator; the time series that we use as an input in the DL architectures are built with 1 out of every 100 of the last 100 000 computed points.

From the union of raw datasets with $b = 2$ and $b = 8/3$, we obtain the training set. In particular, we split the joint dataset into regular and chaotic samples taking 0.01 as threshold (chaotic if the first approximated LE is bigger than 0.01 and regular otherwise), obtaining 4091 regular and 7907 chaotic time series. To ensure variability, we delete all but one similar samples (that is, samples at distance less than 10^{-4} in the infinity norm). From the resulting set with 4054 regular and 7907 chaotic samples, we randomly choose 3900 samples of each dynamical class as training dataset. To build

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

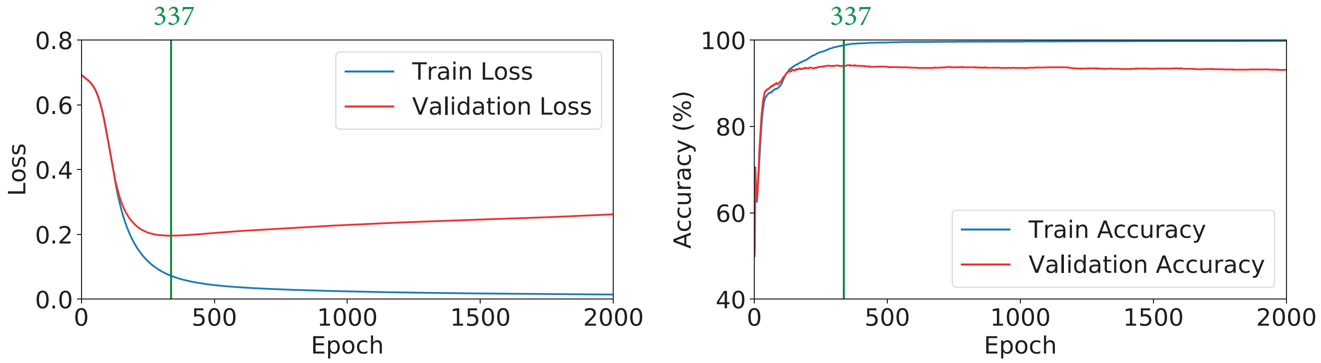


FIG. 6. Evolution along 2000 epochs of the loss (left panel) and the accuracy (right panel) in training (blue) and validation (red) datasets for one MLP trained in the Lorenz system test problem. Green line corresponds to the best epoch (337).

the validation dataset, we consider the raw dataset with $b = 2.4$, with 2278 regular and 3721 chaotic samples. From it, we remove all samples similar (in the previous sense) to any of the training set (2259 regular and 3721 chaotic samples remain) and we choose 1000 of each class as validation dataset. Finally, from the 2902 regular and 3097 chaotic samples forming the raw dataset with $b = 2.8$, we remove all similar samples to any from the previous sets. From the resulting set (with 2884 regular and 3097 chaotic samples), we select 1000 from each class to create the test set.

As with the Logistic map, regular samples are labelled with 0 and chaotic ones with 1. The batch sizes of the training, validation, and test sets are 128, 100, and 100, respectively (the last incomplete batch of the training set is deleted). In the case of the Lorenz system, we normalize each coordinate independently, linearly mapping its range to the interval $[0, 1]$. If a coordinate is constant, we assign to it a random number uniformly sampled from $[0, 1]$.

B. Multi-layer perceptron, convolutional neural network, and long short-term memory

The only change in the architecture of the MLP that we have considered for the Lorenz system case with respect to the one used for the Logistic map is the number of neurons in each layer (except in the output one): 3000 neurons in the input layer (take into account that we still consider the length of each sample equal to 1000, but Lorenz system has dimension 3, so the flattened time series has length 3000) and 1500 on each hidden layer.

As in the case of the MLP, the CNN used for the Lorenz system is similar to the one used for the Logistic map. The number of channels in the convolutional layers has changed: 15 for the first one and 30 for the second. Moreover, the input layer has now three channels instead of one (one for each variable of the system).

In the case of the LSTM, we use the same Deep Learning architecture described for the Logistic map, but now the dimension of the states is 24. Again, we have to take into account the structure of the input because now we work with a three-dimensional dynamical system, so the external input of the LSTM is of size 3 instead of 1 (at each time step).

C. Results

In Table III, we show the loss and accuracy of training, validation, and test datasets for the MLP, the CNN, and the LSTM. For all the networks, the training and test losses are quite small, so we can

21 July 2023 16:30:37

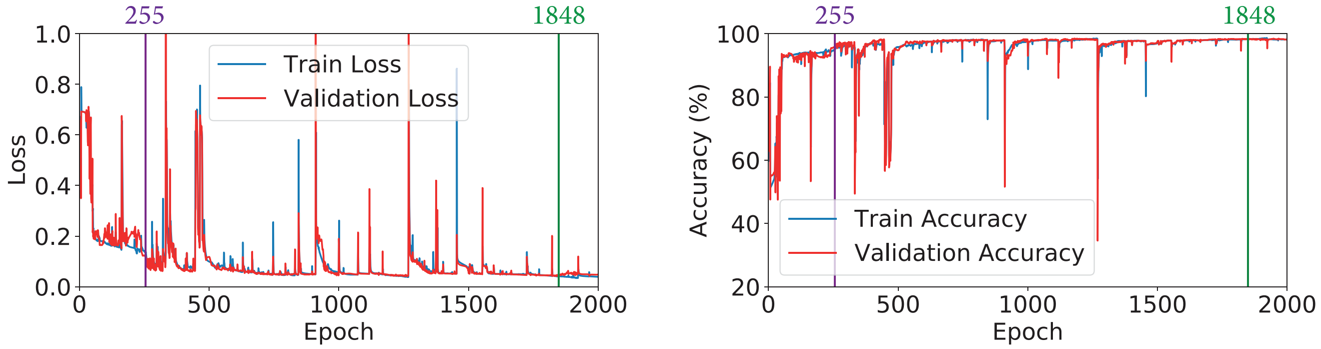
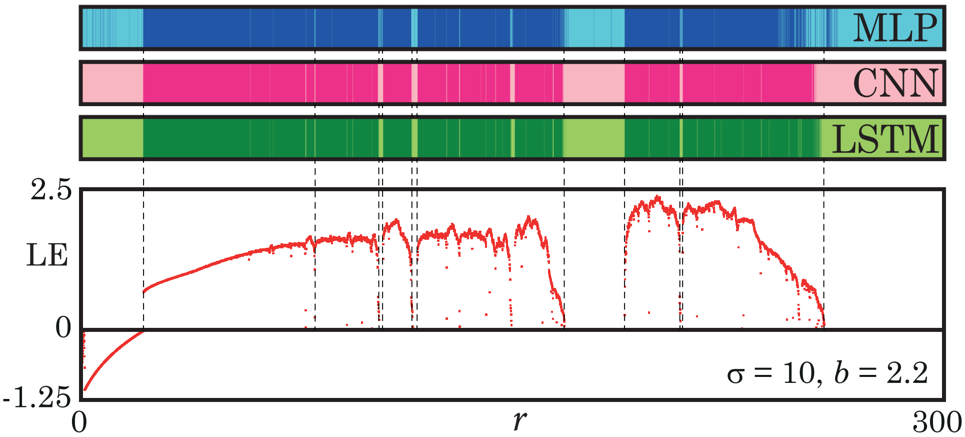


FIG. 7. Evolution along 2000 epochs of the loss (left panel) and accuracy (right panel) of the training (blue) and validation (red) datasets for one LSTM trained in the Lorenz system test problem. Green line corresponds to the best epoch (1848) and purple one to recommended epoch (255).

Published under an exclusive license by AIP Publishing



MLP		CNN		LSTM	
ACCURACY REGULAR (%) 🟦	ACCURACY CHAOTIC (%) 🟦	ACCURACY REGULAR (%) 🟪	ACCURACY CHAOTIC (%) 🟪	ACCURACY REGULAR (%) 🟩	ACCURACY CHAOTIC (%) 🟩
91.015±0.104	96.172±0.174	95.652±1.489	97.638±0.644	96.864±1.020	98.314±0.322

FIG. 8. Chaos detection task performed by the MLP, the CNN, and the LSTM in the Lorenz system test problem in an *r*-parametric line. From top to bottom, three color bars representing the results (light for regular, dark for chaotic) of the trained networks (MLP in blue, CNN in pink, and LSTM in green), approximated first LE, and table with the accuracy of regular and chaotic samples for three types of architectures.

21 July 2023 16:30:37

consider that they have been able to learn from the data correctly. If we compare the results for the test data, we can see that the CNN and the LSTM seem to give better results than the MLP. If we compare the CNN and the LSTM, they show similar performance results. For completion, the accuracy of regular and chaotic samples of the test set are shown in Table IV. Notice that the percentages of both classes are similar for all the trained ANNs, so we conclude that the networks have learnt correctly the main features of both dynamical regimes.

Now, we analyze the evolution of the loss and the accuracy of training and validation datasets along 2000 epochs to study how this learning process is. In the case of the MLP and the CNN, the networks are able to learn with a small number of epochs as the best epoch is usually less than 750. If we visualize such evolution (see Fig. 6 for an example of a trained MLP), it seems that the network has learnt as much as it can because, after the best epoch, the validation loss increases while the training loss decreases (the network suffers overfitting). In the case of the LSTM (see Fig. 7 for an example of a trained LSTM network), the evolution is more similar to that of the Logistic map since acceptable results would be obtained with fewer number of epochs (the minimum of the validation dataset would be similar and less computational time would be needed). In the figure, the best epoch is 1848, but around epoch 250, the loss has already reached a value close to zero and the accuracy is close to 100% (so, 255 is a recommended epoch taking into account the little improvement of going up to 1848). Some erratic jumps highlight

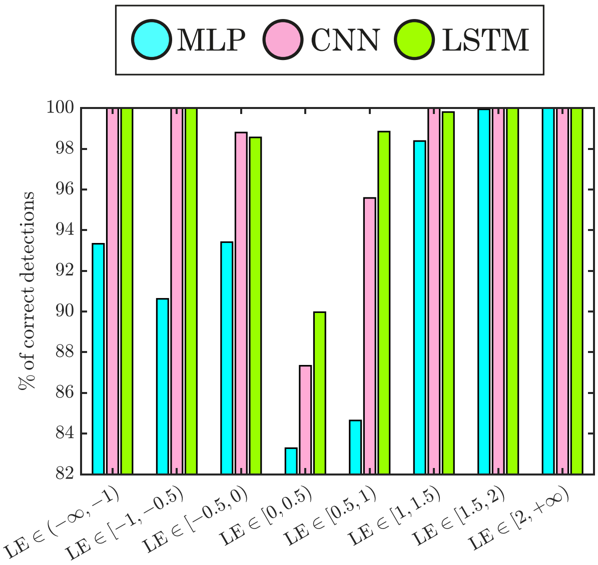


FIG. 9. Histogram of the percentage of correct detections according to the value of the approximated first LE (blue for the MLP, pink for the CNN, and green for the LSTM) for the *r*-parametric line with $\sigma = 10$ and $b = 2.2$ (see Fig. 8) of the Lorenz system test problem.

Published under an exclusive license by AIP Publishing

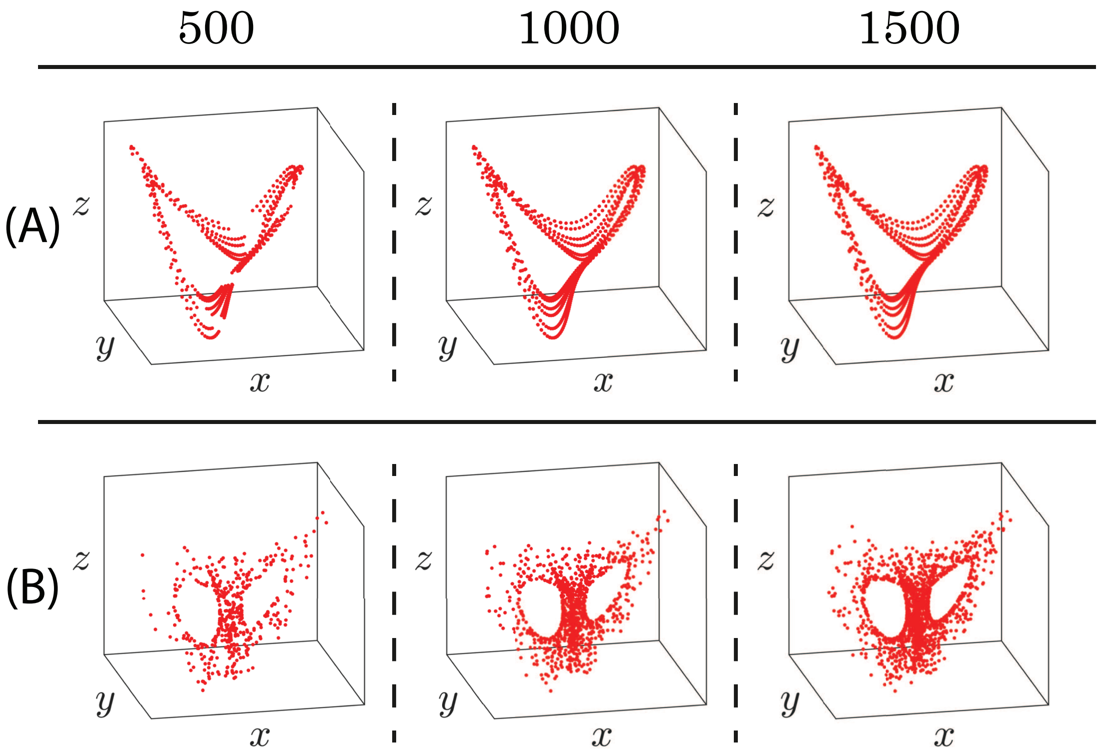


FIG. 10. Orbits I (panel A) and V (panel B) of Fig. 14 for time series length 500 (left), 1000 (middle), and 1500 (right) of the Lorenz system test problem.

21 July 2023 16:30:37

in this learning process. This phenomenon can be explained by the fact that some samples [see, for example, Fig. 14(i)] have a difficult dynamical behavior (they are in the limit between the regular and chaotic dynamics), which causes the networks to have trouble in learning them.

All the networks have learnt correctly from the data. To show the performance of the three DL architectures, we choose an r-parametric line of this continuous dynamical system different from the ones used to create train, test, and validation sets. In particular, we take $\sigma = 10$, $b = 2.2$, and 6000 equidistant values of r in the interval $[0, 300]$. In Fig. 8, we have a table with the accuracy achieved by each architecture for each dynamical regime (regular and chaotic). Moreover, the first approximated LE is represented in red in the middle of the figure. In three color bars, the results given by each network (same code of colors as in Fig. 3) are depicted. It can be seen that all the networks are able to perform the chaos detection task in the Lorenz system test problem (the accuracy is greater than 90% in all cases). In addition, in the color bars, we see that all the ANNs can define correctly the boundary between both dynamical regimes in most cases (see first, seventh, and eighth dotted lines from left to right). However, the networks are not able to detect the last boundary (represented with the last dotted line) perfectly (the LSTM gives the best detection), but the results are quite acceptable. Notice that the MLP shows, in general, noisier results (see region before first dotted line, between seventh and eighth dotted lines, and around last dotted line in the blue bar). All the DL architectures can detect most of the regular windows in the chaotic regions (see remaining dotted lines).

TABLE V. Loss and accuracy (%) of training, validation, and test datasets with different time series length for the Lorenz system test problem.

	Length 500		Length 1000		Length 1500	
	Loss	Accuracy (%)	Loss	Accuracy (%)	Loss	Accuracy (%)
Training	0.050 ± 0.005	97.858 ± 0.239	0.045 ± 0.006	98.125 ± 0.271	0.057 ± 0.037	97.703 ± 1.279
Validation	0.049 ± 0.004	97.980 ± 0.222	0.042 ± 0.005	98.120 ± 0.200	0.054 ± 0.028	97.825 ± 0.972
Test	0.054 ± 0.012	97.855 ± 0.652	0.051 ± 0.030	97.870 ± 1.326	0.062 ± 0.031	97.315 ± 1.711

Published under an exclusive license by AIP Publishing

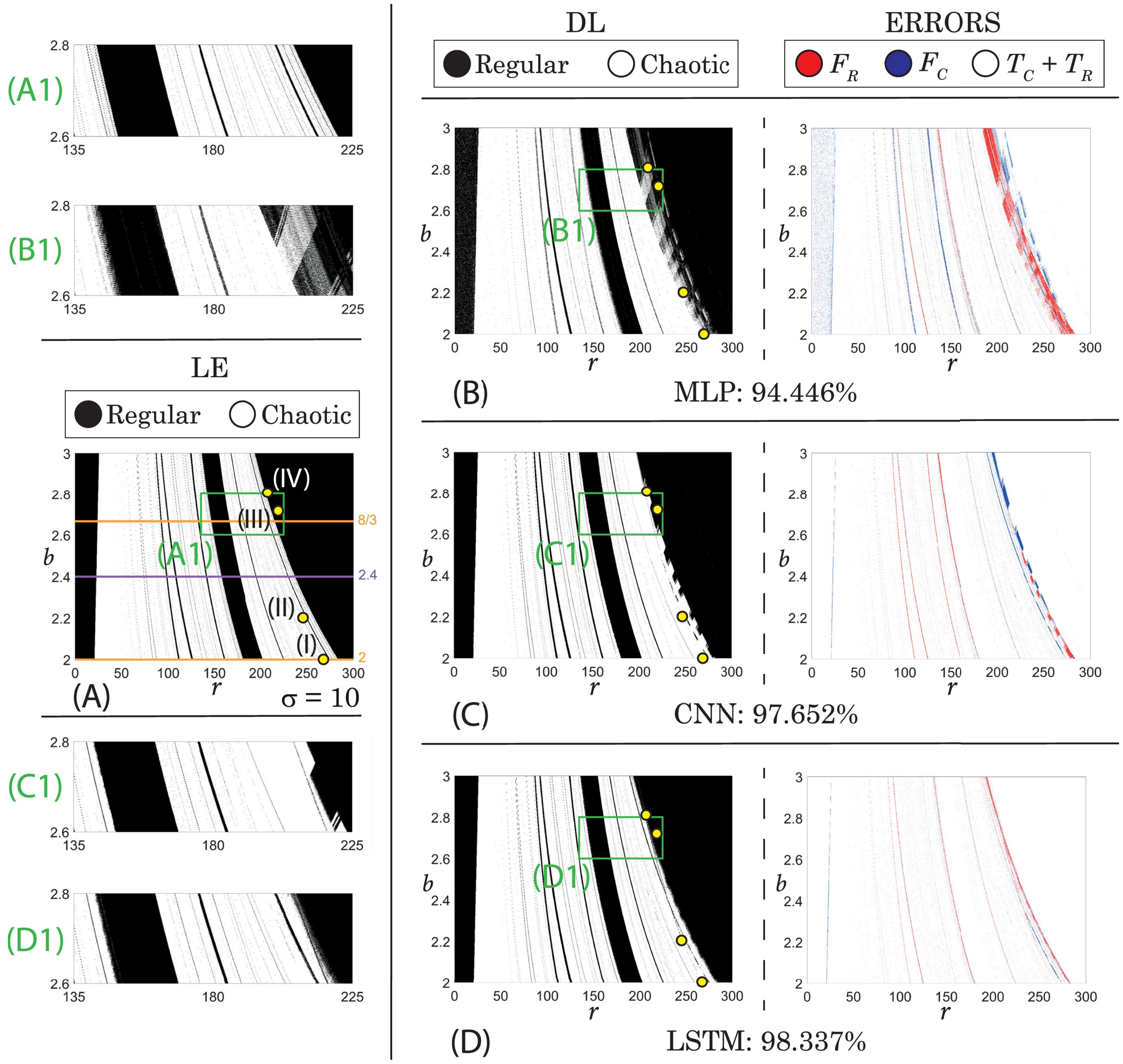


FIG. 11. Chaos detection for the Lorenz system in the biparametric plane (r, b) with $\sigma = 10$. To obtain the plot of panel A, the first approximated LE is used to determine the regular (black) and chaotic (white) regions. The r-parametric lines used to create train ($b \in \{2, 8/3\}$) and validation dataset ($b = 2.4$) are shown in orange and purple, respectively. Panels B, C, and D show the results of the MLP, the CNN, and the LSTM networks, respectively. On the left, classification of the networks (again, regular in black and chaotic in white); on the right, errors committed by the architectures (red for false regulars, blue for false chaotics). The percentage corresponds to the accuracy of the corresponding DL technique. In panels A1, B1, C1, and D1, a zoom of the green framed region in the biparametric plane has been represented for each method. The yellow points correspond to orbits in Fig. 14.

In Fig. 9, we have a histogram that represents the percentage of correctly detected time series in the r-parametric line according to the value of the LE (100% means that all the samples whose first approximated LE is in the corresponding interval have been correctly classified). Blue, pink, and green bars correspond to MLP, CNN, and LSTM networks, respectively. Note that when the first LE is far from 0, the CNN and the LSTM networks perform the task perfectly, and the MLP fails for negative values. Otherwise, the

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

TABLE VI. Approximated time (in seconds) needed for the chaos detection task in the Lorenz system test problem (from data creation to 1D, 2D, and 3D parametric analyses for each DL network). The pure chaos detection DL process, once the data are ready, is just the `Detection (CUDA with PyTorch)` time, pointed with a club symbol. The last three rows correspond to the time used by the classical technique of Lyapunov exponents.

	MLP	CNN	LSTM
Creation of each raw dataset (CPU with parallel computing)	419	419	419
Data selection of train, validation, and test sets (CPU)	882.586	882.586	882.586
Normalization of train, validation, and test sets (CPU)	15.833	15.833	15.833
Training (CUDA with PyTorch)	2171.203	2154.514	8601.143
Test (CUDA with PyTorch)	0.158	0.208	0.461
One-parameter line. Create data (CPU with parallel computing)	1.571	1.571	1.571
One-parameter line. Prepare data (CPU)	8.348	9.427	9.474
♣ **One-parameter line. Detection** (CUDA with PyTorch)	0.012	0.016	0.072
Biparametric plane. Create data (CPU with parallel computing)	307.940	307.940	307.940
Biparametric plane. Prepare data (CPU)	1418.089	1636.286	1623.851
♣ **Biparametric plane. Detection** (CUDA with PyTorch)	3.729	37.434	17.630
Triparametric analysis. Create data (CPU with parallel computing)	…	…	14 156.416
Triparametric analysis. Prepare data (CPU)	…	…	28 929.904
♣ **Triparametric analysis. Detection** (CUDA with PyTorch)	…	…	1139.313
One-parameter line. Classical technique LEs (CPU with parallel computing)	419		
Biparametric plane. Classical technique LEs (CPU with parallel computing)	$9 \cdot 10^4$		
Triparametric analysis. Classical technique LEs (CPU with parallel computing)	$1.402\,13 \cdot 10^6$		

percentage of correct detections is lower for all the networks but still larger than 82% in all the cases. Overall, the MLP shows the worst results, and the LSTM gives the best ones in general.

D. Going deeper into the Lorenz dataset. Changing the length

To train the networks for chaos detection in the Logistic map and the Lorenz system, we have fixed the length of the input time series to 1000 (the same as other authors have considered for this task[3]). The goal of this subsection is to show that this length seems to be a good choice for this problem. For example, in the case of the MLP, the length of the input determines the number of neurons in the input layer, so if we change it, we have to modify the architecture that we have set up in PyTorch. For the LSTM, the length can be variable, that is, we can train the same LSTM architecture with different input lengths. For this reason, we consider that it is interesting to compare the performance of our LSTM when it is trained with time series of different lengths (it tells us how much information is needed by the built LSTM network to learn the task).

We perform the experiment (with the LSTM architecture) for length 500 and 1500, comparing with the current results for length 1000. To obtain the new datasets we have carried out the same integration steps as in the case of length 1000, computing the same number of points but storing the last 500 or 1500. As an example, in Fig. 10, orbits I and V of Fig. 14 are represented with aforementioned lengths.

In Table V, we have the results for three different lengths (information of length 1000 corresponds to Table III). Even though the change in the mean of the loss and the mean of the accuracy is not very considerable, length 1000 gives us the lower mean for the loss and the greatest mean for the accuracy in all the datasets. If we focus on the standard deviation, we can see that the case of length 1500 is always the one with the highest values. Cases of length 500 and 1000 present a similar standard deviation in training and validation datasets. One possible answer for not getting an improvement for length 1500 over length 1000 is the restricted memory capacity of the LSTM recursive structures, which could be affected by long observation sequences and result in the loss of initial evidence. Note that dynamically both datasets provide similar information as it can be seen in Fig. 10, unlike the case of using length 500 where the information is not complete. Therefore, we conclude that length 1000 is a good choice to perform the chaos detection task in our problem.

E. 2D and 3D parametric diagrams

In a dynamical study of a mathematical model, it is very interesting to perform 2D and 3D parametric analyses. In the case of the Lorenz system, several 2D studies have been presented in the literature.[1,2] Therefore, a challenging problem is to see the behavior of DL networks for 2D and even 3D parametric studies.

Just three r-parametric lines on the biparametric plane (r, b) have been involved in the training process of DL networks (see orange and purple lines in panel A of Fig. 11). As shown in Sec. IV C, these trained networks have performed correctly the chaos detection task in a different r-parametric line of the aforementioned plane. Now, we show that they are able to generalize to the whole biparametric plane (r, b) with accurate results. In particular, a biparametric plane (r, b) with $\sigma = 10$, $r \in [0, 300]$, and $b \in [2, 3]$ is studied. It is important to remark that with the data of few lines, DL networks are able to analyze correctly complete biparametric planes.

The use of DL techniques makes the biparametric analysis faster: with the classical technique (LEs), we need approximately 25 h to obtain the biparametric plane of Fig. 11(a) (of 1000 × 1000

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

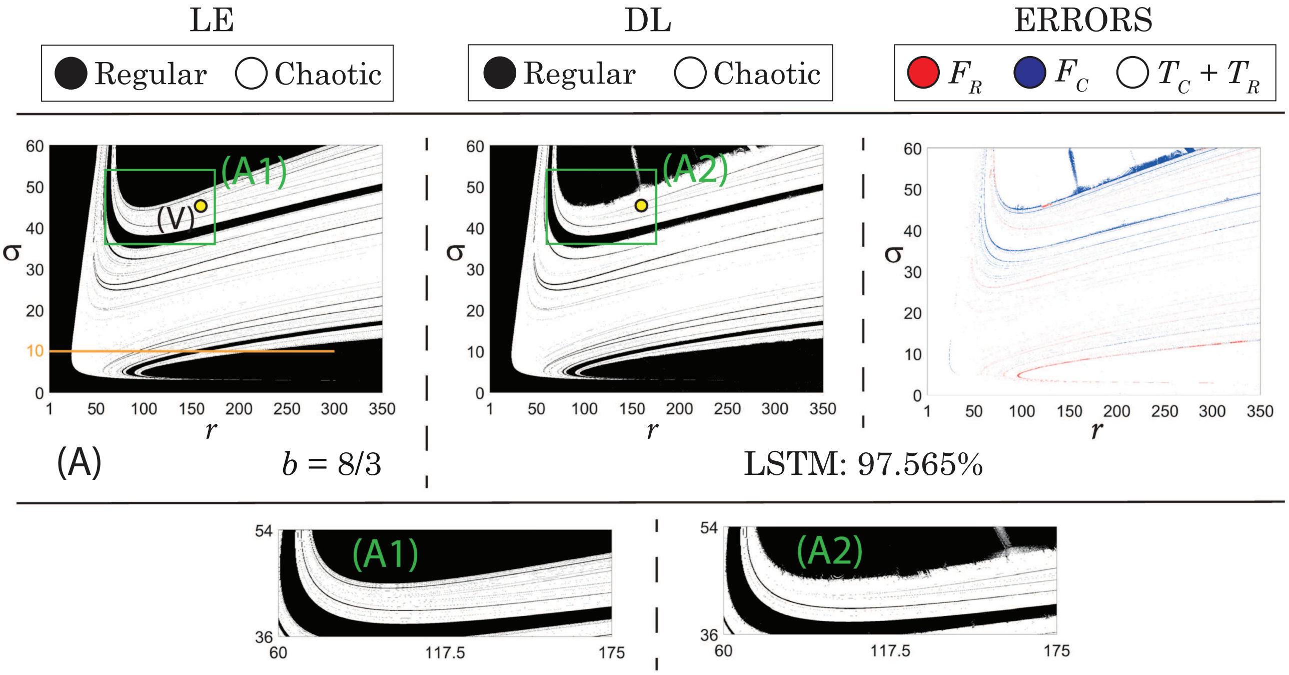


FIG. 12. In panel A, chaos detection for the Lorenz system in the biparametric plane (r, σ) with $b = 8/3$. From left to right: results obtained with LEs, detections with the trained LSTM network, and errors committed by this architecture (same color code than in Fig. 11 is used to represent regular and chaotic behavior, and false regular and false chaotic detections). The percentage corresponds to the accuracy of the LSTM for the chaos detection task. The orange horizontal line shows the samples present in the raw datasets used to create the training set. In panels A1 and A2, a zoom of the green framed region of the biparametric plane has been represented using the classical and DL techniques. The yellow point corresponds to an orbit in Fig. 14.

21 July 2023 16:30:37

points) and with DL (in the same machine) around 30 min are necessary to generate it (see Table VI for more details). Note that LEs usually take a long time to converge to their correct value.

In Fig. 11, we have the study of regular and chaotic behavior in the biparametric plane (r, b) with $\sigma = 10$ of the Lorenz system. In panel A, classification is performed according to the first approximated LE (white for chaotic and black for regular). On the left side of panels B, C, and D, chaos detection with DL architectures (MLP, CNN, and LSTM, respectively) is depicted (same code of colors of panel A). On the right side of such panels, we have the errors committed by networks (we have compared with the biparametric plot obtained with LEs in panel A). In particular, chaotic samples wrongly classified (false regular, F_R) are in red, misclassified regular samples (false chaotic, F_C) are in blue, and samples detected correctly (true regular, T_R, and true chaotic, T_C) are in white. Moreover, for each technique, we have indicated the accuracy (percentage of correct detections) that is always greater than 94% and, as expected, it is bigger for CNN and LSTM networks.

As it can be seen in error plots, most of the incorrect detections correspond to the right boundary between chaotic and regular regimes (in the case of the MLP, a lot of noise can also be seen for r small where there are F_C detections). The F_R detections stand out in this right zone for MLP and LSTM cases, and F_C results highlight for the CNN, being the LSTM the network with the lower number of errors. This right boundary has to be studied carefully with whatever technique used for chaos detection as a dynamical behavior known as transient chaos[8] occurs. In panels A1, B1, C1, and D1, a zoom of the green framed region that includes part of this boundary is depicted for each technique.

The three DL architectures are able to study the behavior of the Lorenz system in the biparametric plane (r, b) where the three r-parametric lines used during the training process are included [see orange and purple lines in Fig. 11(a)]. Now, we show that DL is also able to reconstruct other biparametric planes of parameter space. In particular, we compute the biparametric plane (r, σ) for $b = 8/3$, $r \in [1, 350]$ and $\sigma \in [0, 60]$; the (σ, b) region for $r = 28$, $\sigma \in [0, 40]$ and $b \in [0, 4]$; and the (σ, b) plane for $r = 500$, $\sigma \in [0, 800]$ and $b \in [0, 100]$. To perform this task, we use the LSTM as it gives the best results in the plane (r, b) (see Fig. 11).

In panel A of Fig. 12, we have, from left to right, the biparametric plane (r, σ) obtained with the approximated LEs, the results of the trained LSTM, and the errors committed by this network. Same code of colors as in Fig. 11 is used. The boundaries between both dynamical regimes are well-defined by the LSTM, only the top boundary is noisy (see panel A2 for a zoom of the green framed region where it is included, and panel A1 for the corresponding results of LEs), and F_C detections stand out. The percentage of correct detections (accuracy) is greater than 97%. Notice that in

Published under an exclusive license by AIP Publishing

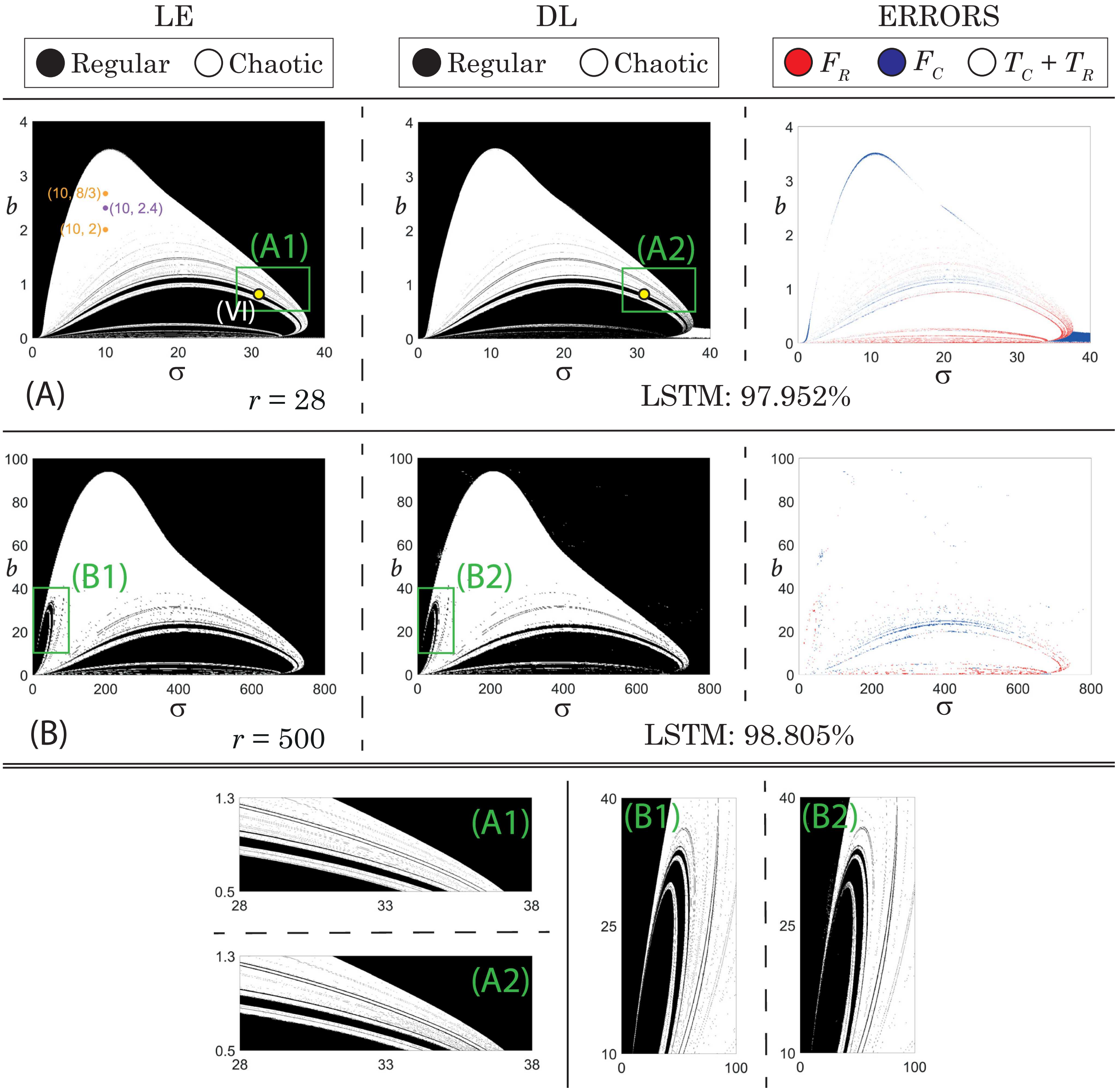


FIG. 13. In panel A, chaos detection for the Lorenz system in the biparametric plane (σ, b) with $r = 28$. In panel B, chaos detection for the Lorenz system in the biparametric plane (σ, b) with $r = 500$. From left to right: results obtained with LEs, detections with the trained LSTM network, and errors committed by this architecture (same color code than in Fig. 11 is used to represent regular and chaotic behavior, and false regular and false chaotic detections). The percentage given in each panel corresponds to the accuracy of the LSTM for such biparametric plane. The orange and purple points show the samples present in the raw datasets used to create the training and validation sets. In panels A1, A2, B1, and B2, a zoom of the green framed region of the biparametric plane has been represented using the classical and DL techniques. The yellow point corresponds to an orbit in Fig. 14.

this case, where the chaotic detection is almost perfect, only the samples in the orange segment belong to the raw datasets used to create the training set. In panel A of Fig. 13, we have the study of the biparametric plane (σ, b) with $r = 28$. Again, we can see that this network is able to define correctly the boundary between both dynamical regimes. It highlights the bottom right corner, a regular zone that has been defined as chaotic by the LSTM. In panel A2 of Fig. 13, that corresponds to the green framed region, it is shown that

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

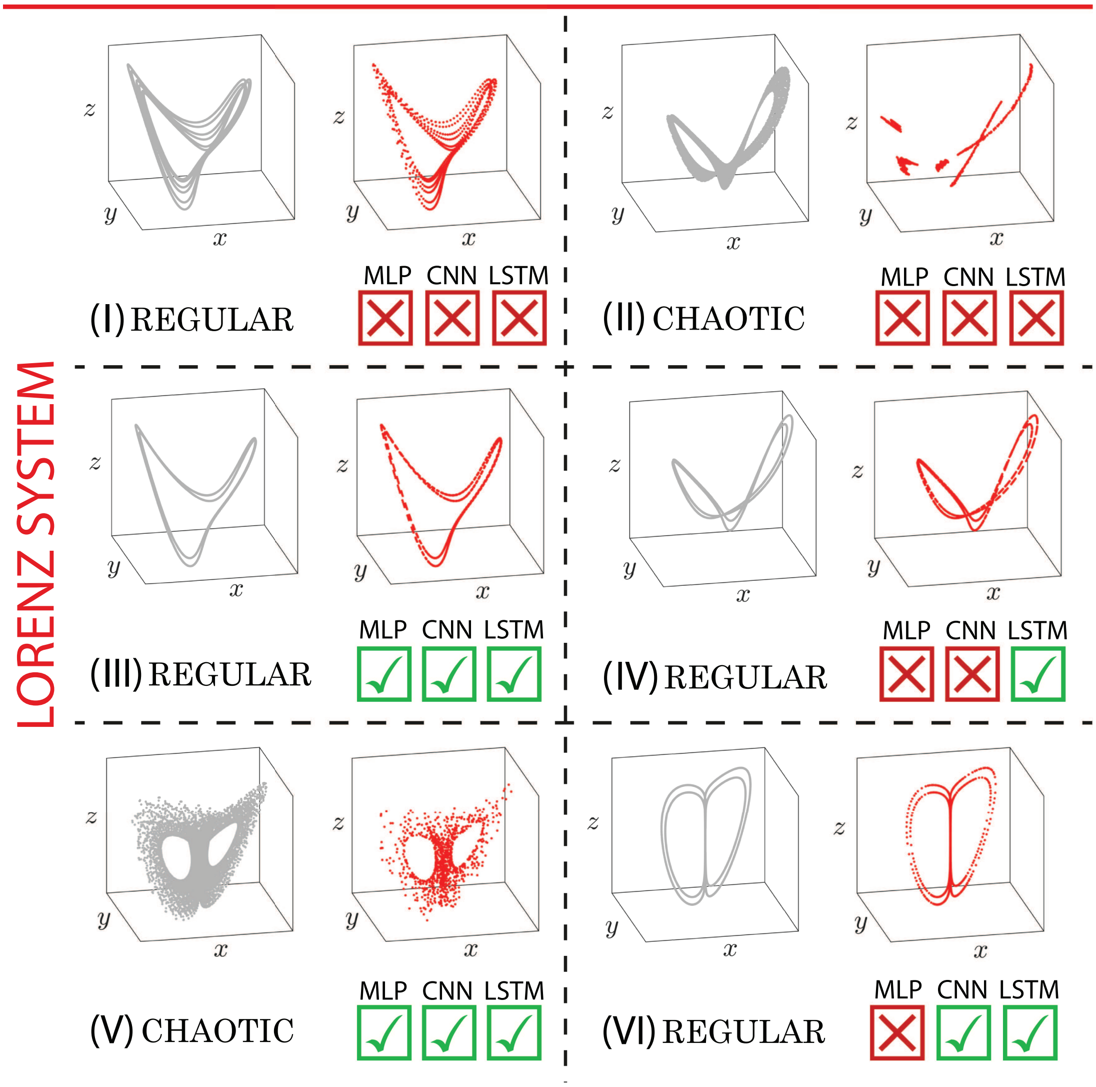


FIG. 14. Some orbits (in gray) of the Lorenz system with their true dynamical behavior, the corresponding sample (in red) used by the networks, and the correctness or incorrectness of the classification made by them. In all the cases, $(x_0, y_0, z_0) = (1, 1, 1)$. The values of the parameters are (i) $(\sigma, r, b) = (10, 267.84464, 2)$, (ii) $(\sigma, r, b) = (10, 246.3910675, 2.2)$, (iii) $(\sigma, r, b) = (10, 220, 2.71)$, (iv) $(\sigma, r, b) = (10, 208.3006, 2.809)$, (v) $(\sigma, r, b) = (45.3, 160, 8/3)$, and (vi) $(\sigma, r, b) = (31, 28, 0.8)$. See yellow points in Figs. 11, 12, and 13.

the trained LSTM performs the task with high precision (panel A1 corresponds to LEs). In fact, the percentage of correct detections in the whole plane is almost 98%. In this case, only three points are in the raw datasets used to create train and validation sets, which shows the high generalization ability of the trained LSTM. In panel B of Fig. 13, the biparametric analysis (σ, b) with $r = 500$ is depicted. As in the other panels, the trained LSTM network distinguishes correctly both dynamical regimes, as it can be seen in panel B2 that is a zoom of the green framed region (see panel B1 for detection performed with LEs). The accuracy of this study is greater than 98%. Most of the errors of the network correspond to small values of b. Notice that no point of this parametric region has been used to train (or validate) the network. With the results of Figs. 12 and 13, we can conclude that DL allows us to study regions of parameter space where training and validation points are almost not present [Figs. 12(a) and 13(a)] or are not present at all [Fig. 13(b)].

Analogously to what we did in the case of the Logistic map, we analyze in Fig. 14 some correct and incorrect classifications of DL networks (see yellow dots in Figs. 11–13). In panel I of Fig. 14, we show a periodic orbit (regular behavior) that DL networks have not

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

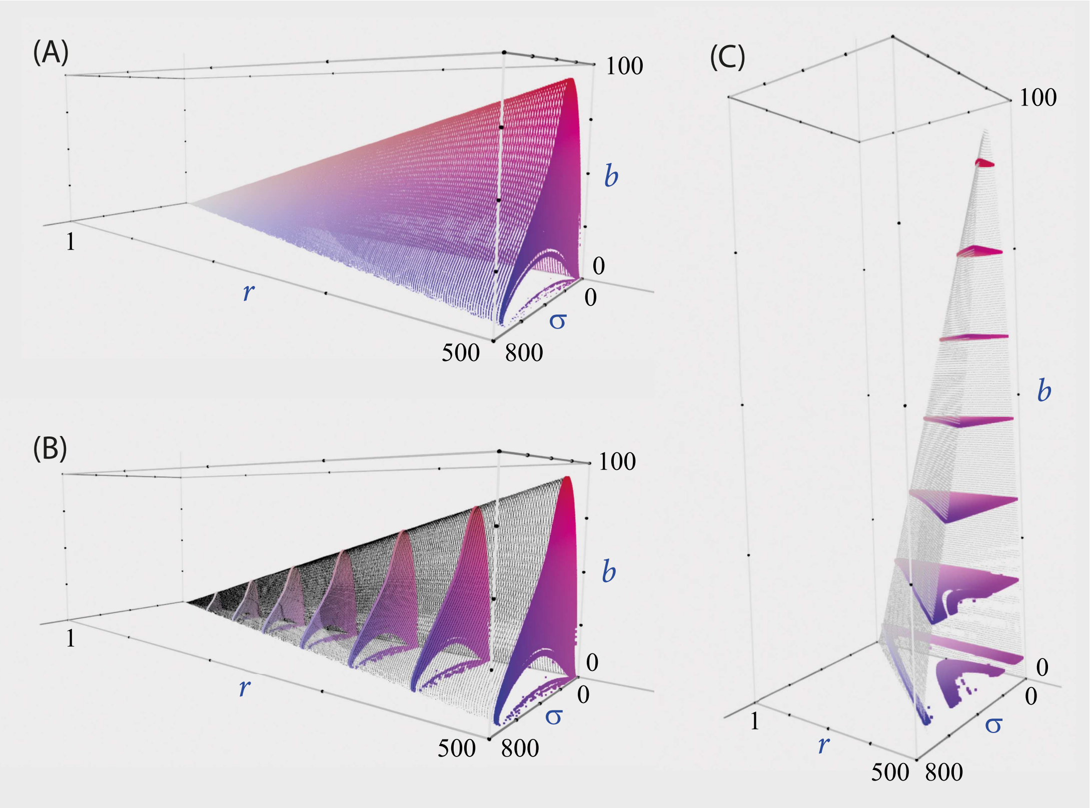


FIG. 15. Representations obtained from the dense triparametric analysis of the Lorenz system performed with the LSTM network. See Integral Multimedia (Multimedia view) for the dense study. In panel A, general view of the upper boundary between both dynamical regimes and chaotic region for $r = 500$. In panels B and C, some 2D vertical and horizontal planes in the triparametric space with chaotic detections in color and the external boundary between both dynamical regimes in shaded black. The colors used have been simply selected for ease of viewing. Multimedia available online.

21 July 2023 16:30:37

been able to classify adequately. The dynamical explanation is due to the nature of this orbit, it is a periodic orbit with a large number of space-expanding loops and a large period. The sample in panel II is chaotic, and the networks have also failed to classify it properly. Now the explanation is due to the kind of chaos of this orbit. The orbit changes slowly in space, being different loops very similar each other (see the 3D plot of the DL data). All the DL techniques fail to identify the kind of behavior of orbits I and II, but there are clear reasons for the fail, and in fact any short time methodology would fail on these orbits. Besides, both cases are in the limit between the regular and chaotic dynamics with a lot of bifurcations located in that area. In contrast, all networks have managed to properly identify the type of behavior of orbits in panels III and V, regular and chaotic, respectively. As an intermediate situation, we show the regular orbits IV and VI for which only LSTM in the first case, and CNN and LSTM in the second have been able to give a correct classification. The complicated points have been selected on the basis of the previous analyses performed on the biparametric planes of Figs. 11–13.

As a final and challenging task, we study a 3D parametric space region of the Lorenz system (allowing the three parameters to change) with the LSTM trained and validated using only three r-parametric segments (see orange and purple lines in Fig. 11). Note that with the classical technique of LEs, this kind of analysis would be highly computationally demanding. Taking into account the time that we need to compute a 2D diagram with LEs, the expected time for the 3D study in this subsection (with $250 \times 250 \times 250$ points) with this classical technique would be of 16 days approximately; with DL, we need around 12 h (see Table VI). We also remark that, up to the knowledge of the authors, there is no dense 3D analysis in the literature for the Lorenz system. In Refs. 1 and 2, the authors studied several 2D planes to reconstruct a 3D figure, but it is not a complete dense 3D plot as in the present work. Again, we remark that we use just a very few lines of data to train and generate a network that is able to completely perform a 3D parametric study. Therefore, these techniques open a window for dense 3D parametric studies in a reasonable time.

The 3D study is performed for $\sigma \in [0, 800]$, $r \in [1, 500]$ and $b \in [0, 100]$, taking 250 values for each parameter. In panel A of Fig. 15, we have used the detections of the trained LSTM network to illustrate a 3D parametric study of the Lorenz system. In particular, for a better representation, we have drawn the boundary

Published under an exclusive license by AIP Publishing

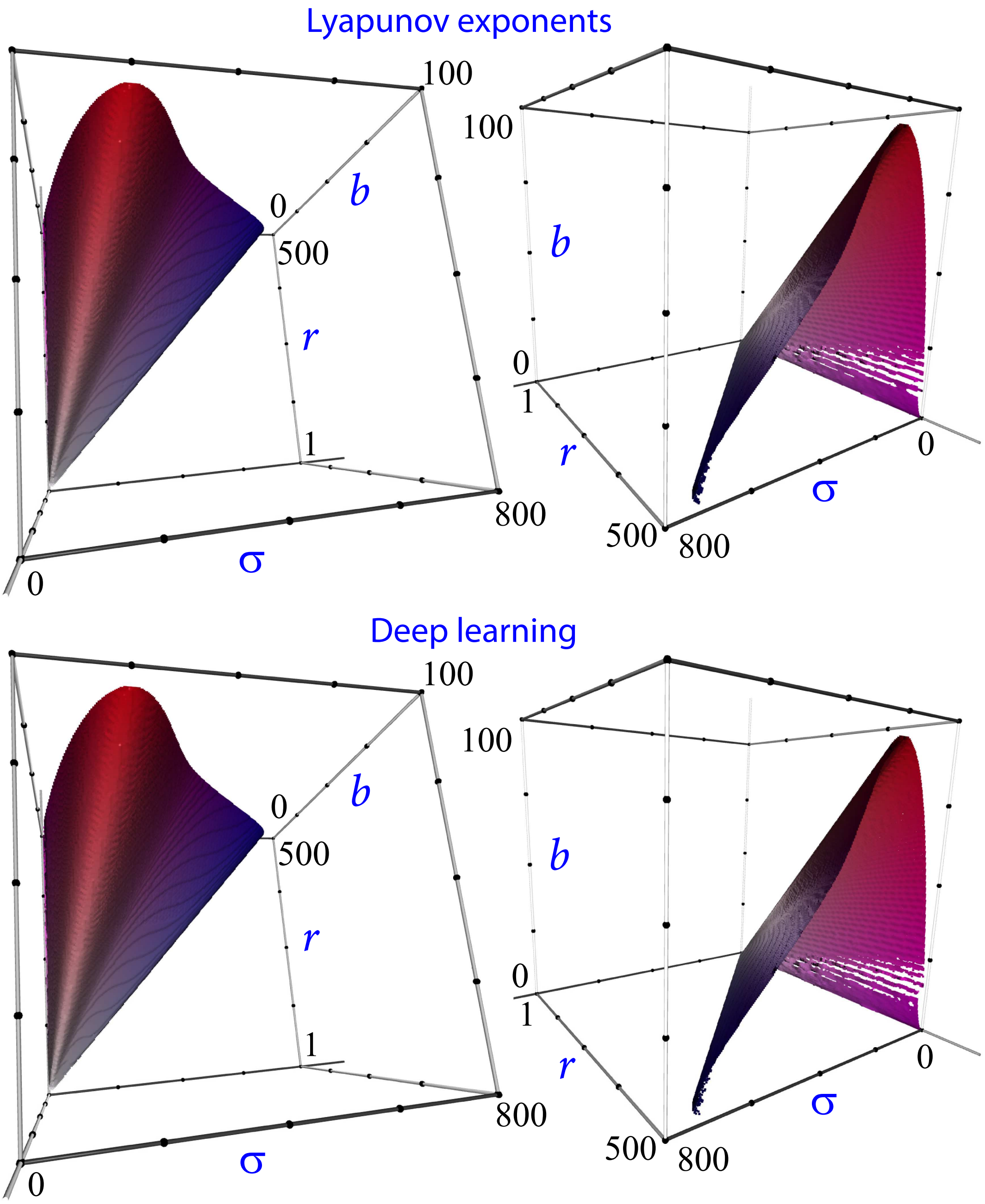


FIG. 16. Detail of the boundary between regular and chaotic regions from a dense 3D parametric analysis of the Lorenz system computed using the Lyapunov exponents and with the LSTM network. See Integral Multimedia (Multimedia view) for an integral view and the dense analysis. Multimedia available online.

between both dynamical regimes of the obtained dense 3D DL figure and the chaotic results for $r = 500$. In panels B and C, we have several 2D planes extracted from the 3D study (in shaded black, we have the boundary between both dynamical regimes). For completion, in Fig. 16, we show two different views of the boundary between regular and chaotic regimes detected by the DL network.

Notice that the trained LSTM detects perfectly the boundary (represented in detail in Fig. 16) between both dynamical regimes. In Fig. 16, we have used both methodologies, the Lyapunov exponents

21 July 2023 16:30:37

Published under an exclusive license by AIP Publishing

and the LSTM network, and the resulting figures are indistinguishable. The dense 3D computation using the Lyapunov exponents has detected 1 852 204 set of parameters of chaotic dynamics, whereas the LSTM network 1 855 862, being the intersection 1 827 514 points (the number of false positives is 28 348 and the number of false negatives is 24 690). Moreover, from 2D planes in Fig. 15, it can be inferred that DL is able to detect the changes between regular and chaotic behaviors that are present for small values of parameter b. We conclude that with an LSTM trained (and validated) in a small region of the parameter space ($\sigma = 10$, $r \in (0, 300]$, $b \in \{2, 2.4, 8/3\}$), an accurate dense 3D analysis can be performed.

As a complement, we include in the Integral Multimedia (Multimedia view) of the article a video where we show the dense 3D parametric study that has allowed analyses in Fig. 15 and the chaotic boundary represented in Fig. 16.

To give an idea of the time savings of using DL networks, we show in Table VI the times (in seconds) needed during the whole study of the Lorenz system test problem. The first three rows correspond to the time used to create train, test, and validation datasets, which includes creation of raw datasets, data selection, and normalization. Fourth and fifth rows correspond to training and test processes. To obtain the one-parameter line of Fig. 8, which contains 6000 different values of r, we indicate the time needed to create the data, to prepare it as a network input, and to be classified by the ANN (sixth, seventh, and eighth rows). The whole one-parameter study takes less than 12 s for all the networks. Notice that the pure chaos detection DL process (if data are given) corresponds to row 8 (club symbol) and is less than one-tenth of a second. With classical techniques (third-to-last row), needed time is around 7 min. In the case of biparametric planes, as these of Figs. 11–13 with 1000 × 1000 points, the whole process (rows 9–11) takes approximately 30 min for all the networks, with pure chaos detection (row 11 with club symbol) of less than 40 s. Time used by classical techniques is given in the second-to-last row, and it is of 25 h. Finally, the triparametric study of Figs. 15 and 16 with 250 × 250 × 250 points takes around 12 h for the LSTM network (see rows 12–14). In this case, the pure chaos detection process (row 14 with a club symbol) is less than 20 min. In the last row of the table, we have the time needed to perform the triparametric analysis with classical techniques. Notice that this analysis with DL takes approximately 12 h, but we would need more than 16 days if classical techniques were used.

V. DISCUSSION AND CONCLUSIONS

In this paper, three well-known Deep Learning networks (MLP, CNN, and LSTM) have been built and trained to carry out the chaos detection task in two test problems: the Logistic map (discrete dynamical system) and the Lorenz system (continuous dynamical system). Usually, time-consuming computational techniques, such as Lyapunov exponents, are used to detect chaos.

All the trained networks have given good results, achieving all of them an accuracy greater than 90%. Moreover, in all the cases, the accuracy of detection of both dynamical regimes (regular and chaotic) is balanced. In the case of the Logistic map test problem, the accuracy for the three networks is very similar. However, in the case of the Lorenz system, the CNN and the LSTM give better results than the MLP. This makes sense since CNNs and LSTM networks take into account spatial and temporal information, respectively.

For the Lorenz system, we have used the trained networks to study the behavior of an r-parametric line and several biparametric planes from different regions of the parameter space. In addition, a dense triparametric plot of the Lorenz system has also been obtained using a trained LSTM network. Up to the knowledge of the authors, this had not been achieved previously with classical or DL techniques. We highlight that the training process used just a few lines of one-parameter data to create a network capable of studying biparametric and even complete triparametric spaces. This is a remarkable result that shows us the power of DL techniques in dynamical system studies.

In this article, we focus on using short time series as data, but longer sequences will give us more complete information, particularly in difficult areas where the LE value is close to zero. To achieve better results, in future research, we will explore the development of improved architectures that are better suited to handle longer sequences or carefully select more challenging training examples to reduce error rates. However, it is important to balance this against the computational cost, as our current focus has been to keep the computational cost low compared to traditional calculation of the LEs.

We conclude that Deep Learning can be used to analyze the behavior (regular and chaotic) of a dynamical system. Our results show that even dense 3D parametric studies can be carried out in a very reasonable time using data from just 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 system tasks.

21 July 2023 16:30:37

ACKNOWLEDGMENTS

R.B., A.L., A.M.C., C.M.C., and S.S. have been supported by the Spanish Research projects PGC2018-096026-B-I00 and PID2021-122961NB-I00. R.B. and S.S. have been supported by the European Regional Development Fund and Diputación General de Aragón (E24-20R and LMP94-21). A.L. has been supported by Diputación General de Aragón project E22-20R. A.M.C. has been supported by Diputación General de Aragón with a PhD student grant. C.M.C. has been supported by Ministerio de Universidades de España with an FPU grant (FPU20/04039). A.M. and A.O. have been supported by Diputación General de Aragón (Grant Group T36-23R: Vivolab). R.V. has been supported by the Spanish Research Project PID2021-122961NB-I00.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Roberto Barrio: Conceptualization (equal); Methodology (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal). **Álvaro Lozano:** Conceptualization (equal); Methodology (equal); Software (equal); Supervision (equal);

Published under an exclusive license by AIP Publishing

Writing – original draft (equal); Writing – review & editing (equal). **Ana Mayora-Cebollero:** Conceptualization (equal); Methodology (equal); Software (equal); Writing – original draft (equal); Writing – review & editing (equal). **Carmen Mayora-Cebollero:** Conceptualization (equal); Methodology (equal); Software (equal); Writing – original draft (equal); Writing – review & editing (equal). **Antonio Miguel:** Conceptualization (equal); Methodology (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal). **Alfonso Ortega:** Conceptualization (equal); Methodology (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal). **Sergio Serrano:** Conceptualization (equal); Methodology (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal). **Rubén Vigara:** Conceptualization (equal); Methodology (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are openly available in Mendeley Data repository at https://doi.org/10.17632/k4x675k5dm.2, Ref. 21.

REFERENCES

[1] R. Barrio and S. Serrano, "A three-parametric study of the Lorenz model," Physica D **229**(1), 43–51 (2007).

[2] R. Barrio and S. Serrano, "Bounds for the chaotic region in the Lorenz model," Physica D **238**(16), 1615–1624 (2009).

[3] N. Boullé, V. Dallas, Y. Nakatsukasa, and D. Samaddar, "Classification of chaotic time series with deep learning," Physica D **403**, 132261 (2020).

[4] J. Bragard, H. Pleiner, O. J. Suarez, P. Vargas, J. A. C. Gallas, and D. Laroze, "Chaotic dynamics of a magnetic nanoparticle," Phys. Rev. E **84**(3), 037202 (2011).

[5] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," arXiv:2104.13478 (2021).

[6] A. Celletti, C. Gales, V. Rodriguez-Fernandez, and M. Vasile, "Classification of regular and chaotic motions in Hamiltonian systems with deep learning," Sci. Rep. **12**(1), 1–12 (2022).

[7] R. Chandra, S. Goyal, and R. Gupta, "Evaluation of deep learning models for multi-step ahead time series prediction," IEEE Access **9**, 83105–83123 (2021).

[8] E. J. Doedel, B. Krauskopf, and H. M. Osinga, "Global invariant manifolds in the transition to preturbulence in the Lorenz system," Indagationes Math. **22**(3-4), 222–240 (2011).

[9] A. Géron, *Hands-On Machine Learning with Scikit-Learn & TensorFlow* (O'Reilly Media Inc., Sebastopol, 2019).

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 770–778.

[11] C. F. Higham and D. J. Higham, "Deep learning: An introduction for applied mathematicians," SIAM Rev. **61**(4), 860–891 (2019).

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**(8), 1735–1780 (1997).

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE* (IEEE, 1998), Vol. 86, pp. 2278–2324.

[14] W. S. Lee and S. Flach, "Deep learning of chaos classification," Mach. Learn.: Sci. Technol. **1**(4), 045019 (2020).

[15] E. N. Lorenz, "Deterministic nonperiodic flow," J. Atmos. Sci. **20**(2), 130–141 (1963).

[16] R. M. May, "Simple mathematical models with very complicated dynamics," Nature **261**(5560), 459–467 (1976).

[17] 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, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., 2019), pp. 8024–8035.

[18] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI Press, 2019), Vol. 33, pp. 4780–4789.

[19] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," Neural Netw. **126**, 191–217 (2020).

[20] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," Physica D **16**(3), 285–317 (1985).

[21] R. Barrio, Á. Lozano Rojo, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara (2023). "Data of 'Deep Learning for Chaos Detection'", Mendeley Data, V2.

21 July 2023 16:30:37

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
0167-2789/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

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

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

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]:

$$\left\{\begin{array}{lcl} \dot{x}_1 & = & \sigma(y_1-x_1), \\ \dot{y}_1 & = & -x_1z_1+r_1x_1-y_1+\lambda_1(x_2-y_2), \\ \dot{z}_1 & = & x_1y_1-bz_1, \\ \dot{x}_2 & = & \sigma(y_2-x_2), \\ \dot{y}_2 & = & -x_2z_2+r_2x_2-y_2+\lambda_2(x_1-y_1), \\ \dot{z}_2 & = & x_2y_2-bz_2, \end{array}\right. \tag{2}$$

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

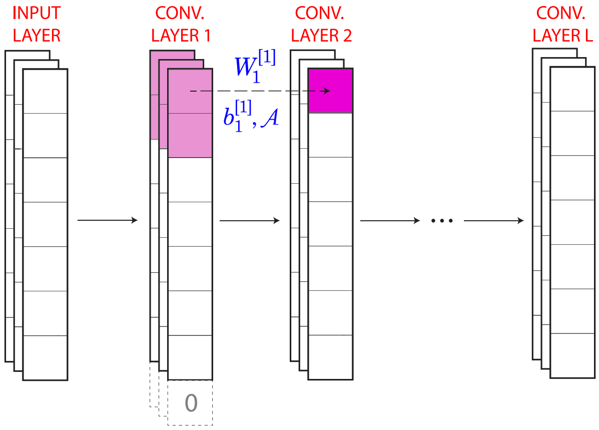


Fig. 1. Schematic representation of a 1D CNN architecture with three channels in the input and depicted convolutional (conv.) layers. Zero-padding is added to convolutional layer 1 to maintain the size for convolutional layer 2 (dashed neurons with a 0 in the middle).

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.
- $W_j^{[l]} = (w_{k,c,j}^{[l]})_{k=1,\dots,K^{[l]};c=1,\dots,C^{[l-1]}}$ 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 ± 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 ± 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 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.

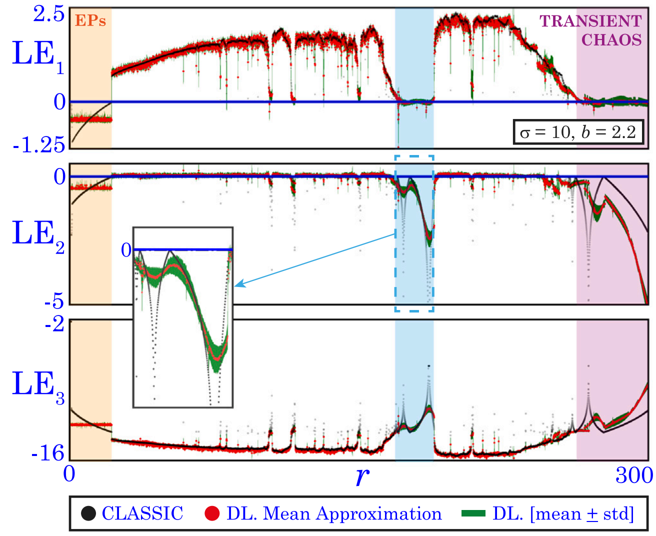


Fig. 2. 1D analysis ($\sigma = 10$, $b = 2.2$) of the Lyapunov exponents in the Lorenz system when training the CNN with non-random data (Huber loss value $[0.091 \pm 0.005]$). The LEs obtained with classical techniques are in black, and the results given by DL are in red (mean value of the predicted 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.

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

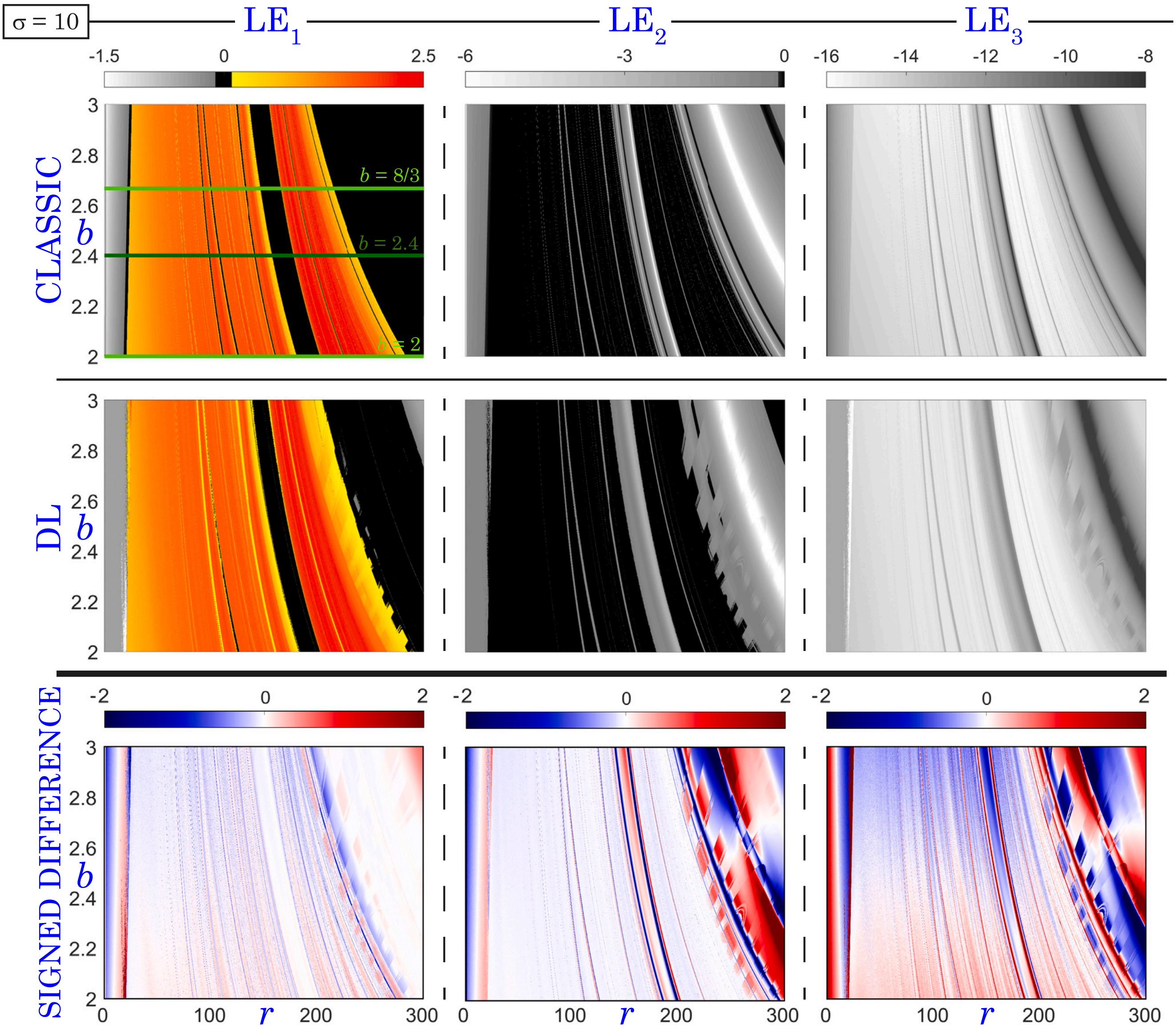


Fig. 3. 2D analysis of Lyapunov exponents in the Lorenz system ($\sigma = 10$) when training with non-random data (Huber loss value $[0.115 \pm 0.005]$). From left to right, LE$_1$, LE$_2$ and LE$_3$ analyses are depicted. From top to bottom, results with classical techniques, with DL techniques, and 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 $\mathrm{LE}_1 \leq -0.4$. 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 $|\mathrm{LE}_1|$ around 0, for more than half of the samples, the error is less than or equal to 0.05. For $\mathrm{LE}_1 > -0.4$ in almost all samples, such prediction error is less than or equal to 0.5. Taking into account that the prediction is performed

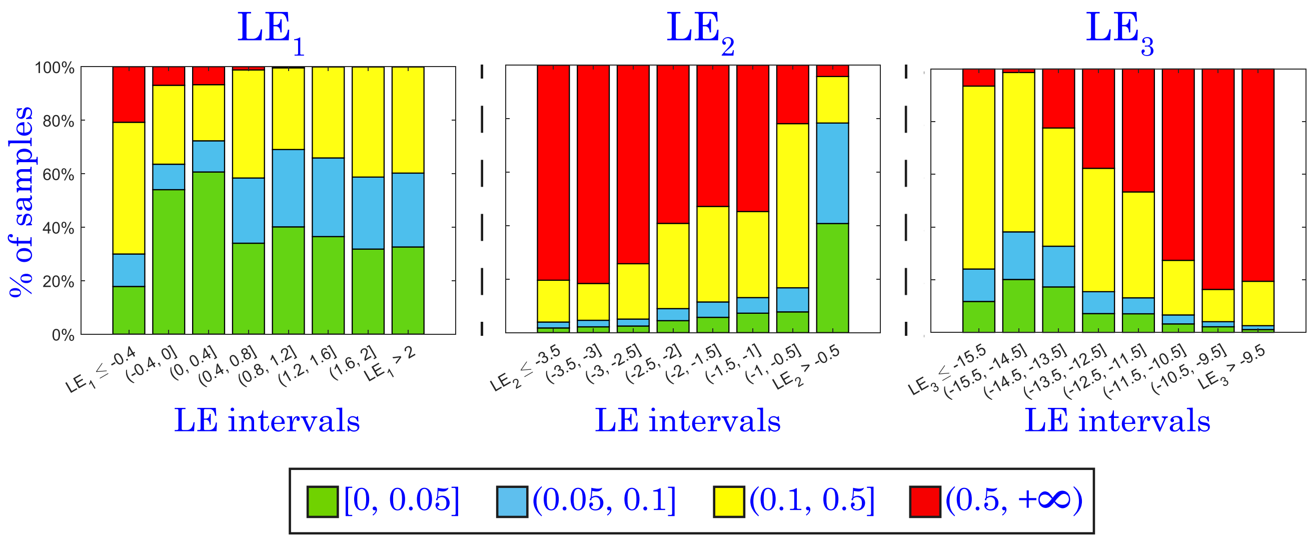


Fig. 4. Error analysis of Lyapunov exponents prediction in the (r, b)-parametric plane (studied in Fig. 3) of the Lorenz system when training with non-random data. From left to right, LE_1, LE_2 and LE_3 results are depicted. The color code is shown at the bottom. (See the text for more details.).

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.

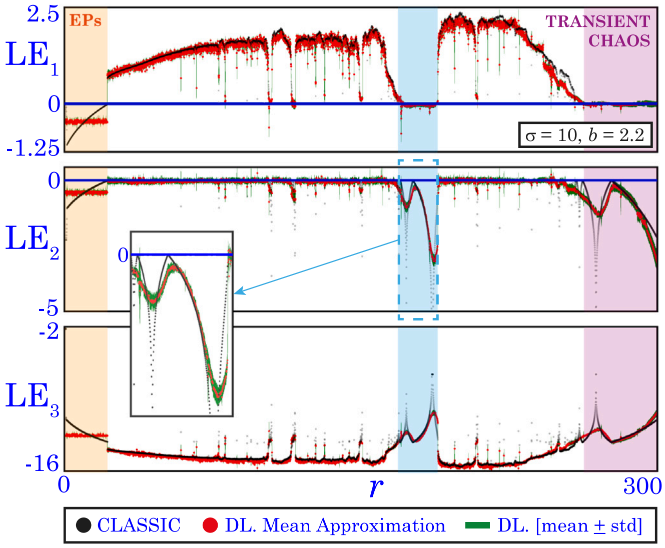


Fig. 5. 1D analysis ($\sigma = 10$, $b = 2.2$) of the Lyapunov exponents in the Lorenz system when training the CNN with random data (Huber loss value $[0.055 \pm 0.003]$). The LEs obtained with classical techniques are in black, and the results given by DL are in red (mean value of the predicted values 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 $\mathrm{LE}_1 \leq -0.4$. It is remarkable that for $|\mathrm{LE}_1|$ 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 $\mathrm{LE}_1 > -0.4$ 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.

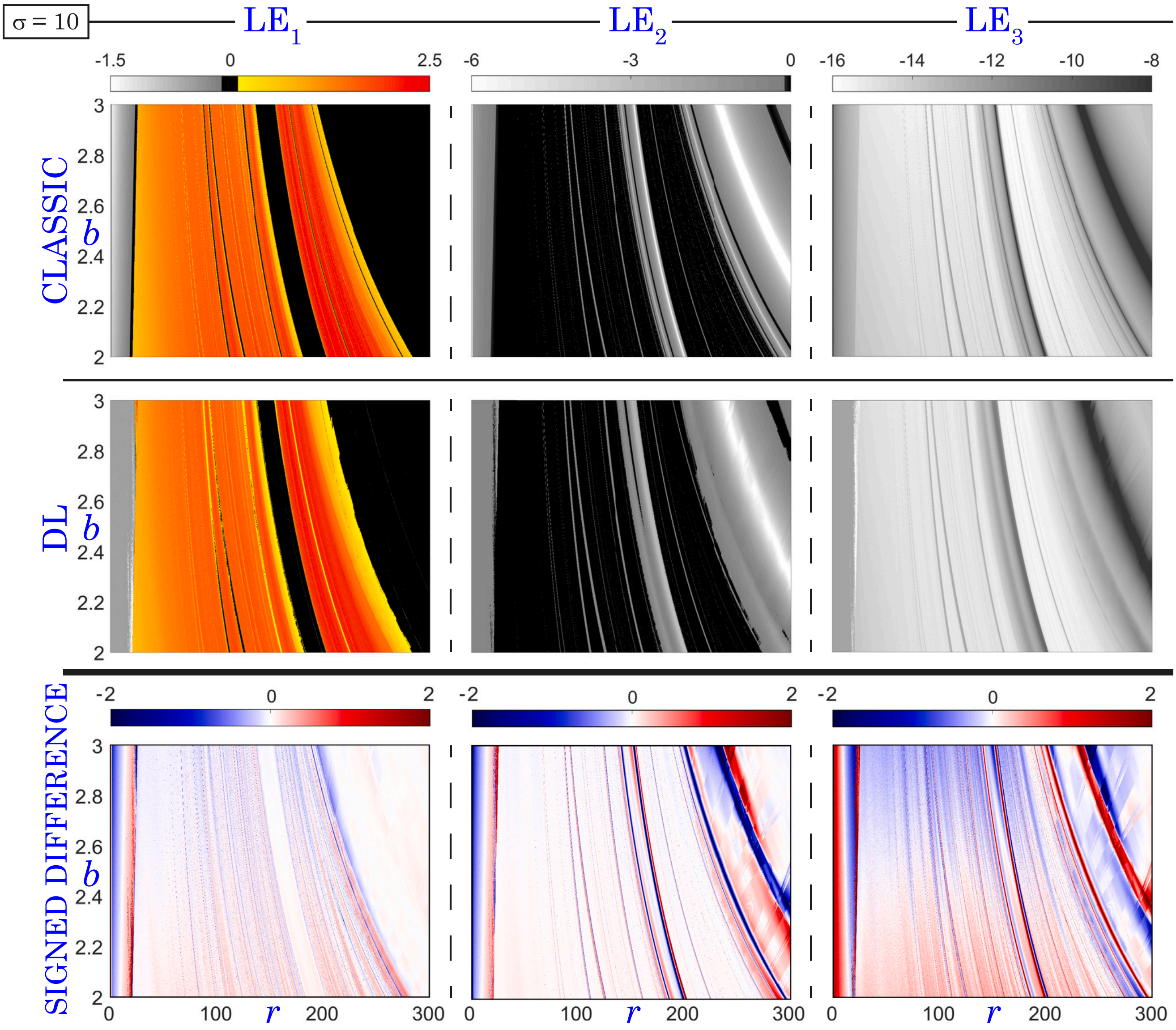


Fig. 6. 2D analysis of Lyapunov exponents in the Lorenz system ($\sigma = 10$) when training with random data (Huber loss value $[0.079 \pm 0.006]$). From left to right, LE$_1$, LE$_2$ and LE$_3$ analyses are depicted. From top to bottom, results with classical techniques, with DL techniques, and signed 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%

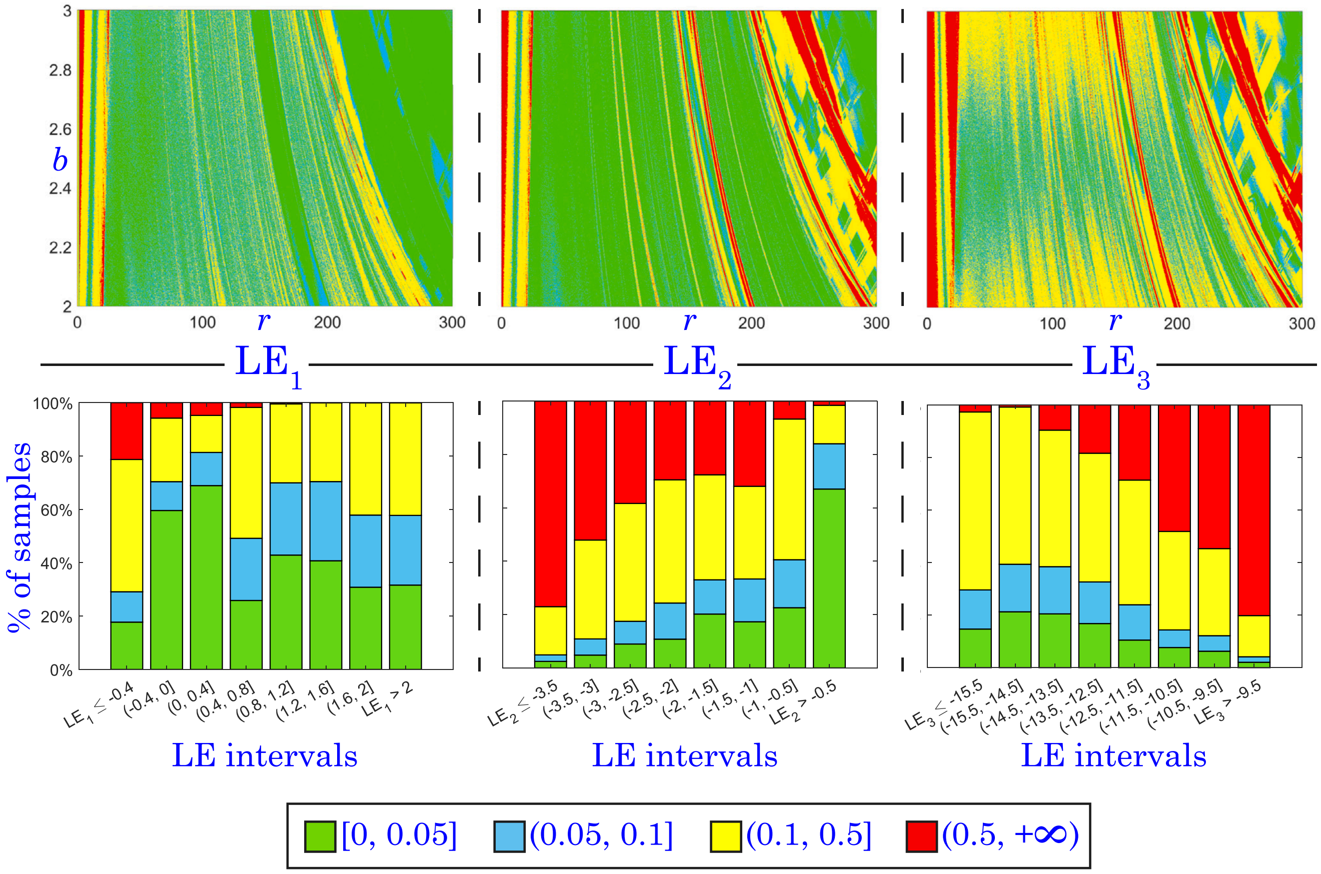


Fig. 7. Error analysis of Lyapunov exponents prediction in the (r, b)-parametric plane (studied in Fig. 6) of the Lorenz system when training with random data. From left to right, LE_1, LE_2 and LE_3 results are depicted. The first row corresponds to the errors represented in the parametric plane. In 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

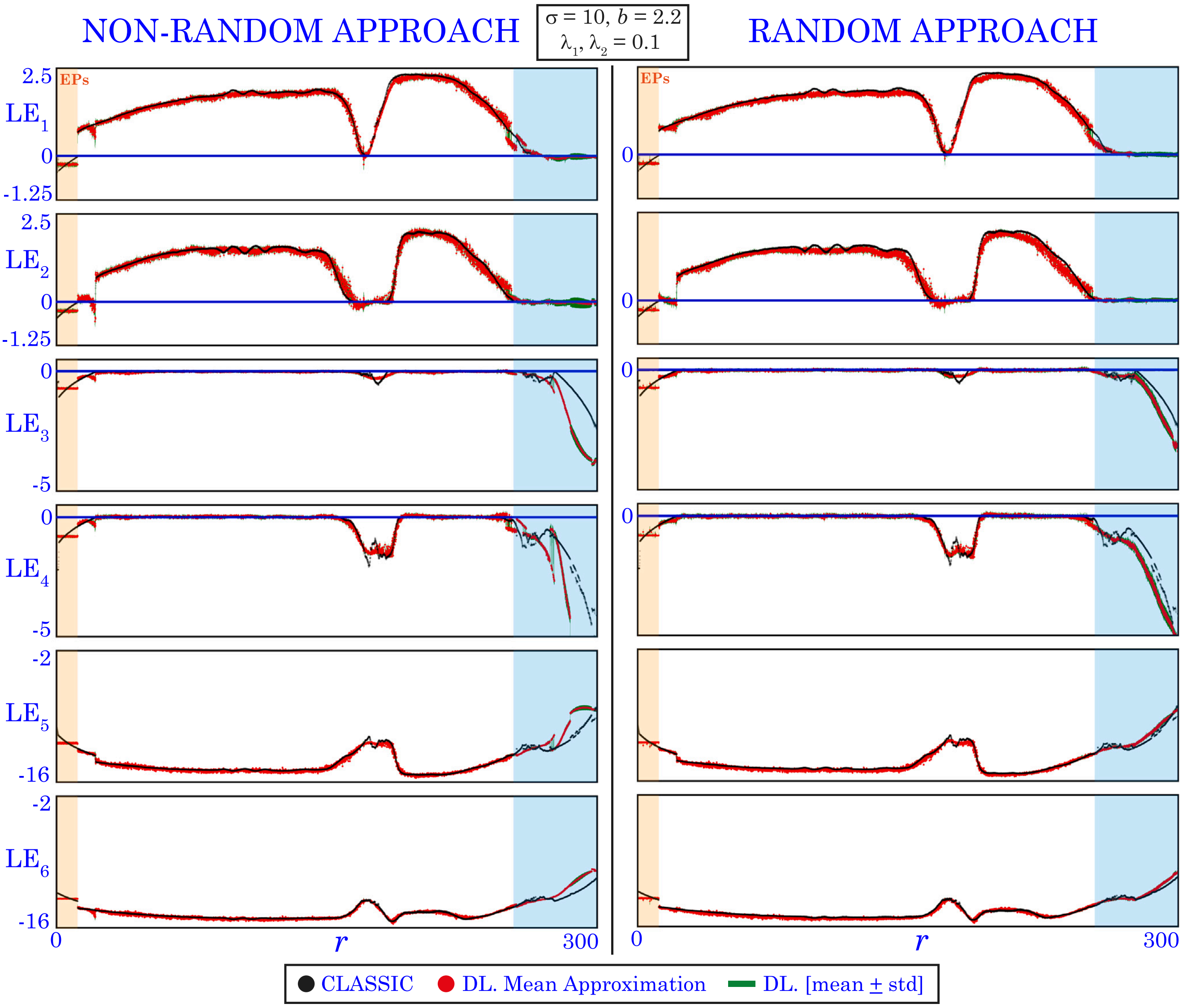


Fig. 8. 1D analysis ($\sigma = 10$, $b = 2.2$, $\lambda_1 = \lambda_2 = 0.1$) of the Lyapunov exponents in the coupled Lorenz system when training the CNN with non-random data (left column) and random data (right column). The Huber loss value in the non-random approach is $[0.274 \pm 0.023]$, and in 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 $\mathrm{LE} \in [-0.1, 0.1]$, 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.

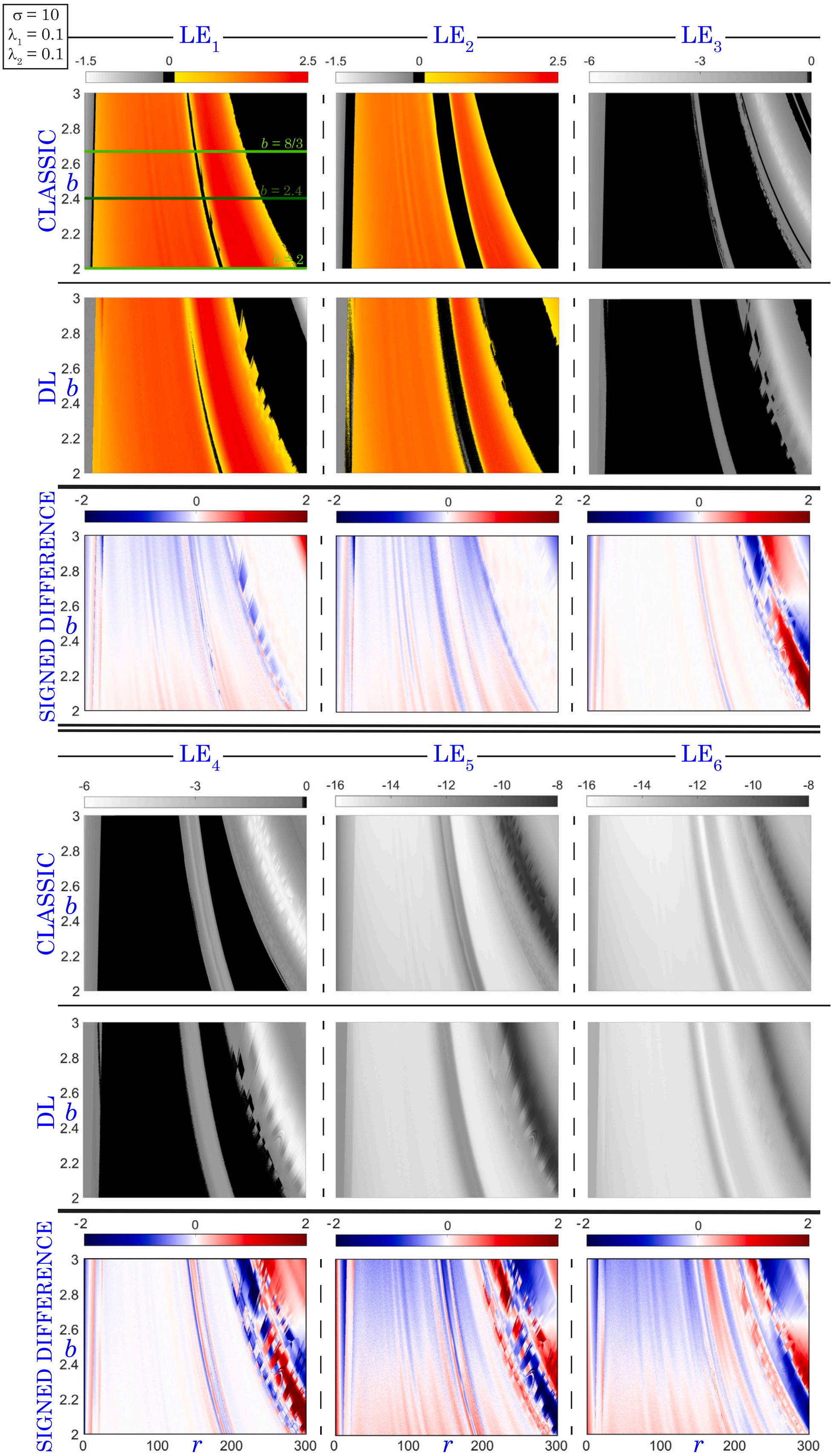


Fig. 9. 2D analysis of Lyapunov exponents in the coupled Lorenz system ($\sigma = 10$, $\lambda_1 = \lambda_2 = 0.1$) when training with non-random data (Huber loss value is $[0.046 \pm 0.002]$). The analyses of the six LEs of the system are depicted for classical and DL techniques. The signed difference 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.).

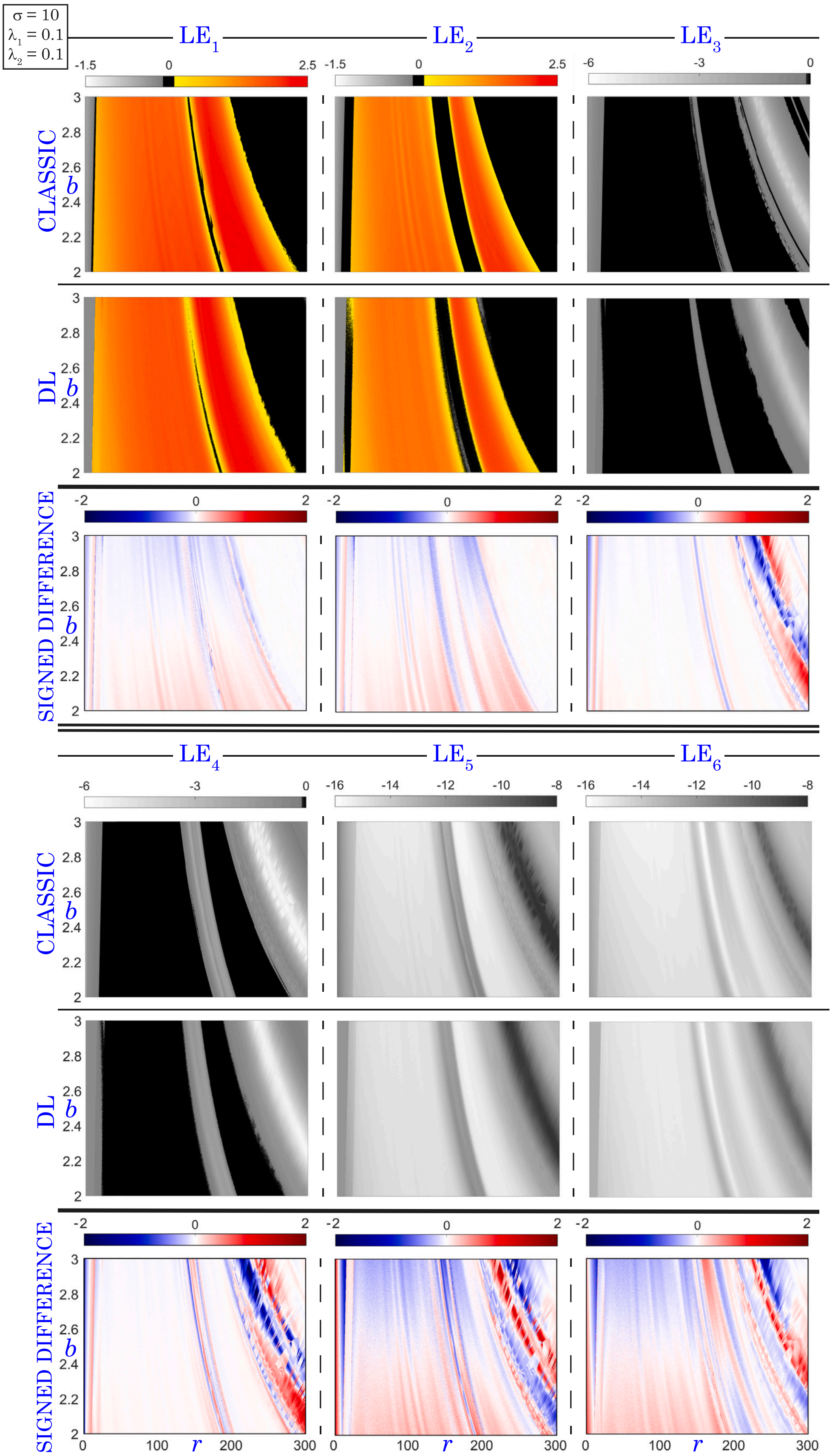


Fig. 10. 2D analysis of Lyapunov exponents in the coupled Lorenz system ($\sigma = 10$, $\lambda_1 = \lambda_2 = 0.1$) when training with random data (Huber loss value is [0.023 ± 0.004]). The analyses of the six LEs of the system are depicted for classical and DL techniques. The signed difference between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

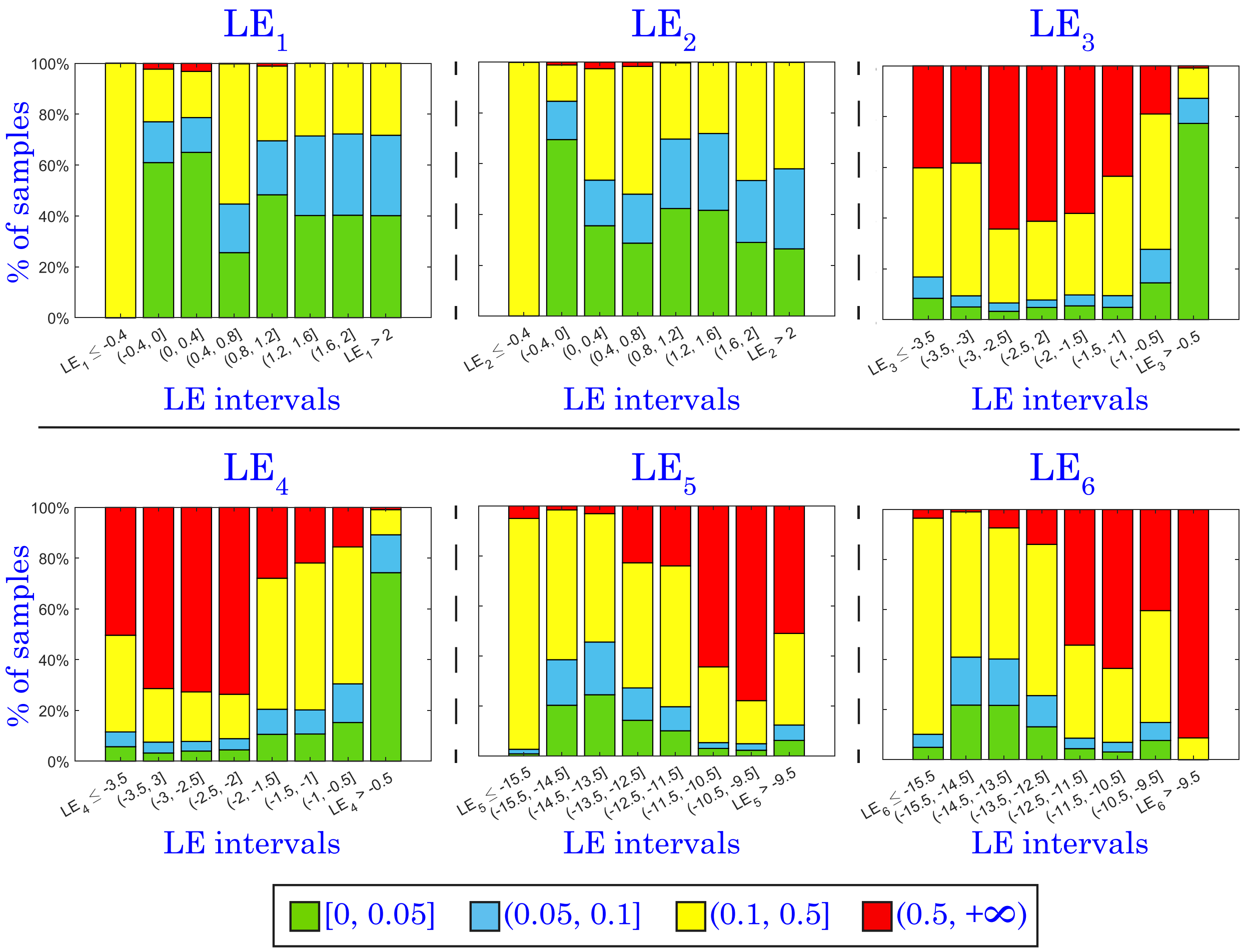


Fig. 11. Error analysis of Lyapunov exponents prediction in the (r, b)-parametric plane (studied in Fig. 9) of the coupled Lorenz system when training with non-random data. The color code is shown at the bottom. (See the text for more details.).

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

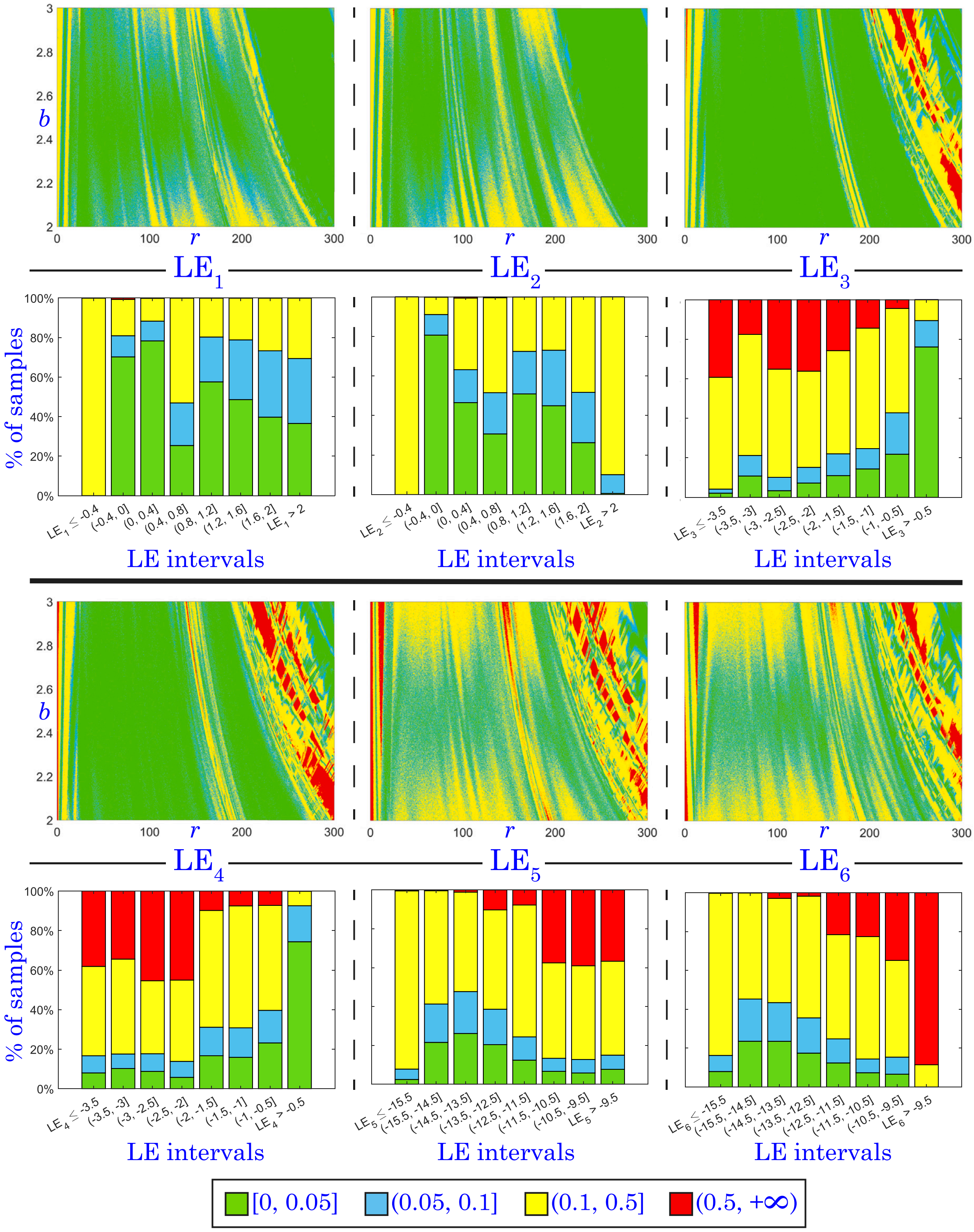


Fig. 12. Error analysis of Lyapunov exponents prediction in the (r, b)-parametric plane (studied in Fig. 10) of the coupled Lorenz system when training with random data. The first and third rows correspond to the errors represented on the parametric plane. In the second and fourth rows, the bar plots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

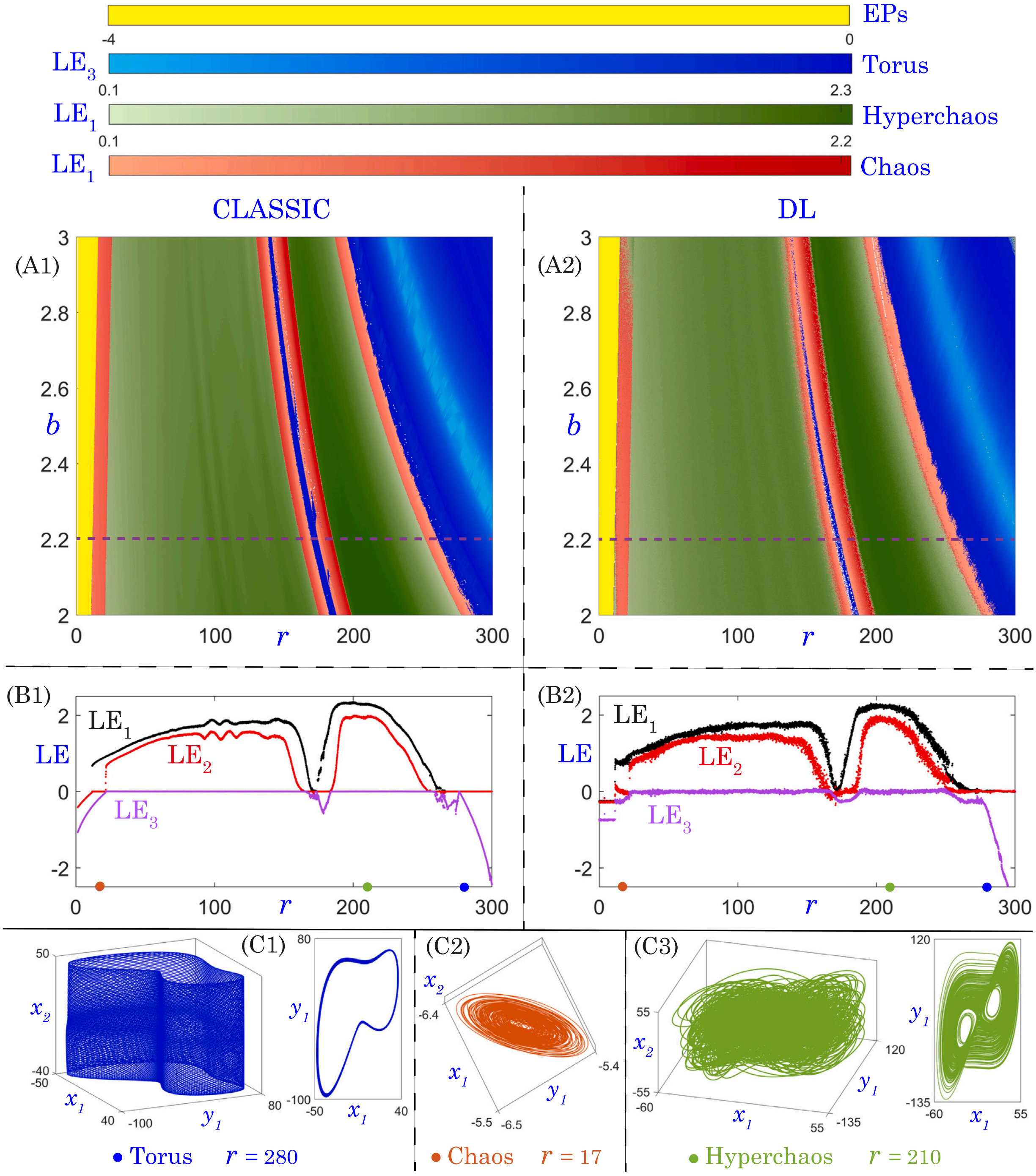


Fig. 13. Study of different dynamics in the coupled Lorenz system. (A1)-(A2) (r, b)-parametric study ($\sigma = 10$, $\lambda_1 = \lambda_2 = 0.1$) of LEs with the classical and DL techniques (random approach), respectively. Each color corresponds to different dynamics as indicated in the colorbars above: 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.

Nonlinear Dyn (2024) 112:15549–15565
https://doi.org/10.1007/s11071-024-09830-2

RESEARCH

Dominant patterns in small directed bipartite networks: ubiquitous generalized tripod gait

Álvaro Lozano · Rubén Vigara · Carmen Mayora-Cebollero · Roberto Barrio

Received: 8 January 2024 / Accepted: 31 May 2024 / Published online: 20 June 2024
© The Author(s) 2024

Abstract The synchronization patterns exhibited by small networks of neurons that regulate biological processes (CPGs) have aroused growing scientific interest. In many of these networks there is a main behavioral pattern within the parameter space. In particular, in the context of insect locomotion, tripod walking stands out as a predominant pattern, both in natural observations (where insects walk on tripod gait) and in mathematical models. This predominance appears to be stable under parameter variations within the network, suggesting a possible correlation with the underlying network topology. Tripod walking can be naturally extended to all CPGs with a bipartite connectivity. Then a natural question arises: Are "generalized tripod gaits" equally dominant among synchronization patterns within those networks? To investigate this, we carried out a comprehensive study covering all bipartite networks of up to nine neurons. For each of those networks we numerically explore the phase space using a quasi-MonteCarlo method to see what are the main synchronization patterns that the network can achieve. Then, all those patterns are grouped according to their dynamics. Generalized tripod gait was observed in all cases examined as the dominant pattern again. However, certain cases revealed additional stable patterns, mainly associated with the 3-colorings of the respective graph structures.

Keywords Small networks · Bipartite networks · Dominant pattern · Central pattern generators

Mathematics Subject Classification 37N25 · 37C27 · 92B20

1 Introduction

The study of patterns in directed networks is today a current line of research [1] due to the large number of practical applications. In the large amount of possible network configurations, the bipartite networks are a common type of networks whose nodes are organized into two main groups. Therefore, bipartite networks are useful for pairwise relationships between two different groups of nodes [2], as in predator/pray, producer/consumer, left-side/right-side, and so on. These

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11071-024-09830-2.

Á. Lozano (✉)
Departamento de Matemáticas and IUMA, Computational Dynamics group, University of Zaragoza, 50009 Zaragoza, Spain
e-mail: alozano@unizar.es

R. Vigara · C. Mayora-Cebollero · R. Barrio
Departamento de Matemática Aplicada and IUMA, Computational Dynamics group, University of Zaragoza, 50009 Zaragoza, Spain
e-mail: rvigara@unizar.es

C. Mayora-Cebollero
e-mail: cmayora@unizar.es

R. Barrio
e-mail: rbarrio@unizar.es

kinds of situations are quite common in nature, biology, social relationships, food web, …

The review [3] provides a description of certain important classes of biological networks that exhibit a native bipartite structure and their associated data. Ecological networks are a typical example and can be subdivided into three main types: food webs, mutualistic webs, and host-parasitoid webs. Most of them can be described using bipartite networks [4,5]. Beyond ecological networks, bipartite biomedical and biomolecular networks, such as gene-disease network [6] and modeling of protein complexes as networks, are more abstract since the network is usually designed using various indicators of human diseases or molecular interactions. Epidemiological networks are another common type of models that use bipartite networks. In this case the nodes are related with individual patients: bipartite structures can be built based on individuals who are classified by gender, location, infectious agent,… Related with these networks are global networks of social relations based primarily on behavioral patterns of individuals or groups.

Recently, the use of more specific models is growing, especially in the study of small neuron networks [7], like the Central Pattern Generators (CPGs), which are neuronal circuits that when activated can generate rhythmic motor patterns even in absence of sensory input (see [8–10]).

In such networks, structure can be defined at two levels: (a) at the structural connectivity level (how neurons are connected with each others), (b) at the functional connectivity level (how subgroups of neurons are firing in synchrony). At the structural connectivity level, some connectivities are such that they can be represented as exclusive subsets of neurons with connections only among subsets, and no connections among neurons within a subset. When connections are between two subsets only, we will call it a bipartite connectivity (see Fig. 1a where black white circles form the two structural bipartition). At the functional connectivity level, some networks yield rhythmical activities where some subgroups of neurons fire together in turn, so neurons can be now partitioned according to the dynamical pattern of activity (see Fig. 1). When there are two subgroups of synchronized neurons, we will denote this *bipartite (dynamical) pattern*, and when there are three, we will denote this *tripartite (dynamical) pattern*.

Hence, in CPGs, the existence of bipartite connectivity is of special interest with regards to their relation with bipartite patterns, like in the CPG that controls the movement of animals [11,12]. Therefore, bipartite networks can be applied to a wide range of problems, and so a theoretical study of possible small bipartite networks that can lead to some special pattern configurations is of great interest.

A first common attempt to study the dynamics of a network is to consider networks of oscillators [13–15], which allows to provide some theoretical insights. In the case of generic small networks we have recently proposed [16] two numerical techniques to deal with small neuron CPGs. In [16,17] we applied these techniques to the bursting neuron CPG model introduced by Ghigliazza and Holmes [18] to model the movement of insects (cockroaches). Throughout this article, we will call *GH network* this network.

GH network has a bipartite connectivity and can display both bipartite and tripartite patterns, depending on the parameters of the system, showing synchronization patterns that correspond to insect gaits that can be observed in nature, as the tripod (Fig. 1) or the tetrapod gaits (Fig. 2) [16]. In particular, the tripod gait appears as the dominant pattern [19,20] in a wide region of the parameter space [17]. Since the connections in the model are mutually inhibitory, it is natural to expect that the connected neurons should not all be active at the same time, and so that synchronization pattern would determine a partition of the set of neurons. It is also natural to wonder if the prevalence of tripod gait is to some extent a consequence of network bipartiteness.

The main goal of this article is to investigate if bipartite patterns are dominant in bipartite networks as in the GH network. To see at which extend network connectivity determines the synchronization patterns, we have considered every network made of 6 to 9 neurons with bipartite connectivity. For each of them, we have sampled initial conditions to describe the distribution of its emerging synchronization patterns. We report here some results about those distributions.

This article is organized as follows. In Sect. 2 we introduce the equations and methodologies of the model. Section 2.4 is dedicated to giving the definitions of color synchronization patterns. Section 3 presents the results, shows the dominance of bipartite patterns, and illustrates why in some network topologies it is possible to have non-bipartite patterns. Finally, we present our conclusions at the end of the paper.

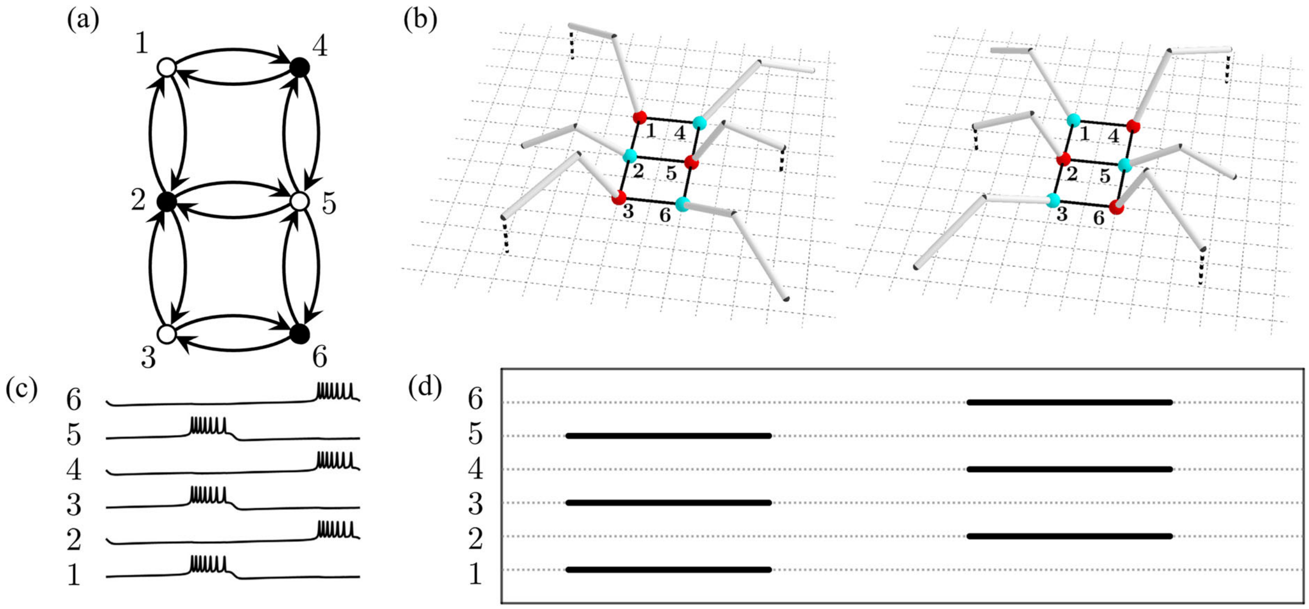


Fig. 1 Tripod gait. In the GH neuron network, with graph structure as in (**a**), each neuron is assumed to control one of the hexapod's legs, as it is suggested in (**b**), where active neurons are painted in red. A leg is moving if its control neuron is active (bursting). During tripod gait, neurons' bursting intervals synchronize with each others as depicted in (**c**). Activation and inactivation intervals of neurons are usually represented schematically in an hexagram as the one in (**d**). The neurons activate in groups of three, i.e., legs move alternatively in groups of three and there are always three or six legs on the ground. The synchronization pattern is associated with the 2-coloring of the graph depicted in (**a**)

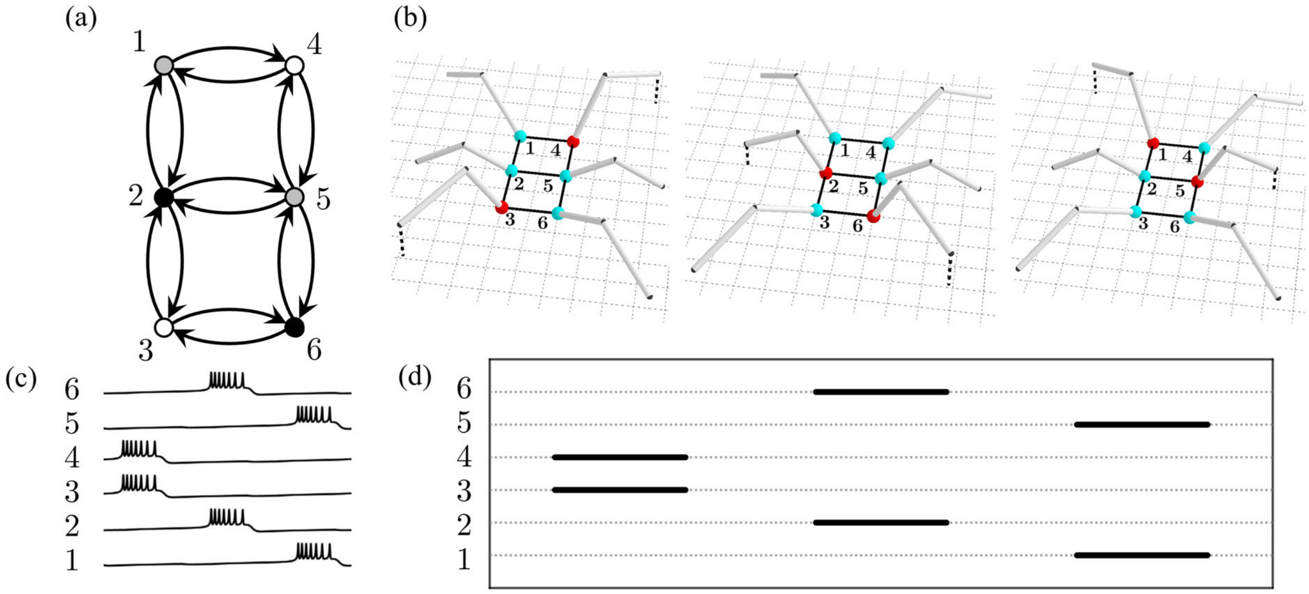


Fig. 2 Tetrapod gait. Neurons activate in pairs. Legs move cyclically in three groups of two. This synchronization pattern is associated with the 3-coloring of the graph depicted in (**a**). Changing one of the parameters of the system (8), the network changes its pattern to the tetrapod gait. See [16] for details

2 Neuron model, graph neuron networks and methodologies

2.1 Neuron model

In this article we construct small neuron networks where each neuron follows a Hodgkin-Huxley formalism adapted to the movement of insects (cockroaches [18]).

The dynamics of one isolated neuron are described in terms of a fast nonlinear calcium current, I_{Ca}, a slower potassium current I_K, a very slow current I_{KS}, a linear leakage current I_L, and an external current I_{ext}. The ODE system describing the dynamics of the action potential v, the potassium gate m and the slow potassium gate w is

$$\begin{cases} C\dot{v} = -(I_{Ca} + I_K + I_L + I_{KS}) + I_{\text{ext}}, \\ \dot{m} \ = \dfrac{\epsilon}{\tau_m(v)}[m_\infty(v) - m], \\ \dot{w} \ = \dfrac{\delta}{\tau_w(v)}[w_\infty(v) - w], \end{cases} \tag{1}$$

with the auxiliary ionic current functions defined by

$$\begin{aligned} &I_{Ca} = g_{Ca} n_\infty(v)(v - E_{Ca}), \quad I_K = g_K m(v - E_K), \\ &I_L = g_L\,(v - E_L), \quad I_{KS} = g_{KS}\, w\,(v - E_K), \end{aligned} \tag{2}$$

and where the different time scales and steady state gating variables are

$$\begin{aligned} &\tau_m(v) = \operatorname{sech}\bigl(\tfrac{k_K^0\,(v - v_K^{\text{th}})}{2}\bigr), \\ &\tau_w(v) = \operatorname{sech}\bigl(k_{KS}^0\,(v - v_{KS}^{\text{th}})/2\bigr), \end{aligned} \tag{3}$$

and

$$\begin{aligned} &m_\infty(v) = \bigl(1 + e^{-2k_K^0\,(v - v_K^{\text{th}})}\bigr)^{-1}, \\ &w_\infty(v) = \bigl(1 + e^{-2k_{KS}^0\,(v - v_{KS}^{\text{th}})}\bigr)^{-1}, \\ &n_\infty(v) = \bigl(1 + e^{-2k_{Ca}^0\,(v - v_{Ca}^{\text{th}})}\bigr)^{-1}. \end{aligned} \tag{4}$$

The parameters C, ϵ and δ determine the time scales of v, m and w. If X denotes any of the considered ions, E_X is the Nernst potential, g_X is the maximal conductance, and k_X^0 is the steepness of the transition happening at threshold potential v_X^{th}. This model can produce alternation between a rest state with a flat potential and a burst of fast oscillations of v around a high voltage, mimicking the action potential of a neuron without explicit refractory periods nor activation thresholds (as in LIF or QIF neurons). See §3.3 and §4 of [21] for details on its parameters and global dynamics.

The results of these experiments are expected to be similar for other types of excitable neuron models.

2.2 Neuron networks

Consider an undirected, connected and without loops graph with N vertices. We denote the degree of one of its vertices x, the number of adjacent vertices, by d_x. To build its associated neuron network model we consider a copy of (1) for each vertex x which represents a neuron of the network. For any variable "A" of the ODE system (1) the subscripted term "A_x" denotes the copy of this variable for the neuron x.

Since synapses are directional, we will consider each edge as two opposed synapses between the neurons. We modeled them using the voltage based synapse model of [18], given by the following ODE system

$$\begin{aligned} &\dot{s}_x = \alpha\, s_\infty(v_x)(1 - s_x) - \beta s_x, \\ &s_\infty(v_x) = \tfrac{T_{\max}}{1 + e^{-k_{\text{pre}}(v_x - E^{\text{pre}})}}. \end{aligned} \tag{5}$$

This new synapse variable s_x of neuron x will affect its post-synaptic neurons as follows: In this paper all synapses are inhibitory, thus, we add a subtractive (inhibitory) term to the first equation of (1) getting

$$C\dot{v}_x = -\bigl((I_{Ca})_x + (I_K)_x + (I_L)_x + (I_{KS})_x\bigr) + I_{\text{ext}} - (I_{\text{syn}})_x, \tag{6}$$

being

$$(I_{\text{syn}})_x = g_{\text{syn}} \frac{v_x - E^{\text{post}}}{d_x} \sum_{y \text{ adj } x} s_y, \tag{7}$$

where y runs over all the vertices of the graph adjacent to x (cf. [16,18]) and g_{syn} denotes the synaptic strength. As we can see in (7), we average the action of all pre-synaptic neurons. Hence, it can be seen that the maximal inhibitory current is the same across the network.

For instance, in the case of the GH network we have that the different values of the extra current $(I_{\text{syn}})_x$ for each neuron potential v_x are:

$$\begin{aligned}
(I_{\text{syn}})_1 &= \tfrac{1}{2}\, g_{\text{syn}}\,(v_1 - E^{\text{post}})\,(s_2 + s_4),\\
(I_{\text{syn}})_2 &= \tfrac{1}{3}\, g_{\text{syn}}\,(v_2 - E^{\text{post}})\,(s_1 + s_3 + s_5),\\
(I_{\text{syn}})_3 &= \tfrac{1}{2}\, g_{\text{syn}}\,(v_3 - E^{\text{post}})\,(s_2 + s_6),\\
(I_{\text{syn}})_4 &= \tfrac{1}{2}\, g_{\text{syn}}\,(v_4 - E^{\text{post}})\,(s_1 + s_5),\\
(I_{\text{syn}})_5 &= \tfrac{1}{3}\, g_{\text{syn}}\,(v_5 - E^{\text{post}})\,(s_2 + s_4 + s_6),\\
(I_{\text{syn}})_6 &= \tfrac{1}{2}\, g_{\text{syn}}\,(v_6 - E^{\text{post}})\,(s_3 + s_5).
\end{aligned} \tag{8}$$

In general, with this construction, we obtain a $4N$-dimensional ODE system. Obviously, this ODE system presents the same symmetries as the underlying network.

Notice that the inhibitory actions along edges can be asymmetric: the inhibitory current that x receives from y could be different from the inhibitory current that y receives from x, as those are averaged among number of neighbors as in the example of the GH network.

On the other hand, while all the neuron networks are directed (as synapses are), for the sake of simplicity, we just show the topology of the connection graph identifying opposed synapses.

Since we are investigating the ubiquity of a generalization of the tripod gait for other neuron networks, we have taken a set of parameter values for which the tripod gait is ubiquitous in the GH network [17]. We have fixed the parameters of the system to

$$\begin{aligned}
&g_{Ca} = 4.4, \quad g_K = 8, \quad g_{KS} = 0.15, \quad g_L = 2,\\
&C = 1.2, \quad E_{Ca} = 120, \quad E_K = -80, \quad E_L = -60,\\
&E^{\text{pre}} = 2, \quad E^{\text{post}} = -70, \quad I_{\text{ext}} = 35.5,\\
&T_{\max} = 0.002, \quad v_{Ca}^{\text{th}} = -1.2, \quad v_K^{\text{th}} = 2,\\
&g_{\text{syn}} = 0.03, \quad v_{KS}^{\text{th}} = -24, \quad k_{Ca}^0 = 0.055,\\
&k_K^0 = 0.1, \quad k_{KS}^0 = 0.4, \quad k_{\text{pre}} = 0.22, \quad \alpha = 5000,\\
&\beta = 0.18, \quad \delta = 0.005, \quad \epsilon = 4.9.
\end{aligned}$$

We should remark that the weights of the synaptic variables s_i used in [17] are different from those of (8). In our previous work we used the conditions of [18] where pre-synaptic s_i variables are weighted averaged: the weight of the synapses from 1 to 2, 3 to 2, 4 to 5 and 5 to 6 was half of the weight of the others (neurons numbered as in Fig. 1). Despite of this difference, fixed the rest of the parameters, the dynamic patterns are the same for both dynamical systems, namely, the tripod gait of Fig. 1, see [16,17].

2.3 Quasi-Monte Carlo sweep. Patterns

The main goal of this article is to investigate if bipartite patterns are dominant in bipartite networks as in the 6-neurons GH CPG network. That is, we want to know the fraction of initial conditions producing a bipartite pattern in the long run compared to the rest of patterns, for all bipartite networks with up to 9 neurons. Since the phase space is $4N$-dimensional, where N is the number of neurons, it is computationally unfeasible to sweep a region of this space with a (traditional) fixed step sweep. Even the coarsest of the partitions with 2 single points in each dimension of the swept region would require millions of initial conditions to integrate and the result would not be representative of the swept region (they would be just the corners of the $4N$-dimensional cube of initial conditions selected). Hence, we have applied the quasi-Monte Carlo sweep technique [16,17] described below. For each neuron network, this technique combines: (i) the choice of an appropriate set of initial conditions; (ii) a numerical integration of the neuron network ODE system for each of these initial conditions; and (iii) a final post-processing of the solution until we obtain a pattern, which will be a discrete, purely combinatorial object associated with the underlying network.

Now, for each neuron network we select 200 initial conditions (this is true for the rest of the simulations on this paper) using Halton sequences [22], a well-known low discrepancy sequence generator.

The initial membrane potentials are between -20.2mV and 4.8mV, and the rest of variables are confined to [0, 1].

The system is integrated, using the DOPRI78 integrator (a classical embedded RK method of Dormand and Prince [23,24]), for $t \in [0, 100]$ to allow it to eventually reach a periodic orbit. If this is the case, we compute the bursting (active) intervals of each neuron (the duty cycle). To do so, we compute a moving standard deviation along the voltages and declare a neuron active if this standard deviation surpasses a given threshold. Then, we build the firing pattern with this information: the period of the orbit is divided into ranges along which the status of the network (the sets of active and inactive neurons) remains constant. The limit points of each of these ranges coincide with the moment of activation or inactivation of one or more neurons, and vice versa: every moment of activation or inactivation of any neuron is a limit point for these ranges.

Since we want to identify dynamical patterns that are similar, independently of the global time scale, we drop the ranges which are too small with respect to the longest one. More precisely, we fix a parameter $t_{\text{small}} \in (0, 1)$, and a range is discarded if its length is smaller than $t_{\text{small}} \cdot T_{\text{MAX}}$, where T_{MAX} is the greatest length among all ranges. After this filtering, we obtain a sequence of ranges with the status of each of them, and we remove the length of events to group patterns with the same combinatorial behavior. If the network has p neurons $x_1, \ldots, x_p$, the status of a specific range can be stored in the binary expansion $a_1 \ldots a_p$ with $a_i = 0$ or 1 if x_i is inactive or active respectively for $i = 1, \ldots, p$, or equivalently in the number $s = \sum_{i=1}^{p} a_i 2^{i-1}$ lying between 0 and $2^p - 1$. If the period of the solution has been divided into r ranges, at the end we get an integer vector $(s_1, \ldots, s_r)$ with $s_j \in \{0, 1, \ldots, 2^p - 1\}$ for $j = 1, \ldots, r$ which is the pattern of the solution for the chosen initial conditions.

For each of the considered networks we finally obtain (at most) 200 patterns, but some of them could be equivalent. Two patterns are considered equivalent if they are related by: (i) a reordering of the neurons of the network; (ii) a cyclic translation of the pattern coordinates; (iii) an automorphism of the network graph; or (iv) a combination of the previous operations. After a final post-processing to check equivalences, the pattern list is divided into equivalence classes. The final output of the processing is, for each network, a list of patterns (representatives of each equivalence class), together with the size of the equivalence class of each of them.

2.4 Color synchronization patterns: definitions

A graph is *q-colorable* if its vertices can be grouped into q disjoint non-empty subsets in such a way that adjacent vertices lie on different subsets of the partition. If this is the case, such decomposition is a *q-coloring* of the graph and the subsets are the *colors*. A graph is bipartite if and only if it is 2-colorable. Two q-colorings of a graph are equivalent if they are related by an automorphism of the graph, and they are said to be different otherwise. It is easy to deduce that all 2-colorings of a bipartite graph are equivalent.

On the other hand, a synchronization pattern on a neuron network defines in a natural way a partition of the set of neurons: two neurons x, x' are in the same subset of the partition if they have overlapping bursting intervals or if there is a sequence of neurons

$$x = x_0, x_1, \ldots, x_Q = x' \tag{9}$$

from x to x' such that each x_i has an overlapping bursting interval with x_{i+1}. We say that the pattern is a *q-color pattern* if the partition defined by the pattern is a q-coloring of its underlying graph. The pattern is a *color pattern* if it is a q-color pattern for some q. Following the notation from graph theory, a 2-color pattern is also called a *bipartite pattern*. With this notation, the tripod gait on the GH network is a bipartite pattern while the tetrapod gait is a 3-color pattern. All the patterns that were obtained in [16,17] for the GH network are q-color patterns for $q = 2$, 3 or 4.

Remark 1 As already commented, since synapses are inhibitory, it is natural to expect that connected neurons would not be active at the same time. Therefore, it is also natural to expect that color patterns form an important subfamily of synchronization patterns.

3 Results

In order to study all the possible patterns, we have applied the quasi-Monte Carlo sweep (Sect. 2.3) to all the 982 bipartite bidirectional networks constructed as in Sect. 2.2 from the bipartite graphs with up to 9 vertices of the dataset of [25]. The following results were obtained taking $t_{\text{small}} = 0.25$. We took this high value to simplify the resulting patterns as much as possible. Nevertheless, with smaller values of this parameter the results were quite similar, with the exception that is discussed in Sect. 3.3.

We show possible examples of results on a particular network in Fig. 3 to illustrate the existence of numerous patterns on each network. For display purposes, we only show the connectivity of the network, but all synapses are bidirectional (see Sect. 2.2). We also show the corresponding time series and synchronization patterns. Synchronization patterns are usually represented schematically in diagrams, like those in Figs. 1d and 2d,

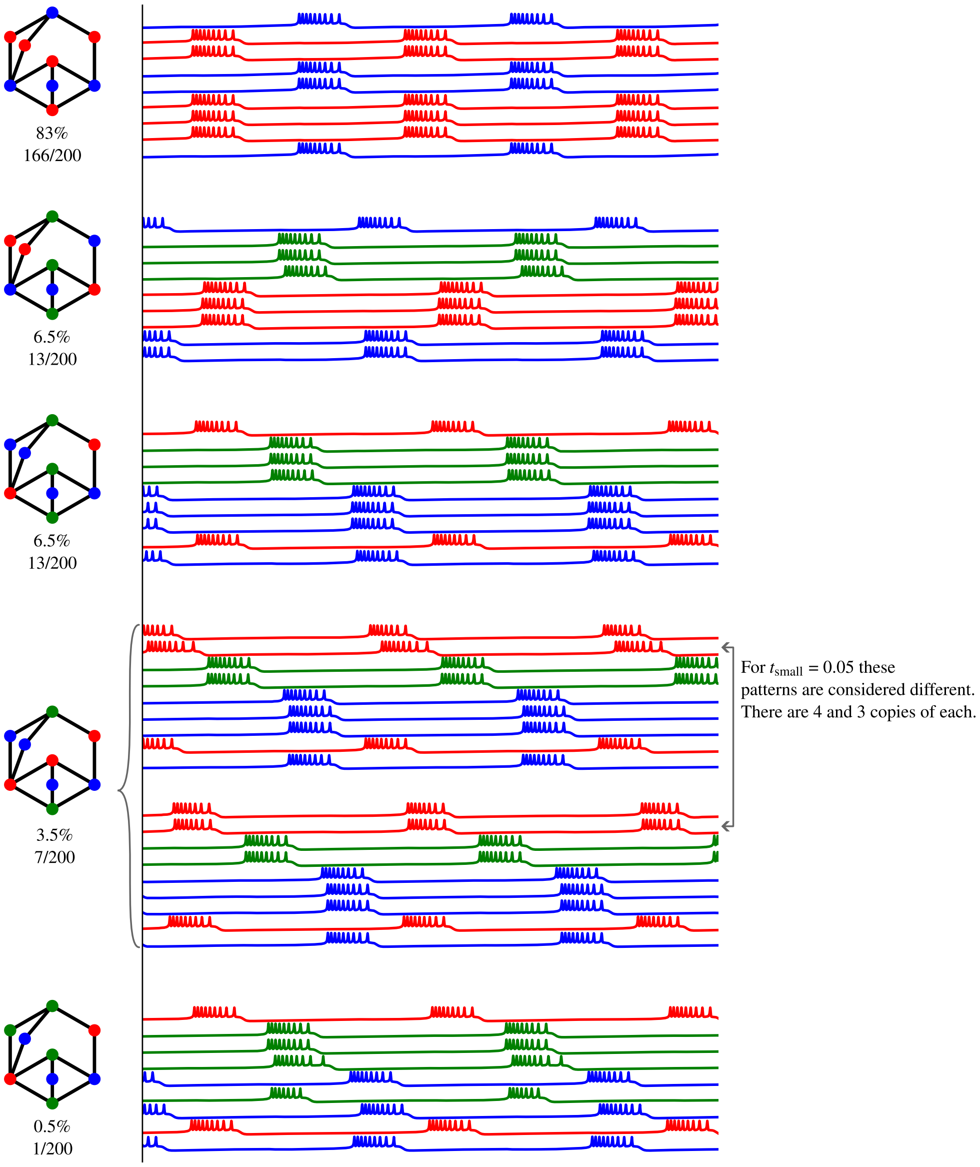


Fig. 3 All observed patterns in a 9-neurons CPG considering a set of 200 initial conditions in a Halton sequence. The total percentage of each pattern is indicated. The time series of the 9 neurons are presented for each pattern, and they show the bursting dynamics of each neuron and the synchronization patterns. We have colored the groups of neurons according to their synchronization with $t_{\text{small}} = 0.25$

where each horizontal line represents the state (active or inactive) of a single neuron. These diagrams fit well with the examples presented.

Note that any network can have a large number of patterns (multistability), but the size of its basin of attraction can be very small. We estimate it by considering the number of initial conditions that converge to each one and giving the corresponding percentage. In Fig. 3 we show for a particular 9-neurons CPG example all the patterns (5 patterns) found considering a set of 200 initial conditions in a Halton sequence. By increasing this number more patterns may appear, but with a low percentage value. Furthermore, most of the patterns maintain the same bursting dynamics (same number of spikes per burst), but some of them change the number of spikes per burst in some neurons (see neuron 6 in the last pattern). We remark that we have set along the article a fixed threshold percentage (5%) to perform the analysis, but in this example we show all the found patterns. We have colored the groups of neurons according its synchronization with $t_{\text{small}} = 0.25$. Note that increasing the value t_{small} allows more patterns to be considered equal and therefore increase their overall percentage and pass the threshold percentage. For the fourth pattern we show two sets of time series with a slight difference on one of its neurons that are considered equal with $t_{\text{small}} = 0.25$. This difference is seen taking $t_{\text{small}} = 0.05$.

In the following, we will represent synchronization patterns for generic networks together with network topology with a 3D representation (Fig. 5 and following) where we draw some copies of the network in parallel horizontal planes and vertical cylinders represent the activation intervals of the corresponding neurons. We remark that the use of graph techniques in networks has also been recently considered in [26].

3.1 Dominance of bipartite patterns

The first result that we discuss is that for the selected parameter values, which are within the tripod gait dominance region for the insect movement network, there is an evident predominance of bipartite patterns, that is, tripod-type patterns. The results we have obtained are the following:

- 100/70 rule: in 100% of the graphs considered, the bipartite pattern appeared in more than 70% of the sampled initial conditions, and in near 70% of the graphs (627 of the 982 graphs) the bipartite pattern appeared in 100% of the sampled initial conditions.
- 95/90 rule: for around 95% of the graphs (929 out of 982), the bipartite pattern appeared in at least 90% of the sampled initial conditions, and for nearly 90% of the graphs (863 out of 982), the bipartite pattern appeared in at least 95% of the sampled initial conditions.

Moreover, in more than 99% of the graphs (978 out of 982), the bipartite pattern appeared in at least 80% of the simulations. Thus, it is clear that the bipartite pattern is predominant for all the networks, at least with the chosen set of parameters.

Remark 2 The actual patterns depend heavily on the initial conditions used for the quasi-Monte Carlo method. Different initial conditions may report slightly different patterns, since we depend on 200 sampled points on high dimensional spaces. However, the global picture is the same for other sets of initial conditions.

3.2 Non-bipartite patterns

In our simulations there appeared other synchronization patterns different from the bipartite ones. We will focus in those non-bipartite patterns that have a relevant statistical significance: we will take into account only those patterns that appear in more than 5% of the simulations of the corresponding network. Among all the considered networks, 83 have a non-bipartite pattern (Fig. 4): 71 of them have just one non-bipartite pattern and 12 of them have two non-bipartite patterns. This makes a total number of 95 non-bipartite patterns, 87 of which are 3-color patterns. The remaining 8 non-bipartite patterns are not color patterns and they can be divided into two classes. On one hand, there are three patterns where there are no connected neurons firing at the same time but such that the sequences of partial overlappings of the bursting intervals produce that connected neurons are in the same subset of the partition defined by the pattern. These examples are depicted in Fig. 5, where it can be seen that in fact all the neurons of the network are connected by the sequence of bursting interval overlappings and thus the partition defined by the pattern is the trivial partition containing only the whole set of neurons. On the other hand, there are 5 patterns (Fig. 6) where the partition defined by the pattern contains 3 subsets, but such that there are “bad edges” connecting neurons that fire at the same time despite inhibitory coupling. Those patterns are the only ones that contradict Remark 1, and they deserve a further investigation.

3.3 t_{small} and strong vs weak synchronization patterns

For a given activity pattern, we can distinguish two levels of synchronization according to the type of intersections between the bursting intervals of different neurons that can be found. On the one hand, there are patterns where the bursting intervals of the neurons are fully synchronized: if two neurons are both active at some point, the coincidence of their bursting intervals is complete. On the other hand, there are patterns where partial overlappings between bursting intervals occur: there are neurons which are all active at some common time but whose bursting intervals do not coincide (Fig. 5). We will call *strong* patterns to the former ones and *weak* patterns to the latter.

We have investigated if the obtained patterns are strong or weak synchronization patterns. For non-bipartite patterns our findings depend heavily on the pre-processing of the pattern data. As we have explained in Sect. 2.3, in order to identify similar patterns, in some step we discard parts of the dynamical pattern which happens for a small amount of time, quantified by a fixed parameter $t_{\mathrm{small}} \in (0, 1)$. When this parameter is small, very few non-bipartite patterns

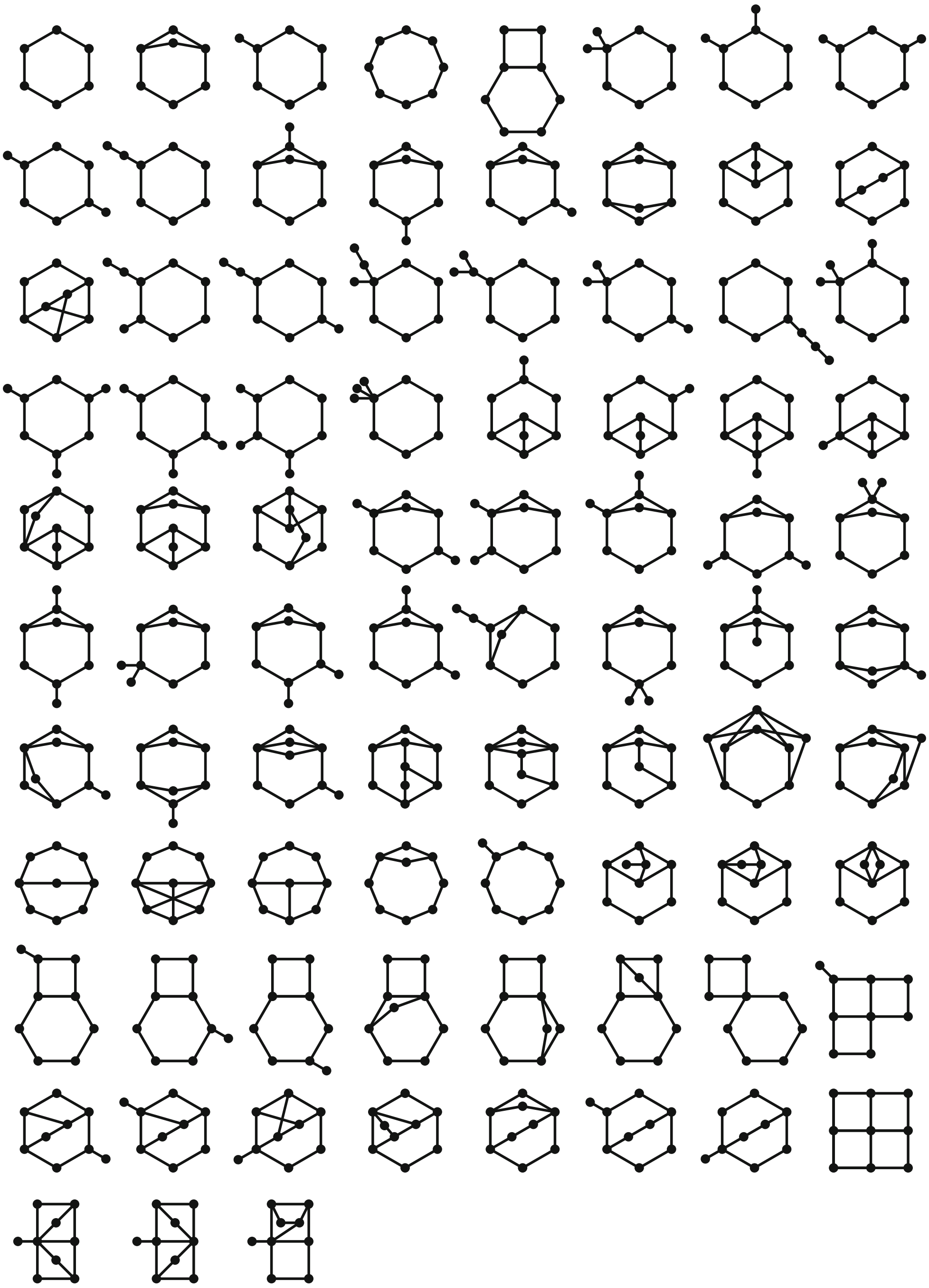


Fig. 4 All the graphs having a non-bipartite pattern from 6-neurons up to 9-neurons CPG networks. In the sake of visualization, we just show the connectivity of the network, but all the synapses are bidirectional, see Sect. 2.2

are strong synchronization patterns. As this parameter grows, more synchronization patterns are associated (as we are discarding short length differences between the patterns) and more of them become strong. For example, if t_{small} is set equal to 0.05 only 3 non-bipartite patterns are strong patterns (Fig. 7). On the contrary, when t_{small} is set equal to 0.25, there are only 4 weak synchronization patterns. Three of them can be seen in Fig. 5, the remaining one has a “bad edge” (Fig. 6e). On Fig. 6a-d we present the 4 non-bipartite strong with a “bad edge”, that is, with an edge connecting neurons that fire at the same time despite inhibitory coupling.

It is important to remark that this dependence on the parameter t_{small} does not occur for bipartite patterns: *bipartite patterns appear always as strong patterns even for very small values of* t_{small}.

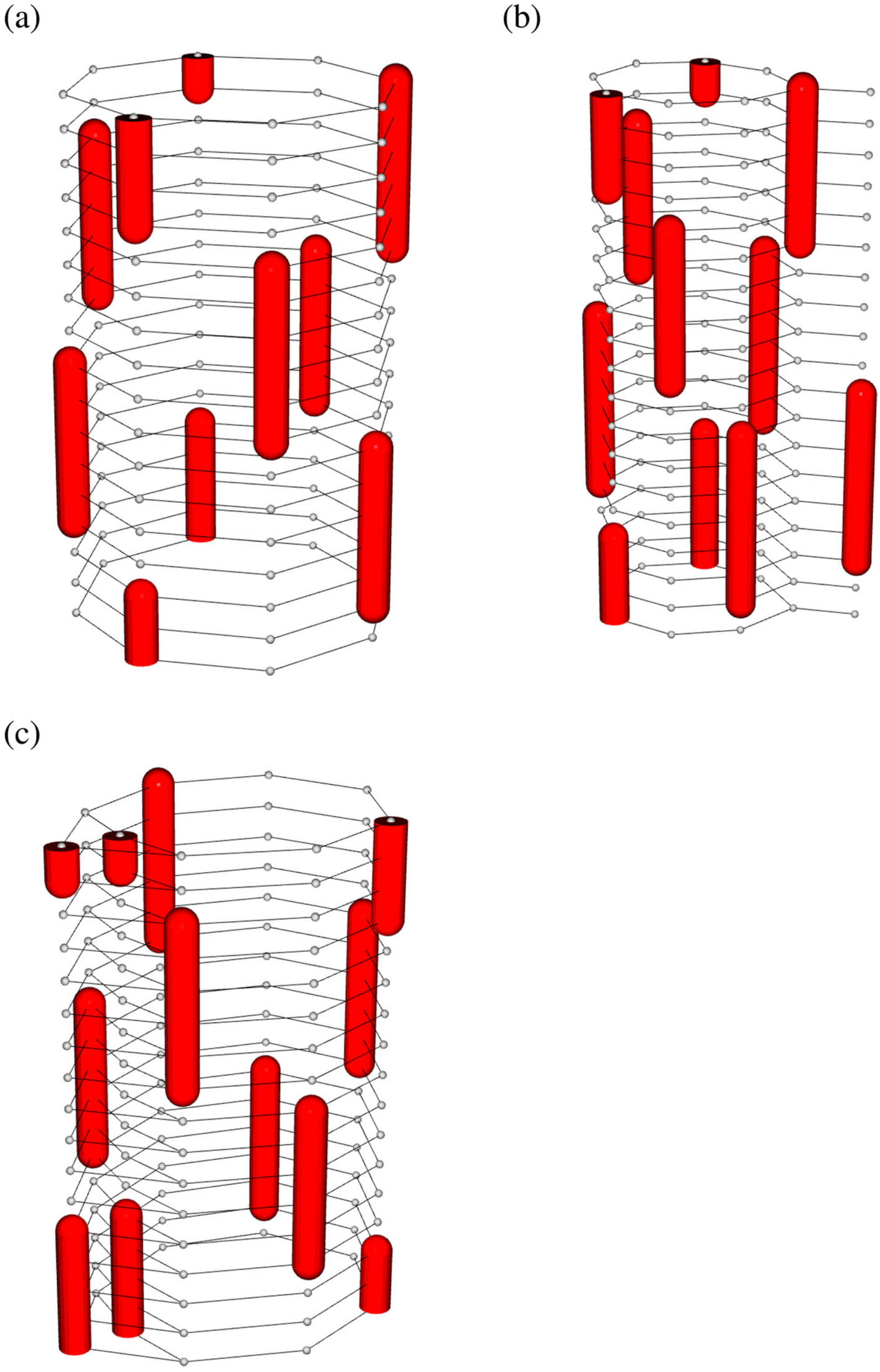


Fig. 5 Three weak patterns with $t_{\text{small}} = 0.25$. Although connected neurons never fire at the same time, the concatenation of partial overlappings of the bursting intervals produce that connected neurons are in the same subset of the partition defined by the pattern. There is another weak pattern but it has a bad edge (Fig. 6e)

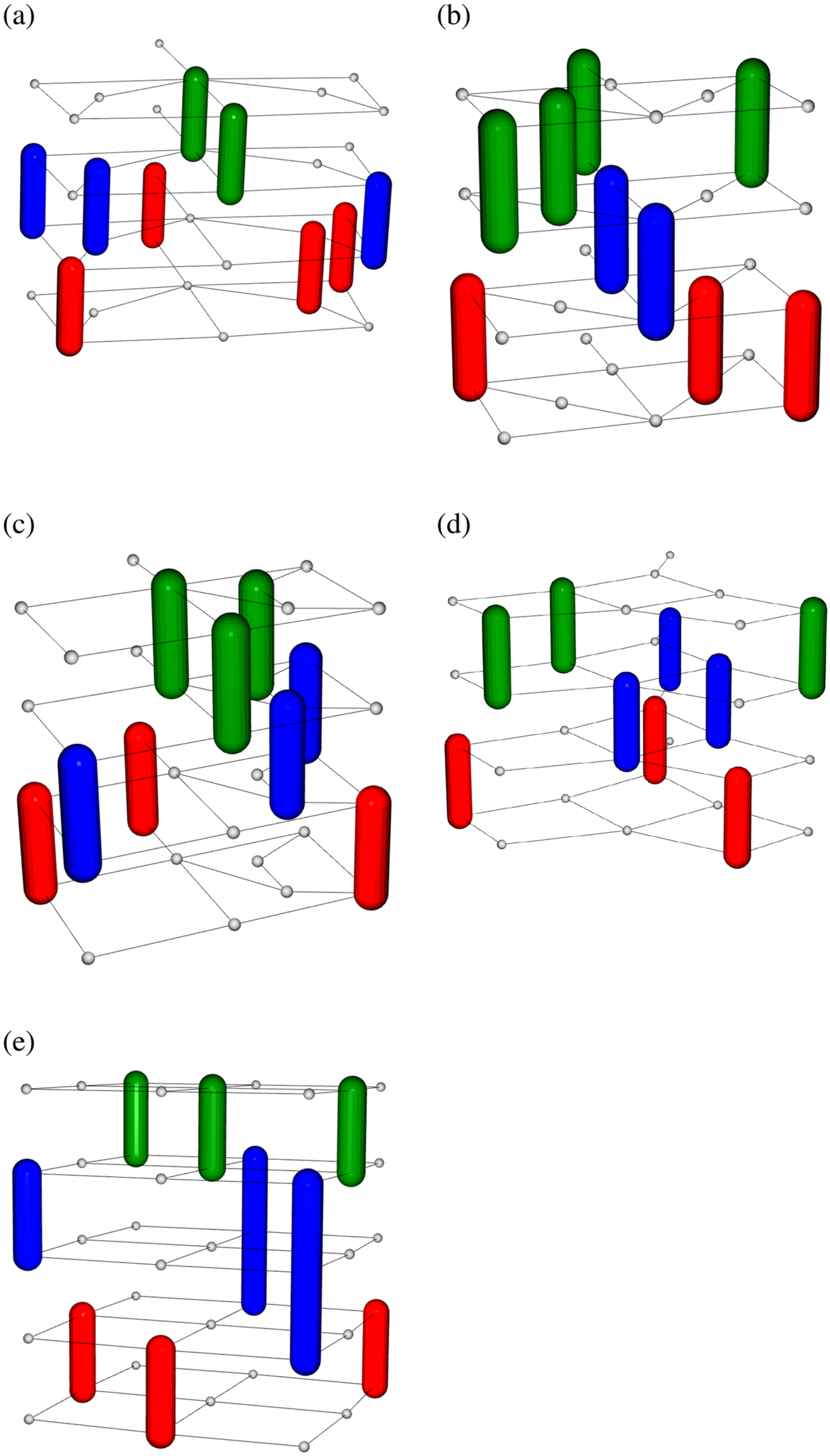
3.4 Some remarkable findings

We end this section with some remarkable results about patterns on cyclic graphs and non-bipartite graphs.

3.4.1 The octagon: starry traveling waves on cyclic graphs

The diagram of Fig. 5a shows a special pattern on the cyclic graph of 8 vertices (the octagon). In this pattern

Fig. 6 Non-bipartite patterns with a "bad edge". There are connected neurons whose bursting intervals coincide



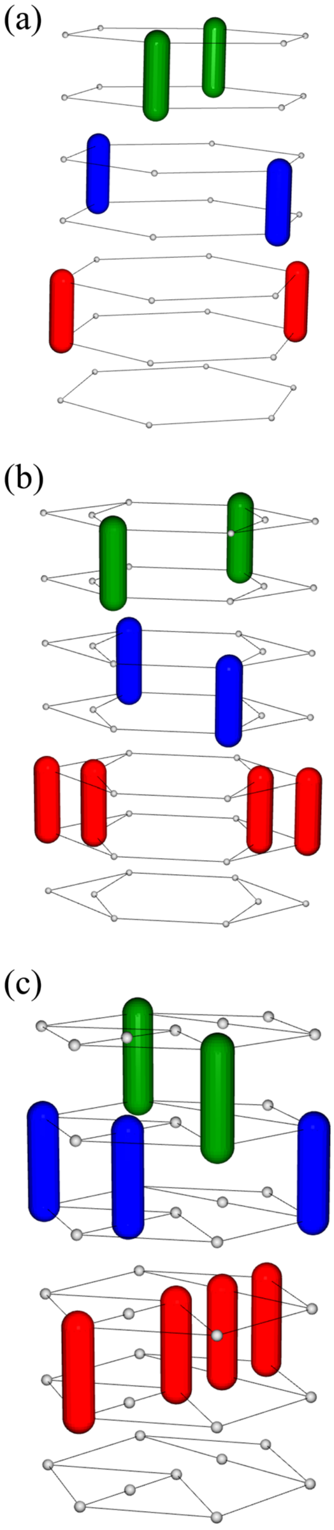


Fig. 7 The only non-bipartite strong patterns with $t_{\text{small}} = 0.05$

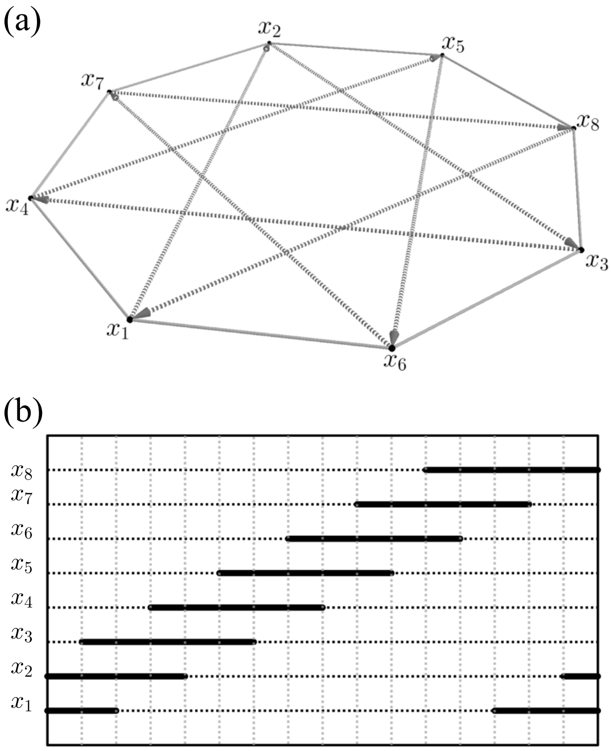


Fig. 8 The (8, 3)–star traveling wave pattern

the neurons are firing consecutively following a starry-like traveling wave. There is an ordering of the eight neurons

$$x_1 \to x_2 \to x_3 \to x_4 \to x_5 \to x_6 \to x_7 \to x_8 \quad (10)$$

such that x_i fires exactly after x_{i-1} for $i = 2, \ldots, 8$ (and x_1 fires again after x_8) with a certain delay, causing an overlapping of their bursting intervals. The ordering of the neurons is the one used for drawing the 8-pointed star out from the vertices of the regular octagon (Fig. 8a): (i) starting from any vertex we jump to the vertex located 3 positions further (clockwise or counterclockwise, clockwise in Fig. 8a); (ii) we repeat this movement with all the jumps in the same sense (clockwise or counterclockwise) until we are back at the starting vertex. As we can see in Fig. 8b, the pattern period is divided into 16 ranges such that at each range there are alternatively 2 or 3 firing neurons.

In our experiment for the octagon, the bipartite pattern was present in 82% of the simulations, while the starry traveling wave pattern (considering as the same the clockwise and counterclockwise cases) appeared in the remaining simulations. We have investigated if similar starry traveling wave patterns have the same

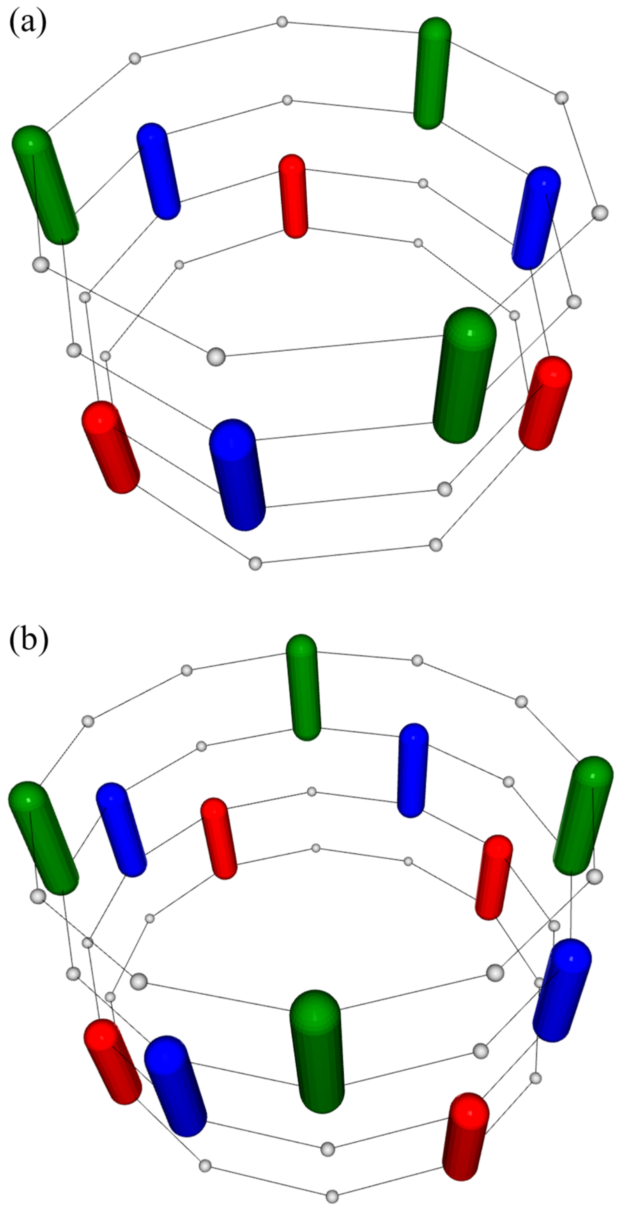


Fig. 9 3-color patterns in the 9-gon and in the 12-gon

importance for other cyclic graphs. From now on we will use the terms *p-gon* or C_p to denote the cyclic graph with p vertices. In our original experiment there only appeared the cyclic graphs of orders 4, 6 and 8 (the only bipartite cyclic graphs with up to 9 vertices). The bipartite pattern appeared in around 100% and 90% of the simulations for the 4-gon and for the 6-gon, respectively. In the remaining simulations for the 6-gon there appeared the 3-color pattern of Fig. 7a.

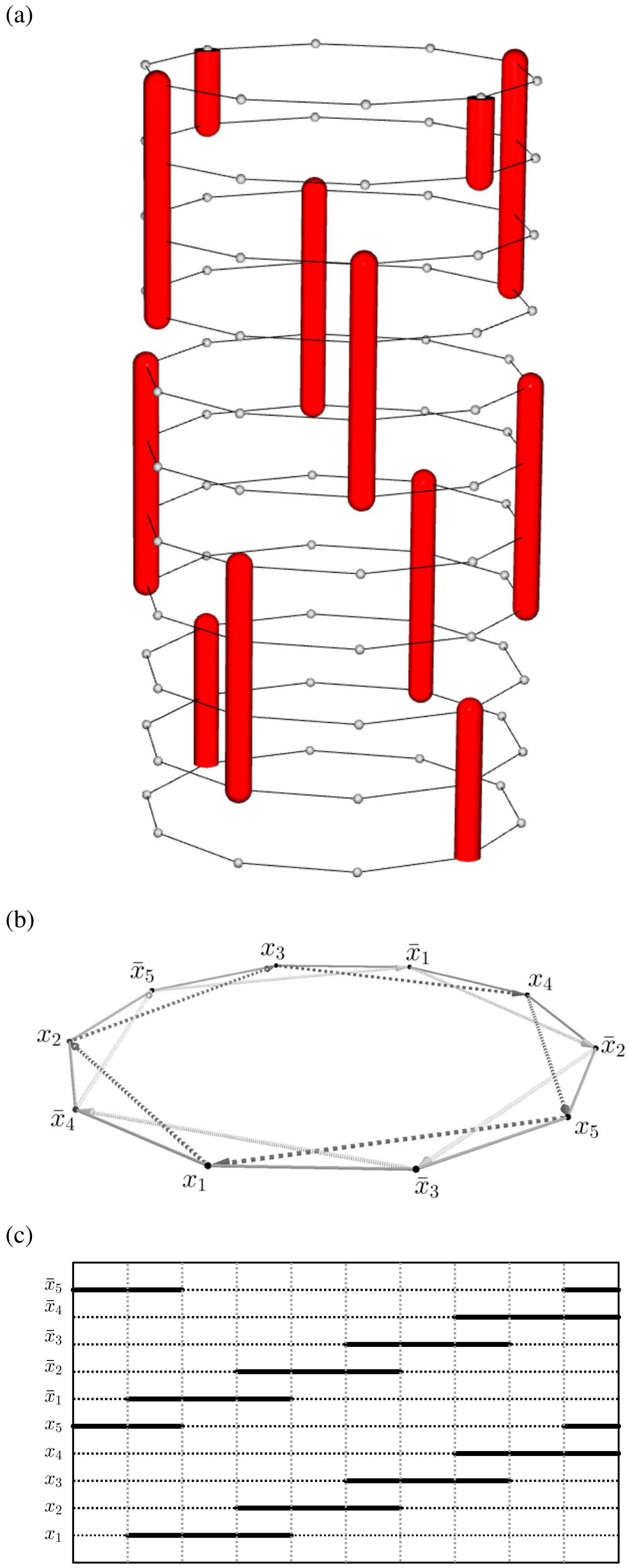
There is a well-known general procedure for drawing a p-pointed star out from the vertices of C_p:

(i) choose a positive integer number q coprime with p and with $q < p/2$;
(ii) starting from any vertex x_1 of C_p, jump to the vertex located q positions further (say clockwise) and join both vertices with a segment;
(iii) repeat (ii), each time starting from the final vertex of the previous step, until reaching x_1 again. If we call (p, q)–*star* to the resulting figure, we can consider an associated (p, q)–*star traveling wave pattern* in the network with underlying graph C_p in the same way as we explained for C_8 before. With this notation, the pattern in C_8 considered before is an (8, 3)–star traveling wave pattern.

We made the simulations also for the cyclic graphs of order 5, 7, 9, 10, 11 and 12, with the following findings:

- For $p = 5, 7, 11$ (prime numbers) the whole set of found patterns were star traveling wave patterns. For $p = 5, 7$ almost 100% of the patterns are the (5, 2)– and (7, 2)–star traveling wave patterns, respectively. For $p = 11$, the (11, 2)–star traveling wave pattern appears in 92% of the simulations while the (11, 3)–star traveling wave pattern appears in the remaining ones.
- For $p = 9$, the (9, 2)–star traveling wave pattern appears in 97% of the simulations.
- For even p ($p = 10, 12$) the resulting graph is bipartite and as it could be expected the bipartite pattern is again dominant, appearing in 71% and 65% of the simulations respectively. For $p = 12$, almost all the remaining cases took the form of the (12, 5)–star traveling wave pattern.
- For p multiple of 3 ($p = 9, 12$) the 3-color patterns of Fig. 9 appear but without statistical significance: only in 1.5% and 0.5% of the simulations for C_9 and C_{12} respectively.
- For $p = 10$, there appears no star traveling wave pattern. Instead, the non-bipartite cases converged to the pattern depicted in Fig. 10a. In this pattern, each neuron is fully synchronized with its opposite one. Starting from a couple of firing neurons $x_1, \bar{x}_1$ located at opposite vertices of C_{10}, with a delay of 2/3 of their bursting interval, the neurons $x_2, \bar{x}_2$ located 2 positions further (clockwise in the picture) from $x_1, \bar{x}_1$ respectively start their activity intervals, and so on. This pattern can be seen as the coupling of two interlaced/synchronized (10, 2)–star traveling wave patterns, having in mind that a

Fig. 10 The non-bipartite pattern on the 10-gon. Opposite neurons are fully synchronized. When 2/3 of the bursting interval of one pair of opposite neurons has elapsed, the neurons located two positions further from them (in clockwise order in the picture) fire up



(10, 2)–star is not even a star but a 5-gon, indeed. With this viewpoint, the 3-color pattern of the 6-gon depicted in Fig. 7a can be considered also as the coupling of two (6, 2)–star traveling wave patterns.

Therefore, it seems that, at least for the chosen set of parameters, for odd p (non-bipartite connectivities) the $(p, 2)$–star pattern is the dominant pattern in C_p. For even p the bipartite pattern is dominant in C_p but it is not clear which must be the second pattern in importance for a generic p. It is also interesting to note that when p is a multiple of 3 ($p = 9, 12$) the quite symmetric patterns of Fig. 9 rarely appear.

If p is even, as p increases, the percentage of simulations giving rise to the bipartite pattern decreases in favor of the star traveling wave patterns.

3.4.2 Non-bipartite examples

We wonder if it is possible to predict the dominant patterns of non-bipartite networks from the network topology. In Sect. 3.4.1 we have seen a few examples (cyclic networks with odd number of neurons), where the starry traveling wave patterns appeared as dominant. We have studied a couple more examples: the 3-color networks whose underlying graphs are the complete 3-color graphs K_{222} and K_{333} of Fig. 11. In general, the complete 3-color graph K_{abc}, with $a, b, c \in \mathbb{N}$, is constructed by taking three sets with a, b and c vertices and connecting each vertex of a set with all the vertices of the other two sets (and not connecting vertices within the same set). These graphs are interesting because they are not bipartite, their 3-coloring is unique, and every 3-colored graph with a, b and c vertices in its colors is a subgraph of K_{abc}.

We have performed the simulation for those two networks. Their 3-color patterns of Fig. 11c, d are the most relevant ones, but their dominance is weaker than that of the bipartite pattern in bipartite networks. The 3-color pattern appears in around 50% of the simulations for K_{222}, and in 40% for K_{333}. Thus, the dominance of bipartite patterns for bipartite networks cannot be extended to 3-color patterns in 3-colorable, non-bipartite, networks.

4 Conclusions

This article shows how in some situations the topology of the network has a strong influence on its dynamics. The tripod and tetrapod gait pattern appearing in the

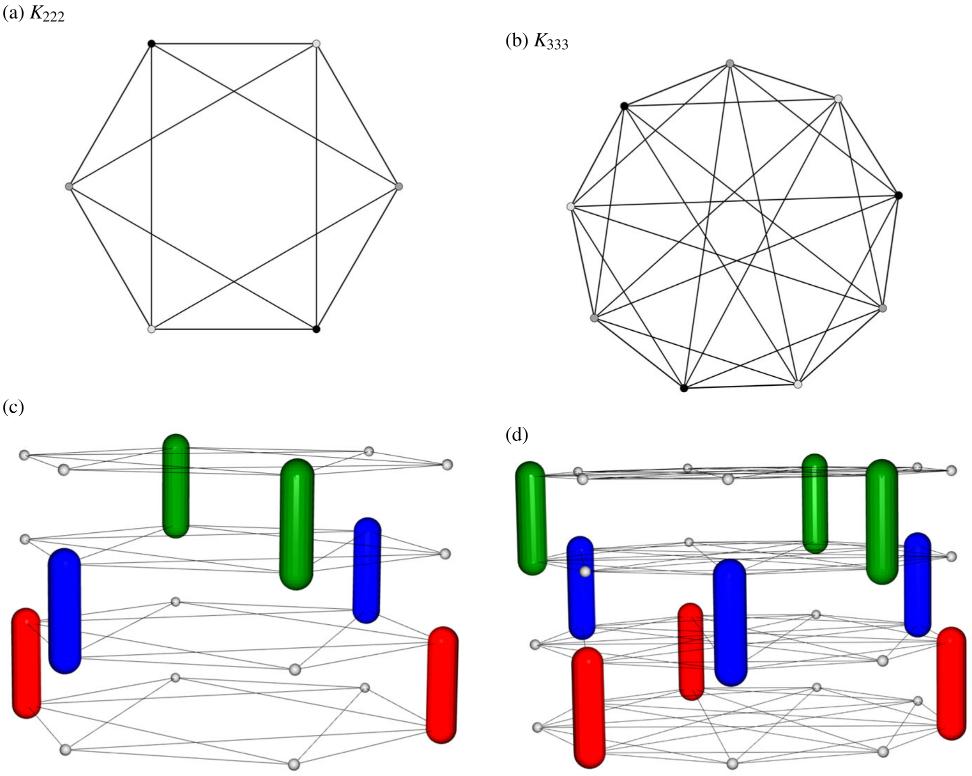


Fig. 11 3-color patterns on complete 3-color graphs

Ghigliazza and Holmes CPG model for the movement of insects have natural generalizations in color patterns for general directed networks.

In particular, tripod gait has a natural generalization as bipartite patterns for directed bipartite networks. We have generalized GH network to networks over a general graph, and using a set of parameters for which the tripod gait is ubiquitous for this model we have checked if bipartite patterns are also ubiquitous in other cases. We have explored all bipartite networks of up to 9 neurons, and the bipartite pattern appears as the dominant pattern in all cases. In the particular case of cyclic graphs, we have studied the possibilities of the graphs of order 5, 7, 9, 10, 11 and 12 and we have seen that for odd p the $(p, 2)$–star pattern is the dominant pattern in C_p. This result is a first attempt to study larger networks where these structures have a lot of sense (like in the movement of centipedes).

This dominance of bipartite patterns and other interesting cases that appear too, suggest some topics that deserve further research, such as:

- the dominance of bipartite patterns for longer networks and other regions of the parameter space;
- patterns with bad edges, where connected neurons are fully synchronized;
- patterns in cyclic networks, with special attention to starry traveling wave patterns;
- the importance of 3-color patterns in 3-colorable, non-bipartite networks; and
- which dynamical patterns are stable under random perturbation of the parameters?

Author contributions All authors have contributed equally to this manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. AL, RV, CMC, and RB have been supported by the Spanish Research project PID2021-122961NB-I00. AL has been supported European Regional Development Fund and Diputación General de Aragón (ER22_23R) RV, CMC and RB have been by the European Regional Development Fund and Diputación General de Aragón (E24-23R). RB has been supported by the Spanish Research project TED2021-130459B-I00 and by the European Regional Development Fund and Diputación General de Aragón (LMP94-21).

Data availability Graphs, initial conditions and resulting patterns are available as Supplementary Material.

Declarations

Conflict of interest The authors declare that they have no Conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. da Fontoura Costa, L.: Discovering patterns in bipartite networks. BioRxiv (2022). https://doi.org/10.1101/2022.07.16.500294
2. Kitsak, M., Papadopoulos, F., Krioukov, D.: Latent geometry of bipartite networks. Phys. Rev. E **95**(032), 309 (2017)
3. Pavlopoulos, G.A., Kontou, P.I., Pavlopoulou, A., Bouyioukos, C., Markou, E., Bagos, P.G.: Bipartite graphs in systems biology and medicine: a survey of methods and applications. GigaScience **7**(4), giy014 (2018)
4. Pastor, J.M., Santamaría, S., Méndez, M., Galeano, J.: Effects of topology on robustness in ecological bipartite networks. Netw. Heterogeneous Media **7**(3), 429–440 (2012). https://doi.org/10.3934/nhm.2012.7.429
5. Dunne, J.A., Williams, R.J., Martinez, N.D.: Food-web structure and network theory: the role of connectance and size. Proc. Natl. Acad. Sci. U.S.A. **99**, 12917–12922 (2002)
6. Goh, K.I., Choi, I.G.: Exploring the human diseasome: the human disease network. Brief. Funct. Genom. **11**(6), 533–542 (2012)
7. Lamb, D.G., Calabrese, R.L.: Small is beautiful: models of small neuronal networks. Curr. Opin. Neurobiol. **22**(4), 670–675 (2012). https://doi.org/10.1016/j.conb.2012.01.010
8. Bal, T., Nagy, F., Moulins, M.: The pyloric central pattern generator in crustacea: a set of conditional neural oscillators. J. Comp. Physiol. A **163**, 715–727 (1996)
9. Marder, E., Calabrese, R.: Principles of rhythmic motor pattern generation. Physiol. Rev. **76**, 687–717 (1996)
10. Marder, E., Bucher, D.: Central pattern generators and the control of rhythmic movements. Curr. Biol. **11**(23), R986–R996 (2001)
11. Rybak, I.A., Dougherty, K.J., Shevtsova, N.A.: Organization of the mammalian locomotor CPG: review of computational model and circuit architectures based on genetically identified spinal interneurons. eNeuro (2015). https://doi.org/10.1523/ENEURO.0069-15.2015
12. Bidaye, S.S., Bockemühl, T., Büschges, A.: Six-legged walking in insects: how CPGS, peripheral feedback, and

descending signals generate coordinated and adaptive motor rhythms. J. Neurophysiol. **119**(2), 459–475 (2018). https://doi.org/10.1152/jn.00658.2017

13. Ashwin, P.: Symmetric chaos in systems of three and four forced oscillators. Nonlinearity **3**(3), 603–617 (1990)
14. Ashwin, P., Burylko, O., Maistrenko, Y.: Bifurcation to heteroclinic cycles and sensitivity in three and four coupled phase oscillators. Physica D **237**(4), 454–466 (2008)
15. Rosenblum, M.G., Pikovsky, A.S., Kurths, J.: Phase synchronization in driven and coupled chaotic oscillators. IEEE Trans. Circuits Syst. I Fund. Theory Appl. **44**(10), 874–881 (1997)
16. Barrio, R., Lozano, Á., Rodríguez, M., Serrano, S.: Numerical detection of patterns in CPGs: gait patterns in insect movement. Commun. Nonlinear Sci. Numer. Simul. **82**(105), 047 (2020)
17. Barrio, R., Lozano, Á., Martínez, M., Rodríguez, M., Serrano, S.: Routes to tripod gait movement in hexapods. Neurocomputing **461**, 679–695 (2021)
18. Ghigliazza, R., Holmes, P.: A minimal model of a central pattern generator and motoneurons for insect locomotion. SIAM J. Appl. Dyn. Syst. **3**(4), 671–700 (2004)
19. Chun, C., Biswas, T., Bhandawat, V.: Drosophila uses a tripod gait across all walking speeds, and the geometry of the tripod is important for speed control. eLife **10**, e65878 (2021)
20. Ramdya, P., Thandiackal, R., Cherney, R., Asselborn, T., Benton, R., Ijspeert, A., Floreano, D.: Climbing favours the tripod gait over alternative faster insect gaits. Nat. Commun. **8**(14), 494 (2017)
21. Ghigliazza, R.M., Holmes, P.: Minimal models of bursting neurons: how multiple currents, conductances, and timescales affect bifurcation diagrams. SIAM J. Appl. Dyn. Syst. **3**(4), 636–670 (2004). https://doi.org/10.1137/030602307
22. Halton, J.H.: Algorithm 247: radical-inverse quasi-random point sequence. Commun. ACM **7**(12), 701–702 (1964)
23. Hairer, E., Nørsett, S., Wanner, G.: Solving Ordinary Differential Equations I Nonstiff problems, 2nd edn. Springer, Berlin (2000)
24. Prince, P., Dormand, J.: High order embedded Runge–Kutta formulae. J. Comput. Appl. Math. **7**(1), 67–75 (1981)
25. Alcalde Cuesta, F., González Sequeiros, P., Lozano Rojo, A., Vigara Benito, R.: An accurate database of the fixation probabilities for all undirected graphs of order 10 or less. In: Rojas, I., Ortuño, F. (eds.) Bioinformatics and Biomedical Engineering, pp. 209–220. Springer, Berlin (2017)
26. Curto, C., Morrison, K.: Graph rules for recurrent neural network dynamics. Not. Am. Math. Soc. **70**(4), 536–551 (2023)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Discussion of Published Articles

In this chapter we discuss the published articles that are part of the doctoral thesis. Section 3.1 is devoted to the article **Dynamics of excitable cells: Spike-adding phenomena in action** [1] published in *SeMA Journal*. Section 3.2 focuses on the article **Deep Learning for chaos detection** [2], published in *Chaos: An Interdisciplinary Journal of Nonlinear Science*. In Section 3.3, the article **Full Lyapunov exponents spectrum with Deep Learning from single-variable time series** [3], which has been published in *Physica D: Nonlinear Phenomena*, is discussed. Finally, the discussion of the article **Dominant patterns in small directed bipartite networks: Ubiquitous generalized tripod gait** [4] published in *Nonlinear Dynamics* is in Section 3.4.

3.1 Dynamics of Excitable Cells: Spike-Adding Phenomena in Action [1]

Our objective in the article ***Dynamics of excitable cells: Spike-adding phenomena in action*** *(SeMA Journal) [1] is to study the dynamics (in particular, the spike-adding phenomena) of excitable cells, such as neurons and cardiomyocytes (cardiac muscle cells), using standard techniques as spike-counting sweeping and numerical continuation of bifurcations. On the one hand, we perform a detailed analysis of the dynamics of the Hindmarsh-Rose model when the small parameter ε is varied. For standard values of ε, the numerical continuation of bifurcations allows us to describe the scenario of the spike-adding phenomenon. Moreover, geometric bifurcations help to understand dynamical changes when ε is varied out of that standard range. Finally, topological templates provide insight into the topological structure of the chaotic attractors of the system. On the other hand, we study the implications of the spike-adding phenomenon in more realistic models related to insect movement and cardiac dynamics.*

The Hindmarsh-Rose (HR) model [9] is a simplified neuron model derived from the Hodgkin-Huxley framework [8] that, while being computationally simple (and therefore suitable for dynamical analyses), provides a good qualitative description of the behavior of biological neurons. It is given by the following system of ordinary differential equations:

$$\left\{\begin{array}{rcl} \dot{x} & = & y - ax^3 + bx^2 - z + I, \\ \dot{y} & = & c - dx^2 - y, \\ \dot{z} & = & \varepsilon[s(x - x_0) - z], \end{array}\right.$$

where x is the membrane potential, y and z are the fast and slow gating variables (that represent the dynamics of ion channel gates that control the opening and closing of specific ion channels), and $(a, b, c, d, I, s, x_0, \varepsilon)$ are the parameters. In particular, ε is the so-called small parameter.

In neuron models as the HR model, one of the main dynamical behaviors is bursting (see Figure 1 in [1] for an example of fold/hom or square-wave bursting), which follows a fast-slow dynamics. The spike-adding phenomenon, that produces the appearance of extra spikes, is present in systems with this type of dynamics (see Figure 2 of [1]). In the Hindmarsh-Rose model, the variation of the value of the small parameter ε allows us to perform a global study of its dynamics.

For standard small values of parameter ε, the use of the spike-counting sweeping technique [12] (which counts the number of spikes of an orbit during one period in the membrane potential variable) allows us to clearly visualize the spike-adding phenomenon in the HR model (see Figure 3 in [1]).

The numerical continuation of bifurcations (using the software AUTO [102, 103]) provides insight into this phenomenon. In Figure 4 of [1], the main bifurcations involved in it are depicted in the (b,I)-parametric plane: primary homoclinic bifurcations, saddle-node of periodic orbits, period-doubling, and codimension-two bifurcation points (inclination-flip and orbit-flip). A detailed study permits us to describe the spike-adding process using such bifurcations (see diagram of Figure 5 in [1]): the isolas of homoclinic bifurcations contain the codimension-two bifurcation points in which the pencils of the bifurcations resulting in the increase of one spike (n to $n+1$ spikes) are generated.

If we allow the value of parameter ε to vary beyond its standard range, we will obtain a global analysis of the parameter space. In Figure 7 of [1], the (b,I,ε)-parametric spike-counting sweep of the HR system is depicted. This analysis shows that when ε increases, some structures change or are not present anymore. In fact, with bifurcation continuation techniques (software AUTO [102, 103]) we can check that some codimension-two bifurcation points (inclination-flip and Belyakov) have disappeared. These phenomena can be explained with the so-called geometric bifurcations (non-topological bifurcations related to the way dynamics are represented) introduced in [22].

Notice that as we are working in the triparametric space, now the isolas of homoclinic bifurcations that are present in each plane for fixed values of ε (see Figure 5 in [1]) generate tubular homoclinic surfaces, and the codimension-two bifurcation points in the plane are bifurcation curves in the 3D space (see Figure 11 of [1]). The codimension $(2+1)$-(visible or invisible) geometric "fold" bifurcations can explain and clarify the disappearance of inclination-flip and Belyakov curves (see Figures 8 and 10 of [1]). The geometric bifurcation points in which the homoclinic surfaces have a maximum (see Figure 8 in [1]) are of "isola-type" with codimension $(1+1)$ and can explain the change in the stripes of the spike-adding cascade (different colors in the spike-counting sweep are visible in Figure 7 of [1] when ε increases). Finally, geometric bifurcations of codimension $(1+1)$-"saddle-type" are also present in the tubular homoclinic surfaces that for small values of ε have two disconnected isolas that give rise to only one isola when ε increases (see Figure 9 in [1]).

Finally, for the Hindmarsh-Rose model we can use topological templates [24] to study the topological structure of chaotic attractors when small parameter ε varies. As schematized in Figure 12 of [1], topological templates are obtained by projecting the three-dimensional flow onto the stable direction, which maintains the relative positions of the periodic orbits. In the system that concerns us, some orbits in the template are not present in the chaotic attractor (it is non-hyperbolic), although all the orbits in the attractor are in the template. Therefore, the template of the attractor is a subtemplate of the complete Smale horseshoe template in which paths corresponding to orbits that are not present in the attractor are closed. Considering that in the HR model the number of spikes increases as the value of ε approaches zero (see Figure 14 of [1]), and that traversing more spike-adding pencils results in fewer closed paths in the template, we can hypothesize that when ε decreases, the subtemplate is being filled. Consequently, in the limit case, it will correspond to the entire Smale horseshoe template (see Figure 15 of [1]).

The spike-adding phenomenon that we have visualized and studied in the Hindmarsh-Rose model using spike-counting sweeping techniques and numerical continuation of bifurcations is also related to real processes that can be investigated with more realistic Hodgkin-Huxley type models. Changes in insect movement patterns and the creation of Early Afterdepolarizations (cardiac dynamics) seem to be related to that phenomenon.

Rhythmic and coordinated biological processes as walking can be represented mathematically with small neuron networks known as Central Pattern Generators (CPGs). One particular case is the 24-dimensional CPG model described by Ghigliazza and Holmes [26, 27] that mimics the movement of cockroaches (hexapods). The study of changes in the movement of hexapods is of increasing interest from both biological and robotics perspective. If the dynamics of just one isolated neuron of the Ghigliazza and Holmes model is studied, we can observe bursting behavior and the spike-adding phenomenon (see Figure 17 of [1]). We choose a one-parameter line in the region where bursting occurs in the single neuron case to perform the study of the six-neuron network (the whole CPG). In addition to using the spike-counting sweeping technique, the quasi-Monte Carlo sweeping method is applied to obtain all the possible gait patterns for each parameter value (and the frequency with which each one occurs). In Figure 18 of [1] the results are depicted alongside the main movement patterns. It is clear that spike-adding regions correspond to zones where the smoothness of pattern transition is affected. This relates the dynamical spike-adding process and the changes in insect movement gaits. Another noteworthy result is that the tripod gait (see Figure 16 of [1]) is always present (unstable for some parameter values, and dominant when stable).

The electrical signals generated in a cardiomyocyte (cardiac muscle cell) as a result of changes in the membrane potential are known as action potentials. When the membrane potential unexpectedly increases during the phase 2 or the phase 3 of the action potential, an Early Afterdepolarization (EAD) occurs (see Figure 19 in [1]). Early Afterdepolarizations with enough magnitude that take place in a large tissue area can produce some dangerous phenomena, such as arrhythmias [18]. Therefore, the analysis of the creation of EADs is relevant. Notice that the creation of Early Afterdepolarizations is what we have called spike-adding phenomenon (term used in neuroscience) so far.

To tackle the study of the creation of EADs, we consider two mathematical models of cardiac muscle cells: the realistic Sato model [104] with 27 variables, and the 3D Luo-Rudy model [105, 106] with 3 variables. As both represent similar behaviors, we study the high-dimensional Sato model to conjecture what is happening, and then, with the low-dimensional one we check the conjecture. In Figure 20 of [1], the ratio between the number of peaks and action potentials in a biparametric plane and a one-parameter bifurcation diagram are depicted for the Sato model. With these results, a possible scheme for the creation of EADs (spike-adding phenomenon) is conjectured. Using numerical continuation (software AUTO [102, 103]) in the simpler model (3D Luo-Rudy model), the conjecture is verified: a periodic orbit suffers a bifurcation that permits the appearance of alternans that evolve, potentially presenting EADs.

In this paper [1], some techniques as spike-counting sweeping, numerical continuation of bifurcations and topological templates have allowed us to obtain a global overview of the dynamics of mathematical models and to explain the spike-adding phenomenon that is present in excitable cells. Moreover, the analyses carried out in this article can be reproduced to study other models.

3.2 Deep Learning for Chaos Detection [2]

*Our objective in the article **Deep Learning for chaos detection** (Chaos: An Interdisciplinary Journal of Nonlinear Science) [2] is to determine if Deep Learning (DL) can be used to perform chaos detection analyses in the parameter space of a dynamical system (binary classification from the point of view of DL). We show the strengths and limitations of this new method and compare it with classical techniques. In particular, we consider a classical discrete dynamical system, the Logistic map, and a continuous one, the Lorenz system. For both dynamical systems, we compare the performance of three well-known DL Artificial Neural Networks (ANNs): the Multi-Layer Perceptron (MLP), the Convolutional Neural Network (CNN), and the Long Short-Term Memory (LSTM) cell. To show that our results are independent of the initialization of the trainable parameters of the networks, for each architecture (MLP, CNN and LSTM), we present most of the results in the format mean±standard deviation of* 10 *trained networks randomly initialized. All our experiments are performed using PyTorch [107].*

The Logistic map [68] is a classical one-dimensional discrete dynamical system that, despite of its simple formulation, presents complex dynamics, such as chaos and period-doubling bifurcations. The equation of the Logistic map is

$$x_{n+1} = \alpha x_n(1-x_n), \tag{3.1}$$

with x_n the variable at discrete time n, and α the bifurcation parameter.

Our goal is to obtain a DL-based tool such that given whatever time series from the Logistic map, it will be able to classify it as regular or chaotic. To do this, after the selection of an appropriate architecture (number of layers in the MLP, kernel size in the CNN, dimension of the states in the LSTM, etc.), we have to train the network (in a supervised way). The data used for training is derived from the Logistic map ($x_0 \in \{0.5, 0.9\}$, $\alpha \in [0,4)$), and after a data selection process, the training set comprises

4000 time series, each of length 1000, with half exhibiting regular behavior and the other half chaotic behavior.

We use the trained networks to perform a chaos detection study in an α-parametric line with $x_0 = 0.6$ and $\alpha \in [0,4)$. In Figure 3 of [2], we have graphically represented the analyses obtained with classical [77] and each DL technique (MLP, CNN and LSTM). It can be seen that the boundaries between regular and chaotic regimes have been properly detected by all the networks. In particular, for all the architectures and for each dynamical behavior, the mean accuracy is always greater than 98% and standard deviation is quite small (see table at the bottom of Figure 3 in [2]). So, more than 98% of the regular and the chaotic samples have been correctly classified independently of the used network.

Moreover, it is interesting to analyze the detection given by the networks for some time series of such α-parametric line. The results are in Figure 5 of [2]. In particular, we show the classification given by each DL technique in two regular (examples (I) and (II)) and two chaotic samples (examples (III) and (IV)). The time series (I) and (IV) are properly classified by all the DL architectures. However, DL encounters problems in classifying sample (II), which is misclassified by all the architectures, and sample (III), which is only detected correctly as chaotic by the CNN. These fails can be expected because of the dynamics of the samples: sample (II) corresponds to a periodic orbit with many oscillations, and sample (III) presents weak chaos (small irregularity). They are boundary orbits that would also be complicated to classify with standard techniques. From Figure 4 of [2], it could be deduced that, although with more than 90% of success, DL presents more problems in the classification of orbits with Lyapunov exponent (LE) value close to 0.

The Lorenz system [63] is the classical example of a continuous dynamical system with chaotic dynamics. The system of ordinary differential equations of the Lorenz model is

$$\left\{\begin{array}{lcl} \dot{x} & = & \sigma(y-x), \\ \dot{y} & = & -xz+rx-y, \\ \dot{z} & = & xy-bz, \end{array}\right. \tag{3.2}$$

where (x,y,z) are the variables, and (σ,r,b) are the parameters. The parameter σ is the Prandtl number, r is the relative Rayleigh number, and b is a positive constant. The classical values of the Lorenz chaotic attractor are $(\sigma,r,b) = (10,28,8/3)$.

As with the Logistic map, we want a DL-based method that, once trained on the Lorenz system, it will be able to perform chaos detection in this continuous dynamical system. In fact, as the Lorenz system has a three-dimensional parameter space, we want to obtain a global analysis of the system just training the DL networks in a small region of such parameter space. In particular, training data consists of 7800 time series (half with regular behavior and the other half with chaotic behavior) of length 1000 from two r-parametric lines with $\sigma = 10$, $b \in \{2,8/3\}$ and $r \in (0,300]$.

As a first test study, in Figure 8 of [2], we show the results obtained when the trained ANNs are used to detect chaos in an r-parametric line ($\sigma = 10$, $b = 2.2$, $r \in [0,300]$) not involved in the DL supervised training process. From the graphical representation of the study, we can deduce that the MLP, even providing reasonable results, does not seem to properly detect the behavior of some zones. The CNN and the LSTM give similar chaos detection analyses (which are better than the one of the MLP), with the difference that the LSTM seems to be more accurate (considering as ground truth the classical LEs [58]) in the most-right boundary between chaotic and regular behavior. These conclusions are supported by the numerical results given in the bottom table of the figure as the accuracy is greater for the LSTM network (mean accuracy for regular is 96.864% and for chaotic is 98.314%). Analyzing the performance according to the first LE value approximated by classical techniques (see Figure 9 in [2]), as occurred in the Logistic map, lower accuracy in the dynamical classification with DL is obtained for samples with a Lyapunov exponent value in a neighborhood of 0.

We use the trained networks to perform the chaos detection analysis of the (r,b)-parametric plane ($\sigma = 10$, $b \in [2,3]$, $r \in [0,300]$) that contains the two r-parametric lines used to train the networks. In Figure 11 of [2], we plot the biparametric analyses obtained with classical [58] and DL techniques (MLP, CNN and LSTM), and the errors made by each network with respect to the LEs classification (ground truth). The accuracy is greater than 94% for all the ANNs, but as already concluded from the r-parametric analysis, the CNN and the LSTM perform better than the MLP. In particular, the region where DL fails the most is the right boundary between chaotic and regular behaviors, with the LSTM defining the boundary between both dynamical regimes much more clearly. In this zone, it occurs a phenomenon know as transient chaos [108, 109], which makes detection challenging for any technique, not just DL.

The LSTM network seems to outperform the other DL methods (the MLP and the CNN), so we continue the global analysis of the parameter space of the Lorenz system with this architecture. In fact, three other parametric planes are studied: (r,σ)-parametric plane for $b=8/3$, $r\in[1,350]$ and $\sigma\in[0,60]$ (see Figure 12 in [2]); (σ,b)-parametric plane with $r=28$, $\sigma\in[0,40]$ and $b\in[0,4]$ (see Figure 13 in [2]); (σ,b)-parametric plane with $r=500$, $\sigma\in[0,800]$ and $b\in[0,100]$ (see Figure 13 in [2]). Notice that in these parametric planes, only a few samples belong to the r-parametric lines used to create train dataset. The results are quite impressive as the accuracy is larger than 97.5% in the three biparametric studies. In fact, an analysis of some time series (see Figure 14 in [2]) allows us to check that some of the misclassifications can be explained due to the nature of the corresponding orbits that usually are in boundary regions where bifurcations and complex dynamics can be found.

To complete the global analysis of the Lorenz system, the LSTM trained in just two r-parametric lines is used to study the triparametric space, performing a dense 3D analysis with $\sigma\in[0,800]$, $r\in[1,500]$, and $b\in[0,100]$. To the best of the authors' knowledge, a dense 3D analysis of the Lorenz system had not been provided previously in the literature. From Figures 15 and 16 of [2], where this analysis is depicted, it can be concluded that the boundary between regular and chaotic regimes is perfectly defined with Deep Learning. In fact, 99.661% of accuracy is achieved.

In addition to the good performance in the chaos detection problem, another advantage of using DL instead of LEs classical technique [58] is time savings (see Table VI in [2]). A biparametric analysis of the Lorenz system with 1000×1000 points takes around 30 minutes with the LSTM network (just 17 seconds are devoted to chaos detection once the time series are computed), while approximately 25 hours are needed when using the classical method (long time is needed by LEs to converge). In the dense triparametric analysis with $250\times250\times250$ points, time savings with DL are even more remarkable: 16 days are needed with classical techniques, while around half day is used when the LSTM is used (just 20 minutes are devoted for the chaos detection once time series are obtained).

From this paper [2], it can be concluded that Deep Learning is a promising technique for dynamical systems behavior analysis. The three proposed architectures (MLP, CNN and LSTM) give similar results in the Logistic map. However, in the case of Lorenz model, the LSTM is the best option. In fact, training with data from a small region of the parameter space of the Lorenz system, a complete analysis (including a dense 3D study) of chaos detection can be performed in this system using DL. Furthermore, time savings achieved with this new powerful tool are considerable: just around 3% of the total time required by classical technique of Lyapunov exponents is devoted to obtain the dense triparametric study with DL.

3.3 Full Lyapunov Exponents Spectrum with Deep Learning from Single-Variable Time Series [3]

Our objective in the article ***Full Lyapunov exponents spectrum with Deep Learning from single-variable time series*** *(Physica D: Nonlinear Phenomena) [3] is to show that Deep Learning techniques are able to provide a good approximation of the full Lyapunov exponents spectrum of a dynamical system with just information from one variable. This method also allows us to detect dynamical behaviors as hyperchaos or tori using partial information. In fact, we consider two different approaches, with non-random and with random data. The non-random approach uses LEs approximations of a few one-parameter lines obtained with classical techniques [58] to train the DL network and expand the partial classical analysis. The random case trains the networks using random data from a biparametric plane. We use both approaches in the well-known Lorenz system. To increase the dimensionality and check* *the success of Deep Learning for Lyapunov exponents approximation, we couple two almost identical Lorenz systems to obtain a six-dimensional dynamical system referred to as coupled Lorenz system. The value of the Lyapunov exponents given by DL is computed as the mean of the approximation provided by* 10 *networks with the same architecture but randomly initialized, and is compared to the approximation given by the classical technique described in [58]. All DL experiments are carried out using PyTorch [107].*

On the one hand, one of the key points of the Deep Learning implementation for this LEs prediction task is the selection of the loss function (that is minimized during training process). Notice that in the Lyapunov exponents approximation, the critical values are those close to zero: a significant error in the prediction of an LE in a neighborhood of zero can result in an incorrect classification of the corresponding orbit. So, to try to mitigate the error for these values, we use the Huber loss function [110] that is less sensitive to outliers than the widely used Mean Squared Error (MSE) loss. On the other hand, it is remarkable that the same CNN architecture is used independently of the studied dynamical system, except for the output layer that has as many neurons as Lyapunov exponents that have to be approximated.

The Lorenz system [63] is a three-dimensional continuous dynamical system given by equation 3.2. In the literature, several LEs analyses have been performed in this system using classical techniques [54, 55]. For the non-random DL approach in this model, we take randomly 8000 time series of length 1000 from two r-parametric lines ($\sigma = 10$, $b \in \{2, 8/3\}$, $r \in (0, 300]$) to train the network. The unique information provided to the CNN is the x-time series of each sample. With the trained CNN, we perform a one-parameter analysis for $\sigma = 10$, $b = 2.2$ and $r \in (0, 300]$. Figure 2 of [3] shows the mean LEs approximation obtained with DL to compare its performance with the approximation provided by classical techniques [58]. In general, the CNN results are quite accurate. However, this new technique seems to fail in two zones of this r-parametric line: for small and for large values of r. The dynamics corresponding to small values of r are equilibrium points. For the proper functioning of Deep Learning, input information has to be normalized. The equilibrium points are constant time series that, according to our normalization rule, are normalized to a random value in the interval $[0, 1]$. This randomness makes learning difficult, and the network is only able to recognize the behavior and assigns a negative almost constant value to all the equilibrium points. The region for large value of r exhibits long transient chaotic dynamics [108, 109], that already presented problems in the chaos detection task [2] with short time series.

Finally, the same CNN trained with non-random data is used to approximate the three LEs in an (r, b)-parametric plane with $\sigma = 10$, $r \in [0, 300]$, and $b \in [2, 3]$. Notice that the r-parametric lines from which the training data is obtained are contained within this plane. Therefore, as already mentioned, we are expanding such classical partial study to the whole biparametric plane. In Figure 3 of [3], we provide the LEs analyses performed using classical and DL techniques, along with the signed difference between both approaches to give an intuition of the error location in the plane. Despite using just short single-variable time series and no additional dynamical information, the results are quite good. As expected after the one-parameter study, the worst approximations correspond to the right boundary between chaotic and regular dynamics where chaotic transient occurs. In Figure 4 of [3], with an absolute error bar plot, we can check the error magnitude according to the classical LEs value. The lowest errors correspond to the first LE, however the approximations of the remaining two exponents are quite accurate taking into account the demanding task. It is remarkable, that as desired, first and second Lyapunov exponents whose classical value is close to 0 are properly approximated by DL.

For the random approach in the Lorenz system, we consider that we do not have any previous classical LEs analyses. Therefore, to train the network, we generate some data (8000 samples) randomly distributed in the (r, b)-parametric plane we want to study with DL ($\sigma = 10$, $r \in [0, 300]$, $b \in [2, 3]$). We use this new trained CNN to reproduce the one-parameter analysis already carried out with the non-random approach (see Figure 5 in [3]). As more dynamical variability is provided to the network while training because of random selection of training data, Lyapunov exponents approximations are more accurate with the DL random approach than with the non-random case. Although this new trained CNN still presents some problems in the transient chaotic region, the approximations are considerably better. If we expand the study to the biparametric plane (see Figures 6 and 7 in [3]), in the graphical representation of the LEs approximation it is clear that this random approach presents better results than the non-random approach. In fact, the analysis provided for the second Lyapunov exponent even permits us to detect some dynamical changes and bifurcation lines. Comparing the signed differences (Figures 3 and 6 of [3]) and the absolute error bar plots (Figures 4 and 7 of [3]) between both approaches, it can be deduced that in general the sign of the difference between the classical and the corresponding DL technique is the same for both approaches, but the error magnitude is considerably reduced when random data is used for training.

The coupled Lorenz system [78] is a six-dimensional dynamical system obtained coupling two almost identical Lorenz systems as follows:

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

with $(x_1, y_1, z_1, x_2, y_2, z_2)$ the variables of the system, (σ, r_1, r_2, b) the bifurcation parameters, and (λ_1, λ_2) the coupling parameters. For our analyses we consider the following relation for the bifurcation parameters r_1 and r_2: $r = r_1 = r_2 - 10$. In [78], the attractors of such coupled system are analyzed when $r_1 = r_2$.

For this coupled system, the non-random and the random approaches are also considered to train the CNN. In both cases, just x_1-variable time series of length 1000 are used. In particular, for the non-random case, equivalent to the Lorenz system, 8000 samples from two r-parametric lines ($\sigma = 10$, $b \in \{2, 8/3\}$, $r \in (0, 300]$, $\lambda_1 = \lambda_2 = 0.1$) comprise the training dataset. For the random case, the same number of samples are randomly selected from the biparametric plane with $\sigma = 10$, $r \in [0, 300]$, $b \in [2, 3]$ and $\lambda_1 = \lambda_2 = 0.1$ to create such dataset. Using the trained networks with each approach, the Lyapunov exponents of an r-parametric line ($\sigma = 10$, $b = 2.2$, $r \in [0, 300]$, $\lambda_1 = \lambda_2 = 0.1$) are approximated (see Figure 8 in [3]). Both approaches provide good approximations for the six Lyapunov exponents despite the complicated task (the unique information provided as input to the network are short single-variable time series). However, thanks to the dynamical variability in training set, better results are obtained with the random approach. As expected, both techniques fail in the approximation of LEs values for equilibrium points (corresponding to small values of parameter r) because of the normalization rule. If we expand the use of both DL approaches to the (r, b)-parametric plane ($\sigma = 10$, $r \in [0, 300]$, $b \in [2, 3]$, $\lambda_1 = \lambda_2 = 0.1$), good LEs analyses are obtained (see Figures 9, 10, 11 and 12 in [3]). As in the Lorenz system case, the boundary regions are better defined with the random approach. Furthermore, although the sign of the difference between the classical and the corresponding DL technique is the same for both DL approaches, the error magnitude is much smaller in the random approach.

As good approximations of all the LEs of the coupled Lorenz system are obtained with Deep Learning techniques, we use the CNN approximations of the random approach in the biparametric plane to carry out a dynamical classification. In Figure 13 of [3] we can see that the main dynamical regimes of the plane (tori, hyperchaos and chaos) can be correctly detected when using the LEs approximations provided by DL. Remember that the LEs approximations of DL have been obtained using just single-variable time series and these predictions have allowed us to detect hyperchaotic and tori zones that with standard techniques would not be possible in such conditions. To the best of the authors' knowledge, no other algorithm can obtain an approximation of all the LEs of a dynamical system with only single-variable time series.

One of the main advantages of using Deep Learning to obtain the full Lyapunov exponents spectrum is time savings. In particular, for the Lorenz system, while the classical technique [58] needs around

25 hours to obtain the biparametric analysis, the whole DL process (from the training of the network to the completion of the biparametric study) takes less than 2 hours. In fact, time devoted to obtain the LEs approximation of all the plane once the network is trained and the time series are computed is of 3 seconds. For the coupled Lorenz system, the time savings are similar: with classical techniques 54 hours are used, while the whole DL process needs less than 2 hours and 30 minutes (again 3 seconds are devoted to the approximation part).

From this paper [3], it can be concluded that Deep Learning is a simple but powerful technique that allows us to approximate the full Lyapunov exponents spectrum of a dynamical system using just single-variable time series without any other dynamical information. To the best of the authors' knowledge, this has not been achieved previously with this or other technique. Furthermore, Deep Learning requires less than 10% of the time needed by classical techniques to obtain a biparametric analysis. Both proposed approaches (non-random and random) applied to the Lorenz system and the coupled Lorenz system have demonstrated good performance in approximating all the LEs, failing only in expected regions. However, the results are more accurate in the random case due to the greater dynamical variability provided during training.

3.4 Dominant Patterns in Small Directed Bipartite Networks: Ubiquitous Generalized Tripod Gait [4]

Our objective in the article ***Dominant Patterns in Small Directed Bipartite Networks: Ubiquitous Generalized Tripod Gait*** *(Nonlinear Dynamics) [4] is to analyze if the generalized tripod gait (bipartite pattern) is the dominant synchronization pattern in Central Pattern Generators (CPGs) with bipartite connectivity (bipartite networks). This would provide a link between dynamics and network topology. To do so, we reinterpret Central Pattern Generators (CPGs) with bipartite connectivity (and bidirectional connections) as bipartite graphs. This allows us to define synchronization patterns as q-colorings of the graph (for some $q \in \mathbb{N}$), and therefore, it permits us to establish a partition of the set of neurons. Considering all the possible configurations of the bipartite CPGs of* 6 *to* 9 *neurons, and using the quasi-Monte Carlo sweeping technique [28, 89], we study the possible patterns, and consequently, the dominant one.*

Central Pattern Generators are small neuron networks that once activated are capable of generating rhythmic motor patterns even in absence of sensory input. We consider CPGs of 6 to 9 neurons (with inhibitory synapses) generalizing the bursting neuron CPG model of Ghigliazza and Holmes [26, 27] that replicates the movement of cockroaches. For neuron i, the corresponding ordinary differential equations are

$$\begin{cases} C\dot{v}_i &=& -(I_{Ca}(v_i)+I_K(v_i,m_i)+I_L(v_i)+I_{KS}(v_i,w_i))+I_{ext}-(I_{syn})_i(v_i,s_j), \\ \dot{m}_i &=& \dfrac{\varepsilon}{\tau_m(v_i)}[m_\infty(v_i)-m_i], \\ \dot{w}_i &=& \dfrac{\delta}{\tau_w(v_i)}[w_\infty(v_i)-w_i], \\ \dot{s}_i &=& \alpha s_\infty(v_i)(1-s_i)-\beta s_i, \end{cases}$$

with v_i the action potential, m_i the potassium gate, w_i the slow potassium gate, and s_i the synapsis variable. The remaining functions and parameters can be consulted in [4]. Therefore, for a network of N neurons we get a $4N$-dimensional system of ordinary differential equations.

These neuron networks are reinterpreted as undirected, connected and without loops graphs in which each vertex corresponds to one of the neurons, and each edge represents two opposed synapses. As we want to study if the bipartite pattern is ubiquitous in bipartite networks, we consider neuron network configurations whose underlying graph is bipartite (2-colorable), that is, vertices can be divided into two groups with adjacent vertices belonging to opposing groups. Such decomposition defines a 2-coloring of the graph. A synchronization pattern in a neuron network is called a bipartite (or 2-color) pattern if the natural partition defined by this pattern forms a 2-coloring of the graph.

To examine if the ubiquity of the bipartite pattern holds for the 982 bipartite bidirectional networks with up to 9 neurons, we fix all the model parameters to a specific point in the parameter space where the standard Ghigliazza and Holmes network (a CPG with 6 neurons) [28] exhibits the tripod gait (bipartite pattern) as the dominant pattern (see [4] for the concrete values of such parameters). Then, for each bipartite network, the quasi-Monte Carlo sweeping technique [28, 89] is used to compute the percentage of initial conditions evolving to each synchronization pattern. The idea of quasi-Monte Carlo sweeping technique is to select 200 initial conditions using Halton sequences [111], to integrate numerically the neuron network model, to preprocess the numerical solutions to obtain a pattern, and finally to divide the 200 patterns into equivalence classes and compute their size. Figure 3 of [4] illustrates the different patterns that are present in a network with a specific topology.

After such analyses, it becomes clear that for the considered parameter values, the bipartite patterns (generalized tripod gait) are dominant. In particular, we can define the 100/70 and the 95/90 rules. The first rule states that in 100% of the graphs, the bipartite pattern has a prevalence larger than 70%; and in 70% of them approximately, the bipartite pattern appears for all the sampled initial conditions (100%). The 95/90 rule establishes that for about 95% of the graphs, the bipartite pattern is present in more than 90% of the simulations; and for around 90% of the graphs, in at least 95% of the initial conditions the dynamics evolves to the bipartite pattern.

It is also interesting to investigate which are the non-bipartite patterns that appear in the simulations of each network with a statistical relevance of more than 5%. In the 982 considered neuron networks, 71 have one non-bipartite pattern, and 12 have two (we have a total of 95 non-bipartite patterns in our simulations). The underlying graphs of these 83 neuron networks with at least one non-bipartite pattern are depicted in Figure 4 of [4]. Among the 95 patterns that are not 2-color patterns, 87 are 3-color patterns (vertices can be divided into three groups, such that adjacent vertices do not belong to the same group), and the remaining 8 are not color patterns. Figures 5 and 6 in [4] illustrate such non-color patterns. The cases in the latter figure need further research as connected neurons fire at the same time and this seems to contradict the use of inhibitory synapses.

This procedure, that permits us to show the ubiquitous of the bipartite pattern (tripod-like pattern) in bipartite networks, can be generalized to study other kind of patterns and topologies. In particular, in [4] it is investigated if the predominant patterns in (non-bipartite) cyclic graphs are the star traveling wave patterns, or if in (non-bipartite) 3-color networks, the ubiquitous pattern is the 3-color pattern. See Figure 8 in [4] for an example of a star traveling wave pattern in a cyclic graph, and Figure 11 for an instance of a 3-color patterns in a 3-color network. For the cyclic networks, when they are not bipartite, the predominant pattern is the expected one (star traveling wave). In the (non-bipartite) 3-color networks, the emergence of the 3-color pattern is not as significant as the bipartite pattern in bipartite networks.

From this paper [4], it can be concluded that the bipartite pattern (generalized tripod gait) is ubiquitous in bipartite networks, so there is a strong relation between network topology and dynamics. These analyses have been possible by reinterpreting neuron networks as graphs and using the quasi-Monte Carlo sweeping technique. As shown for other topologies as cyclic graphs, this method can be applied to study the dominance of patterns in non-bipartite graphs too.

Unpublished Articles

This chapter includes three unpublished articles (Preprints) provided as additional material for the doctoral thesis of Carmen Mayora Cebollero:

- [5] C. Mayora-Cebollero, F.H. Fenton, M. Halprin, C. Herndon, M.J. Toye, and R. Barrio, “Deep Learning for analyzing chaotic dynamics in biological time series: Insights from frog heart signals”, *Preprint*, 2024.
- [6] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, “Graphical preprocessing techniques to generalize dynamical behavior analysis with Deep Learning”, *Preprint*, 2024.
- [7] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, “Bregman Proximal Gradient with extrapolation to train a Reservoir Computing network”, *Preprint*, 2024.

Deep Learning for Analyzing Chaotic Dynamics in Biological Time Series: Insights from Frog Heart Signals

Carmen Mayora-Cebollero[1], Flavio H. Fenton[2], Molly Halprin[2,3], Conner Herndon[2], Mikael J. Toye[2], Roberto Barrio[1]

[1]Departamento de Matemática Aplicada. IUMA. Computational Dynamics group. Universidad de Zaragoza. Zaragoza, Spain
[2]School of Physics, Georgia Institute of Technology. Atlanta, USA
[3]United States Patent and Trademark Office. Los Angeles, USA

Abstract

The analysis and study of experimental data and its classification as regular or chaotic dynamics is a relevant task in several physical, chemical, and biological systems, in particular in cardiac dynamics and its relation to some pathological arrhythmias. We develop an automatic algorithm that uses Deep Learning techniques combined with comparison with cardiac map features to verify validity. This technique is especially useful in the case of very short data where other techniques do not work properly. The algorithm is tested with a set of experimental data obtained from live frog heart experiments, obtaining highly accurate results.

1 Introduction

Nowadays, the analysis and study of experimental data is becoming more and more crucial to develop mathematical models and classify different possible behaviors. In data from real experiments, different features are common, such as different lengths in the time series and possibly a very short amount of data in most cases. In this study, we focus on a first attempt to automate the analysis of chaos in cardiac time series (in our case, frog heart dynamics data) using Deep Learning techniques and comparison with those of a periodically paced cardiac action potential map [1, 2], to verify validity. Chaos dynamics in the heart [3, 4, 5] has been proposed to exist at two levels, in the ECG for healthy individuals [6, 7] and in tissue level during fibrillation [8, 9] or fast pacing [10, 11], as well as higher order periods [12, 13]. Therefore the methods presented here could help in the analysis and classification of chaos at different regimes as well as different animal species, including human.

Classical techniques such as Lyapunov exponents [14, 15, 16] and the 0-1 test for chaos [17] are often used to perform chaos analysis of dynamical systems [18, 19, 20]. However, these techniques sometimes present problems when used in real data as recordings are generally short and noisy [21]. When dealing with large experimental datasets it is necessary to have an automatic algorithm for chaos analysis since human intervention on the entire dataset is not feasible. Recently, some authors have used Deep Learning (Artificial Neural Networks) to detect chaos in a dynamical system [22, 23, 24, 25]. Could Deep Learning (DL) be applied to perform a chaos analysis in an experimental dataset? Notice that a chaos analysis from the dynamical systems point of view consists of detecting if a time series has regular or chaotic behavior. From the DL perspective, it is a classification task.

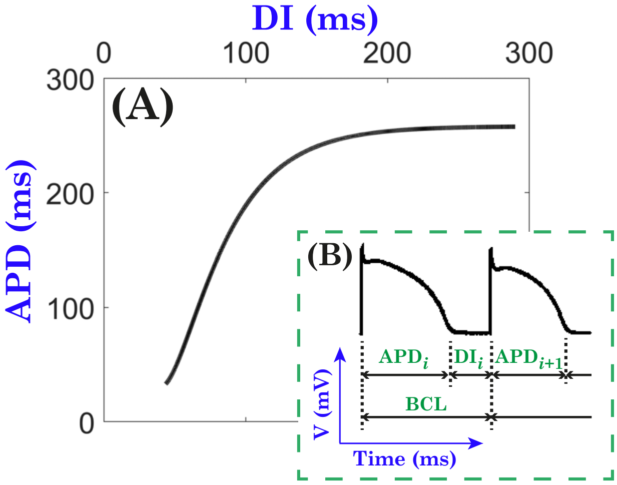


Figure 1: (A) APD restitution curve fitted to the kinetics of the Beeler-Reuter model. (B) Scheme to illustrate the APD, DI and BCL concepts.

During pacing of experimental or simulated a cardiac cell or tissue, trans-membrane voltage signals, called action potentials, can be obtained that look as shown in panel (B) of Figure 1. These signals can be characterized by an Action Potential Duration (APD), Diastolic Interval (DI), and Basic Cycle Length (BCL). The APD is the duration from the onset of cell depolarization to the completion of repolarization, given a certain threshold, where most commonly a value in between 75 to 90% of repolarization is used. The DI is then the time interval from the end of an APD to the next one, for the same threshold, which is basically the recovery time of the cell. The BCL is the time interval between consecutive sinus beats or the pacing cycle length used to stimulate the heart when experiments are carried out.

Using the APD and DI data, an APD restitution curve can be constructed to predict the APD given a particular period of stimulation. This is a one-dimensional map that has been shown useful to predict complex dynamics [26] and which it has been shown experimentally to predict period-doubling bifurcations [27]. Usually, it is a monotonically increasing function, but some experiments have shown that it could be biphasic [28, 29], which could lead to chaotic dynamics [30, 31, 32]. Mathematically, it is represented with a discrete equation of the form $\mathrm{APD}_{i+1} = f(\mathrm{BCL} - \mathrm{APD}_i) = f(\mathrm{DI}_i)$. In panel (A) of Figure 1 we have the APD restitution curve fitted to the kinetics of the Beeler-Reuter model [1]. The APD restitution curve can be more complex, being a function of the history of the stimulation pacing protocol due to memory [33, 34, 35, 36] and of calcium concentrations in the cells [37, 38].

Some recent studies have used Machine Learning techniques to predict chaotic time series from simulated and experimental data [39, 40, 41, 42], however, to our knowledge they have not been used for classification of chaos and Lyapunov exponents. In this paper, we combine the use of the classical Logistic map information [43] to train Artificial Neural Networks, and the Beeler-Reuter APD heart map model [1, 44] to verify the validity and select the most suitable Artificial Neural Network (ANN). With these elements we build an algorithm to detect chaos in experimental time series and we apply it to experimental data obtained from live frog hearts. An important point is to realize how the use of basic information data, i.e. the use of the basic and classical Logistic map as training data, combined with the a posteriori selection of the network using now a heart map (in this case) allows to correctly detect the dynamics. This is a de facto proof of the universality of the chaos dynamics information regardless of the problem being studied.

This paper is organized as follows. In Section 2, we describe the created DL algorithm to perform chaos analysis of heart time series (Subsections 2.1-2.4 are devoted to explain in detail all the steps of the algorithm, and in Subsection 2.5 we provide the pseudocode). In Section 3, we apply such algorithm to an experimental dataset obtained from frog heart dynamics (in Subsection 3.1 the approved animal protocol and experimental setup is described, and in Subsection 3.2 we show the performance of the algorithm in such dataset). Finally, in Section 4 we draw some conclusions.

PyTorch [45] has been used to perform all the DL experiments in this work.

2 DL Algorithm for Analyzing Chaotic Dynamics in Heart Time Series

In this section we propose a new algorithm to analyze chaotic dynamics in heart time series. The algorithm consists on four steps:

Step 1: Artificial Neural Networks Framework. 10 randomly initialized recurrence-like Artificial Neural Networks with the architecture proposed in [23] are trained during 2,000 epochs (early stopping and validation data are used) with time series of length 1,000 from the Logistic Map (data creation as explained in [23, 46]).

Step 2: DL Chaos Analysis of an APD Heart Map. A test analysis is performed in an APD heart map [1] with each Artificial Neural Network trained in *Step 1*.

Step 3: Selection of the Most Suitable Artificial Neural Network. Some criteria are established on the results of *Step 2* to detect automatically which is the most suitable network to perform the chaos analysis of heart time series.

Step 4: DL Chaos Analysis of Heart Time Series. Chaos analysis is performed with the network chosen in *Step 3* in the APD heart experimental data.

2.1 Step 1: Artificial Neural Networks Framework

Deep Learning (DL) is the part of Machine Learning that uses Artificial Neural Networks (ANNs) to learn from data with different levels of abstraction. Since the introduction of ANNs, several architectures have been proposed and widely used for several applications [47, 48, 49, 42]. One of these types is the Recurrent Neural Network (RNN) usually applied for sequential processing. Basic RNNs present several training problems as exploding/vanishing gradient and catastrophic forgetting. On the one hand, some authors have recently proposed alternative training algorithms based on dynamical systems theory to avoid gradient drawbacks [50]. On the other hand, to try to alleviate those problems, new architectures as Long Short-Term Memory (LSTM) cells [47] or Gated Recurrent Units (GRUs) [51] have been developed. For our purpose we will use an architecture based on stacked LSTM cells with a final classification layer.

A Long Short-Term Memory cell (Figure 2) is a DL architecture that processes information over time. At time step t, its inputs are an external data point $x(t)$ (usual input data of an ANN), a cell state $c(t-1)$, and a hidden state $h(t-1)$; and the outputs are the updated cell state $c(t)$, the updated hidden state $h(t)$, and the usual output $y(t)$ of an ANN (we take it equal to $h(t)$). The states $c(\cdot)$ and $h(\cdot)$ are devoted to keep information from previous time steps and are updated according to

$$\begin{aligned} c(t) &= f(t) \otimes c(t-1) + i(t) \otimes g(t), \\ h(t) &= o(t) \otimes \tanh(c(t)), \end{aligned}$$

where $\otimes$ is the element-wise product, and $f(t)$, $g(t)$, $i(t)$ and $o(t)$ are given by

$$\begin{array}{rcl} f(t) & = & \sigma(W_f^{[x]}x(t) + W_f^{[h]}h(t-1) + b_f), \\ g(t) & = & \tanh(W_g^{[x]}x(t) + W_g^{[h]}h(t-1) + b_g), \\ i(t) & = & \sigma(W_i^{[x]}x(t) + W_i^{[h]}h(t-1) + b_i), \\ o(t) & = & \sigma(W_o^{[x]}x(t) + W_o^{[h]}h(t-1) + b_o). \end{array}$$

In these previous expressions, σ and tanh are the activation functions (sigmoid and hyperbolic tangent, respectively), and $W_*^{[\{x,h\}]}$ and b_* ($* \in \{f, g, i, o\}$) are the trainable parameters (weights and biases, respectively) of the network. Note that because of the application of the sigmoid activation function, f, i and o act as gates that screen the information and memory of the network.

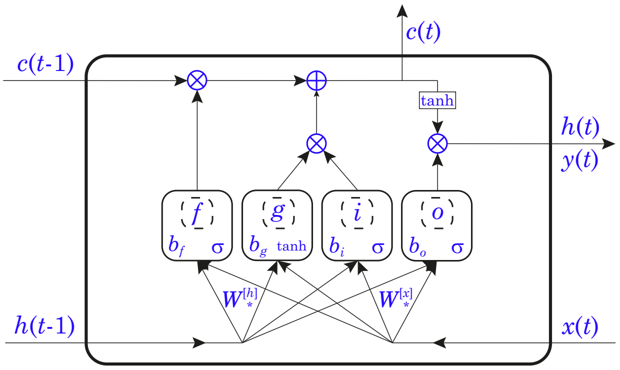


Figure 2: Scheme of an LSTM cell.

The ANN that we use for chaos analysis (introduced in [23]) has two stacked trainable LSTM cells (both are unidirectional with bias term and states of dimension 4) followed by a trainable linear classification layer of two neurons (one for each class: regular and chaotic) whose input is the last hidden state of both LSTM cells. Finally the softmax activation function is applied to the output of the classification layer (to transform output values to scores, as usual for classification DL tasks).

To fit the aforementioned trainable parameters (weights and biases), the network has to be trained using data. It is trained for 2000 epochs using early stopping technique. Moreover, the standard cross-entropy loss function for classification tasks is used with L^2-regularization (weight decay 10^{-5}), and the Adam optimizer with learning rate 0.008 is applied. The used data is obtained from the Logistic map.

A key point related to the training process of the ANN is the selection of the training data. In our case, as the experimental data consists of short time series, we try to use basic and generic discrete data obtained from one of the most well-known discrete maps, the Logistic map [43]. One interesting fact in dynamical systems theory is the universality of numerous dynamical phenomena, like Feigenbaum constants [52], chaos dynamics and so on. Therefore, as training data we use generic data to not particularize too much the training process.

The Logistic map [43] is a well-known one-dimensional model that presents great dynamical richness (period-doubling bifurcation, chaos, etc.) despite its simplicity. It is given by

$$x_{i+1} = \alpha\, x_i\, (1 - x_i), \tag{1}$$

where x_i is the variable at the i-th iteration, and α is the bifurcation parameter. The Logistic map is constructed in such a way that the variable only takes values in the interval $[0, 1]$. As it is a one-dimensional map, the Lyapunov exponents (LEs) [14] can be easily computed applying equation

$$\text{LE} = \frac{1}{T} \sum_{j=1}^{T} \log |\alpha\, (1 - 2x_j)|, \tag{2}$$

with log the natural logarithm, and T large enough. An appropriate transient integration has to be computed before the application of the formula.

In panel (A) of Figure 3 we have represented the bifurcation diagram of the Logistic map when $\alpha \in [0, 4]$ and $x_0 = 0.5$. In panel (B), LEs have been depicted. In the bifurcation diagram the dynamics of the Logistic map can be seen clearly: for small values of α the dynamics converge into an equilibrium point, later there are a cascade of period-doubling bifurcations which result in chaotic dynamics for values of α close to 4. In the LE representation we can check that for regular behavior (equilibrium points and periodic orbits) the LE is negative or zero, and it is positive for chaos.

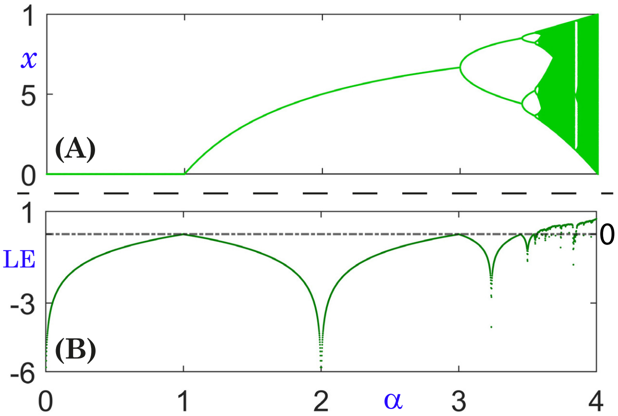


Figure 3: α-parametric line of the Logistic map ($x_0 = 0.5$). (A) Bifurcation diagram. (B) Lyapunov exponents.

To train the network with early stopping technique, the data is divided into three datasets: training dataset contains data to learn from, validation dataset prevents overfitting, and test set checks the performance of the trained network. Each dataset contains time series and the corresponding LE computed with classical techniques that would be used as label to check the behavior detected by the ANN. To obtain the time series, 12,000 time steps are computed with equation (1) and the last 1,000 correspond to the time series. To compute the LE, 12,000 time steps are obtained with equation (1), first 1,000 points are discarded as transient, and equation (2) is applied with the remaining ones.

To obtain train dataset, we create two raw datasets, one with initial condition $x_0 = 0.5$ and other with $x_0 = 0.9$. In both sets, the time series have length 1,000 and the parameter α takes 12,000 equidistant values in $[0, 4)$. Data is screened deleting similar samples (two time series are similar if its distance in infinity norm is less than 10^{-4}), and finally 2,000 regular and 2,000 chaotic samples are selected randomly to build such training dataset. For validation set, a raw dataset with time series of length 1000 with initial condition $x_0 = 0.75$ and the parameter α taking 12,000 equidistant values in $[0, 4)$ is created. It is screened and 1,000 time series of each dynamical behavior are selected randomly to obtain the final validation dataset. The test set is created similarly to validation dataset but taking $x_0 = 0.8$ as the initial condition.

As shown in [23], if we train 10 randomly initialized ANNs with the aforementioned architecture and the built train and validation sets (trained networks are checked with the test set), a powerful tool to detect chaos in the Logistic map is obtained (accuracy greater than 95% in all datasets and in all the trained networks).

Remark 1 *Equivalent experiments have been carried out for different values of the time series length and the total number of epochs during training, but values* 1,000 *and* 2,000, *respectively, seem to give the best results. Notice that in [23], length* 1,000 *was also proposed as the best option to train networks for a chaos detection task.*

2.2 Step 2: DL Chaos Analysis of an APD Heart Map

Our final goal (*Step 4*) is to perform chaos analysis of heart time series. We remark that the 10 ANNs have been trained with data from the Logistic map (*Step 1*), not with heart-like data. Moreover, the time series that will be analyzed in *Step 4* correspond to experimental recordings, that in general are short and noisy. Therefore, in this step we use some heart dynamics information, a simple cardiac map model, to perform several numerical tests on all trained ANNs. This is an important point as it connects generic and universal dynamics of the classical Logistic map with a more and specific cardiac map model. That is, the training is in some way universal, but the selection of the particular ANN is done by a more specific model focused on the nature of the experimental data.

The APD restitution curve (which describes the dynamics of a single cardiac cell) fitted to the kinetics of Beeler-Reuter model [44] gives rise to this discrete equation [1] (named as Beeler-Reuter APD heart map in what follows):

$$\begin{aligned}\mathrm{APD}_{i+1} = 258 + 125\exp\left(-0.068(\mathrm{DI}_i - 43.54)\right)\\ -350\exp\left(-0.028(\mathrm{DI}_i - 43.54)\right),\end{aligned}$$

where APD_{i+1} is the Action Potential Duration (APD) of the $(i+1)$-th stimulus and $\mathrm{DI}_i = n\,\mathrm{BCL} - \mathrm{APD}_i$ is the Diastolic Interval (DI) of previous stimulus. Therefore, the map equation can be rewritten as

$$\begin{aligned}\mathrm{APD}_{i+1} = 258 + 125\exp\left(-0.068(n\,\mathrm{BCL} - \mathrm{APD}_i - 43.54)\right)\\ -350\exp\left(-0.028(n\,\mathrm{BCL} - \mathrm{APD}_i - 43.54)\right),\end{aligned} \tag{3}$$

where BCL is the Basic Cycle Length (BCL), that is, the time between two consecutive pacing stimuli, and n is the parameter block (lower $n \in \mathbb{N}$ such that $n\,\mathrm{BCL} - \mathrm{APD}_i \geq \mathrm{DI}_{\min}$, with $\mathrm{DI}_{\min}$ the minimum DI whose value is set to 43.54ms). Notice that the BCL can be considered as the bifurcation parameter as it is independent of discrete time.

As the Beeler-Reuter APD heart map is a one-dimensional map, Lyapunov exponents [14] can be obtained applying formula

$$\begin{aligned}\mathrm{LE} = \frac{1}{T}\sum_{j=1}^{T} \log\big|8.5\exp\left(-0.068(n\,\mathrm{BCL} - \mathrm{APD}_j - 43.54)\right)\\ -9.8\exp\left(-0.028(n\,\mathrm{BCL} - \mathrm{APD}_j - 43.54)\right)\big|,\end{aligned} \tag{4}$$

with log the natural logarithm, and T large enough. An appropriate transient integration has to be computed before the application of the formula.

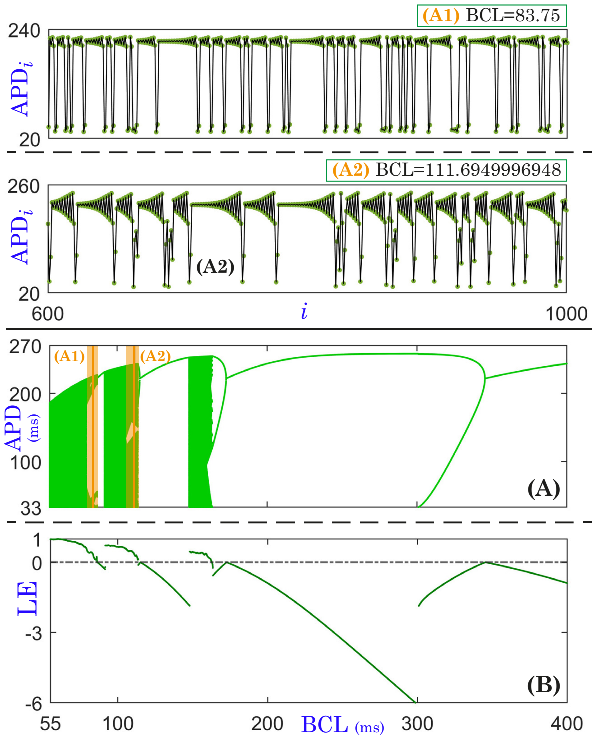


Figure 4: BCL-parametric analysis of the Beeler-Reuter APD heart map ($\mathrm{APD}_0 = 240$ ms). (A) Bifurcation diagram. Orange shading corresponds to regions where the network fails the most. (A1) and (A2) are two of these failing time series (in green we have the points of the time series, and we have joined such points with black segments for a better visualization). (B) Lyapunov exponents.

In panel (A) of Figure 4, we have depicted the bifurcation diagram of the Beeler-Reuter APD heart map for BCL $\in [55, 400)$ ms and $\text{APD}_0 = 240$ ms. In panel (B), LEs have been represented. If we move from higher to lower BCL values in such bifurcation diagram we can see equilibrium points, a period doubling bifurcation, and a 2:1 block (the cell responds one of each two pacing stimulus) followed by a period doubling, chaos, and higher-order blocks (see [1] for more dynamical information). In the LE panel it can be seen that for regular behavior (equilibrium points and periodic orbits) the LE is negative or zero, and it is positive for chaos.

We are going to perform a chaos analysis of one BCL-parametric line (see Figure 4) of this APD heart map using the networks trained in *Step* 1. Such BCL-parametric line has been obtained taking $12,000$ equidistant values for the BCL in the interval $[55, 400)$ ms and initial condition $\text{APD}_0 = 240$ ms. Equation 3 is applied for $20,500$ time points. First 500 points are discarded as transient, and the remaining ones are used for LEs computation (equation 4). The time series obtained for each BCL value are constructed with the last $1,000$ time points. Later, a component-wise Gaussian random noise with strength $\mu \in \{0, 0.5, 1.0\}$ is added to each time series (notice that when $\mu = 0$, no noise is added). The analyses of time series with Gaussian noise are repeated 100 times to give results as mean±standard deviation. Notice that the time series from the Logistic map only take values in $[0, 1]$. This does not occur with this Beeler-Reuter APD heart map, so we have to normalize the data to the interval $[0, 1]$ before performing the DL chaos analysis (a random number sampled uniformly in such interval is assigned to constant time series and a linearly mapping between the range of non-constant samples to the interval $[0, 1]$ is done otherwise).

Remark 2 *The initial condition and the times used for LEs computation in the APD heart map were taken as in [53].*

Now, we present different accuracy indicators adapted to the use of experimental data and the cardiac model used to provide information to select in *Step 3* the most appropriate network (among all computed in *Step 1*) to analyze heart data under noise and the absence of it.

The accuracy is the measure proposed to check the performance of the networks. It is defined as

$$\text{Accuracy (\%)} = \frac{T_R + T_C}{T_R + T_C + F_R + F_C} \cdot 100,$$

where T_R and T_C are the number of true regular and true chaotic (that is, the number of samples that were correctly classified by the network as regular and chaotic, respectively); and F_R and F_C are the false regular and false chaotic (that is, the number of samples that were incorrectly classified by DL as regular and chaotic, respectively). Notice that if the dataset in which this formula is applied is unbalanced (we do not have the same number of samples of each dynamical behavior) as occurs in our BCL-parametric line (19.708% of the samples are chaotic according to LE value, and the remaining 80.292% have a regular behavior), this measure cannot be reliable. For example, in a dataset with a large number of regular and a small number of chaotic samples, even if all the chaotic samples are not detected correctly, the accuracy can be close to 100% if all the regular samples were correctly classified. Of course this does not mean that the network is able to perform the chaos detection analysis properly. To avoid this problem, we will use other variants of the accuracy, the accuracy chaotic and accuracy regular that compute the percentage of chaotic and regular samples, respectively, that are correctly classified. The corresponding formulas are

$$\text{Accuracy Chaotic (\%)} = \frac{T_C}{T_C + F_R} \cdot 100,$$

$$\text{Accuracy Regular (\%)} = \frac{T_R}{T_R + F_C} \cdot 100.$$

Remark 3 *If we consider chaotic as the positive class and regular as the negative one, accuracy chaotic corresponds to the sensitivity or recall (in %), and accuracy regular is the specificity (in %).*

Notice that a regular dynamical behavior includes mainly two different types of dynamics: equilibrium points (EPs) and periodic orbits (POs). To complete our analysis we will also compute the percentage of equilibrium points and periodic orbits that have been classified correctly as regular (accuracy EPs and accuracy POs):

$$\text{Accuracy EPs (\%)} = \frac{T_{\text{R}\,-\,\text{EPs}}}{T_{\text{R}\,-\,\text{EPs}} + F_{\text{C}\,-\,\text{EPs}}} \cdot 100,$$

$$\text{Accuracy POs (\%)} = \frac{T_{\text{R}\,-\,\text{POs}}}{T_{\text{R}\,-\,\text{POs}} + F_{\text{C}\,-\,\text{POs}}} \cdot 100,$$

with $T_{\text{R}\,-\,\text{EPs}}$ and $T_{R\,|\,\text{POs}}$ the number of equilibrium points and periodic orbits, respectively, from the regular samples that have been classified correctly; and $F_{\text{C}\,-\,\text{EPs}}$ and $F_{\text{C}\,-\,\text{POs}}$ the number of equilibrium points and periodic orbits, respectively, that are classified as chaotic. In our parametric line, 79.803% of the regular samples are equilibrium points and the 20.197% present periodic behavior, so we consider that the measures that we have just defined are important to check that with DL both dynamics are detected properly as regular.

Finally, we define an indicator to detect if the network detection is robust against noise. We call it as $\textit{diff}_{0-\tilde{\mu}}$ and it corresponds to the percentage of samples that the network does not detect with the same behavior in the analysis without noise ($\mu = 0$) and with strength noise $\tilde{\mu} \neq 0$ (in our analysis $\tilde{\mu} = 0.5$ or $\tilde{\mu} = 1.0$). That is, if $S_\mu = \{\text{classification obtained from data with strength noise } \mu\}$, then

$$\textit{diff}_{0-\tilde{\mu}} = 100 - \frac{\#[S_{\mu=0} \cap S_{\mu=\tilde{\mu}}]}{\text{number of total samples}} \cdot 100,$$

where # is used to refer to cardinality.

Remark 4 *The tests performed in this paper with Beeler-Reuter APD heart map can also be performed with Lewis and Guevara APD heart map [53] and similar/equivalent results are obtained.*

2.3 Step 3: Selection of the Most Suitable Artificial Neural Network

Once we have performed the DL chaos analysis of the APD heart map (*Step 2*) we have to establish some criteria to choose automatically (without direct human supervision) a robust network against noise that can perform properly chaos detection in the Beeler-Reuter APD heart map. We expect that the selected network will be able to perform properly the DL chaos analysis of biological time series of frog heart dynamics.

For the automatic selection criteria we will take into account accuracy-like measures and the indicator $\mathit{diff}_{0-\tilde{\mu}}$ (for $\tilde{\mu} \in \{0.5, 1.0\}$) defined in *Step 2*. In particular, our selection criteria is as follows: all accuracies (accuracy, accuracy chaotic, accuracy regular, accuracy EPs and accuracy POs) have to be greater than 75% in mean (to ensure good performance in chaos detection task), and $\mathit{diff}_{0-0.5}$ and $\mathit{diff}_{0-1.0}$ have to be less than 1% in mean (to ensure robustness in the detection against noise). Moreover, we impose that the standard deviation for all the measures and indicators has to be less than 1% (to ensure that results are quite independent of randomness).

Notice that we are checking if any of the 10 trained networks (*Step 1*) has gone further, and besides having learned to detect chaos in the Logistic map, is able to generalize its detection to discrete heart dynamics. Note that the ANNs have only been trained with dynamics from the Logistic map, the data was not preprocessed in any special way, and no extra dynamical information has been given to the networks, so the proposed task is not easy and we consider that the aforementioned criteria is reasonable.

In Table 1 we have the results for the DL chaos analysis (*Step 2*) of the network that satisfies all the imposed criteria. In particular, for the cases with noise (second and third columns), the analyses have been performed 100 times because of randomness, and results correspond to mean±standard deviation.

Let us focus on the results of the without noise case ($\mu = 0$). Notice that the value of all the accuracies (except accuracy chaotic) are greater than 95%. Notice that both types of regular behavior (equilibrium points and periodic orbits) are properly detected in almost all the cases. It is necessary to check why the accuracy chaotic is lower than the other accuracy measures (but still greater than 75%). The major part of the samples that were detected incorrectly as regular when their behavior is chaotic are in the boundary parts shaded in orange in the bifurcation diagram of the APD heart map in panel (A) of Figure 4. In panels (A1) and (A2) of this figure we have an example of a failing sample on each orange region (BCL = 83.75 ms and BCL = 111.6949996948 ms, respectively). Notice that both samples are quite similar. As we can see, the general behavior is chaotic. It highlights the upper parts of the time series in which, comparing with the general range of APD values of the time series, there is not much variability. This is a behavior that we do not expect to be present in the Logistic map where the networks were trained. Taking into account that the data used to train the networks does not have extra dynamical information beyond what the time series can provide, and this behavior is not present, to require the network to detect it is very demanding.

For the analyses with noise (strength noise $\mu \in \{0.5, 1.0\}$), all the accuracy results are quite good and similar to those given by the without noise case (with the standard deviation less than 0.15 in all cases). This is confirmed by the $\mathit{diff}_{0-0.5}$ and $\mathit{diff}_{0-1.0}$ indicators that are lower than 0.5% in mean and with standard deviation lower than 0.06, increasing its value with strength noise as expected.

2.4 Step 4: DL Chaos Analysis of Heart Time Series

So far, we have trained 10 randomly initialized recurrence-like Artificial Neural Networks using data from the Logistic map (*Step 1*), we have performed DL chaos analyses of an APD heart map with all the trained networks (*Step 2*), and we have defined some criteria to use these analyses to choose one network that performs well and is robust against noise (*Step 3*). Now, we have all the ingredients to try a DL chaos analysis of biological time series of heart dynamics.

When training the networks, the data comes from the Logistic map. As already highlighted before, this equation is constructed in such a way that time series are in the interval $[0, 1]$. In general, experimental data is not in the interval $[0, 1]$, so we have to normalize it to that interval (the trained networks will not be able to process data properly in whatever other rank). With the data already normalized, time series are given as input to the chosen network in *Step 3*. The output of the network will give us the information about the behavior (regular or chaotic) of such time series. Therefore, the defined algorithm allows to perform chaos analysis of heart time series without human supervision.

Remark 5 *Notice that the trained network is a recurrence-like neural network, therefore, the length of the input is not fixed and time series of whatever length can be processed. For example, with other architectures as the Multi-Layer Perceptron (obtained when perceptrons are stacked) the input size has to be constant.*

2.5 Pseudocode for Analyzing Chaotic Dynamics in Heart Time Series

In Algorithm 1 we have the pseudocode of the DL algorithm for chaos analysis of heart time series that we have described in Subsections 2.1-2.4. For simplicity

	$\mu = 0$	$\mu = 0.5$	$\mu = 1.0$
Accuracy (%)	95.092	94.752 ± 0.050	94.651 ± 0.060
Accuracy Chaotic (%)	76.195	75.951 ± 0.071	75.552 ± 0.112
Accuracy Regular (%)	99.730	99.367 ± 0.060	99.339 ± 0.069
Accuracy EPs (%)	100	99.545 ± 0.075	99.542 ± 0.087
Accuracy POs (%)	98.666	98.664 ± 0.009	98.541 ± 0.037
$\boldsymbol{diff}_{\mathbf{0}-\tilde{\mu}}$ **(%)**	-	0.361 ± 0.049	0.493 ± 0.059

Table 1: Results of the DL chaos analysis of the Beeler-Reuter APD heart map with the network chosen by the criteria established in *Step 3*. Notice that $\mu = 0$ corresponds to the analysis without noise. Results for $\mu \in \{0.5, 1.0\}$ are given as mean±standard deviation for 100 trials.

and better comprehension, we have used some abbreviations. In particular, with `num_nets` we refer to the number of ANNs with the same architecture that were randomly initialized (it is set to 10 as indicated in Subsection 2.1). $\texttt{ANN}_{\texttt{arch}}$ corresponds to the architecture of the network described in Subsection 2.1. $\texttt{dataLM}_*$ with $* \in \{\texttt{train}, \texttt{val}, \texttt{test}\}$ is the data from the Logistic map used to train, validate, and test the network, respectively (such data includes the time series $\texttt{dataLM}_*^{\texttt{time series}}$ and the corresponding labels $\texttt{dataLM}_*^{\texttt{labels}}$). The other two datasets used in the algorithm are the BCL-parametric line of the Beeler-Reuter APD heart map to which we refer as $\texttt{dataBR}_{\texttt{BCL-line}}$ (with $\texttt{dataBR}_{\texttt{BCL-line}}^{\texttt{time series}}$ the time series and $\texttt{dataBR}_{\texttt{BCL-line}}^{\texttt{labels}}$ the labels), and the experimental dataset of heart dynamics indicated as $\texttt{dataExperimental}^{\texttt{time series}}$ (note that in this case we will not have the labels $\texttt{dataExperimental}^{\texttt{labels}}$). In the case of notation $\texttt{acc-like}_{\mu_j}(i)$ for $j = 1, 2, 3$ and $i = 1, \cdots, \texttt{num\_nets}$ (line 15 of the code), we understand it as a list that includes the accuracy, the accuracy chaotic, the accuracy regular, the accuracy EPs and the accuracy POs of net i when noise with strength μ_j is added to the data. In addition, these values (as well as the values of the robustness indicator) are given as mean±standard deviation (mean±std). Therefore in the criteria of *Step 3* (see line 23), for example, condition $\texttt{acc-like}_{\mu_j}^{\texttt{mean}}(i) > 75\%$ indicates that all the elements of the list have to be greater than 75% in mean.

Remark 6 *Notice that in Step 2, for case $j = 1$, the value of $\mu_j = \mu_1$ is 0, so in the part of the algorithm where Gaussian noise is added (line 12 of the code) the data does not change. Moreover, for this value of j in this step of the algorithm, the computation of the robustness indicator $diff_{\mu_1-\mu_j} = diff_{\mu_1-\mu_1}$ (line 16 of the code) is trivial as it will always be equal to 0% (for this reason, it is not included as one of the criteria of Step 3, see line 23 of the code). An* `if` *statement could be added to avoid calculating it.*

3 Results of DL Algorithm for Chaos Analysis: Frog Heart Data

In this section we explain how the experimental data (frog heart signals) is obtained and we apply the previous detailed Algorithm 1 to analyze its chaotic dynamics.

3.1 Data. Frog Cardiomyocyte Recordings

In this subsection we explain in detail the protocol of the experiment and how it has been carried out to obtain the frog heart signals.

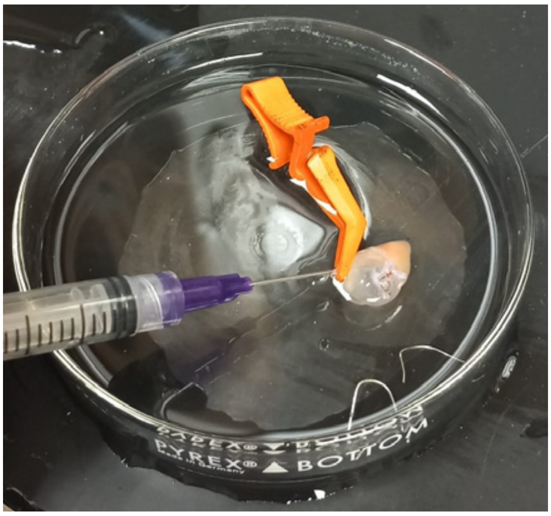


Figure 5: Photo taken during an experiment with a frog heart in the laboratory of Professor Flavio H. Fenton in Georgia Tech.

The frog heart experiments were performed following an approved Georgia Tech, Institutional Animal Care and Use Committee (IACUC) protocol #, 100673. Frogs were euthanized via fast decapitation following by Pithing of the brain and spinal cord which ensures the cessation of neural activity avoiding any pain. The heart is then quickly excised and cannulated via the aorta using a syringe filled with Tyrode solution, a blood substitute, to wash out all the blood (see Figure 5). This process ensured a blood-cloths free preparation and allowed the heart to be kept physiologically viable in a container at room temperature, also filled with Tyrode solution, throughout the experiment.

Transmembrane voltage signals were recorded as described in previous preparations [54, 55]. Briefly, individual heart cells from the whole heart were impaled with a fine-tipped micro-electrode, which was connected to a high-impedance amplifier to minimize signal loss. The amplified signals were routed to a data acquisition (DAQ)

Algorithm 1 DL Algorithm for Analyzing Chaotic Dynamics in Heart Time Series

```
procedure Step 1 ($\text{num\_nets} = 10, \text{ANN}_{\text{arch}}, \text{dataLM}_{\text{train}}, \text{dataLM}_{\text{val}}, \text{dataLM}_{\text{test}}$)   ▷ ANNs Framework
    parallel for $i = 1$ to num_nets
        $\text{Net}(i) \leftarrow$ RandomInitialization$(\text{ANN}_{\text{arch}})$   ▷ Random initialization of ANN trainable parameters
        $\text{T-Net}(i) \leftarrow$ Train$(\text{Net}(i), \text{dataLM}_{\text{train}}, \text{dataLM}_{\text{val}})$   ▷ Training
        Test$(\text{T-Net}(i), \text{dataLM}_{\text{test}})$   ▷ Test
        return $\text{T-Net}(i)$
    end parallel for
end procedure
procedure Step 2 ($\mu_1 = 0,\ \mu_2 = 0.5, \mu_3 = 1.0, \text{T-Net}(1), \cdots, \text{T-Net}(\text{num\_nets}), \text{dataBR}_{\text{BCL-line}}$)   ▷ Analysis APD Map
    parallel for $i = 1$ to num_nets
        for $j = 1$ to 3 do
            $\text{dataBR}_{\text{BCL-line},\,\mu_j} \leftarrow \text{dataBR}^{\text{time series}}_{\text{BCL-line}} + \mu_j \cdot \mathit{gaussian}(0, 1)$   ▷ Add noise with strength $\mu_j$
            $\text{N-dataBR}_{\text{BCL-line},\mu_j} \leftarrow$ normalization$(\text{dataBR}_{\text{BCL-line},\mu_j})$   ▷ Data normalization
            $\text{result}_{\mu_j}(i) \leftarrow$ DLAnalysis$(\text{T-Net}(i), \text{N-dataBR}_{\text{BCL-line},\mu_j})$   ▷ DL chaos analysis
            $\text{acc-like}_{\mu_j}(i) \leftarrow$ AccuracylikeMeasures$(\text{result}_{\mu_j}(i), \text{N-dataBR}^{\text{labels}}_{\text{BCL-line}})$   ▷ Accuracy-like measures
            $\mathit{diff}_{\mu_1-\mu_j}(i) \leftarrow$ RobustnessIndicator$(\text{result}_{\mu_1}(i), \text{result}_{\mu_j}(i))$   ▷ Robustness indicator
            return $\text{acc-like}_{\mu_j}(i)$, $\mathit{diff}_{\mu_1-\mu_j}(i)$
        end for
    end parallel for
end procedure
procedure Step 3 ($\text{acc-like}_{\mu_j}(i)$, $\mathit{diff}_{[\mu_1=0]-\mu_j}(i)$ with $i = 1, \cdots, \text{num\_nets}$, $j = 1, 2, 3$ )   ▷ Selection of suitable ANN
    parallel for $i = 1$ to num_nets
        if $\text{acc-like}_{\mu_1}(i) > 75\%$ and $\text{acc-like}^{\text{mean}}_{\mu_2}(i) > 75\%$ and $\text{acc-like}^{\text{std}}_{\mu_2}(i) < 1\%$ and $\text{acc-like}^{\text{mean}}_{\mu_3}(i) > 75\%$ and $\text{acc-like}^{\text{std}}_{\mu_3}(i) < 1\%$ and $\mathit{diff}^{\text{mean}}_{\mu_1-\mu_2}(i) < 1\%$ and $\mathit{diff}^{\text{std}}_{\mu_1-\mu_2}(i) < 1\%$ and $\mathit{diff}^{\text{mean}}_{\mu_1-\mu_3}(i) < 1\%$ and $\mathit{diff}^{\text{std}}_{\mu_1-\mu_3}(i) < 1\%$ then
            $\text{suitableIndex} \leftarrow i$   ▷ Index of the ANN that satisfies the criteria
            return suitableIndex
        end if
    end parallel for
end procedure
procedure Step 4 ($\text{T-Net}(\text{suitableIndex}), \text{dataExperimental}^{\text{time series}}$)   ▷ Analysis Experimental Data
    parallel for $k = 1$ to size$(\text{dataExperimental}^{\text{time series}})$
        $\text{N-dataExperimental} \leftarrow$ normalization$(\text{dataExperimental}^{\text{time series}})$   ▷ Data normalization
        $\text{result} \leftarrow$ DLAnalysis$(\text{T-Net}(\text{suitableIndex}), \text{N-dataExperimental})$   ▷ DL chaos analysis experimental data
        return result
    end parallel for
end procedure
```

board interfaced with a computer for recording and analysis. This setup enabled precise voltage recordings of the electrical activity of the cardiac cells.

To induce variations in the Basic Cycle Length (BCL) of pacing, an isolated external current stimulation device was employed. The stimulating electrode was positioned about 0.5 *cm* from the recording micro-electrode and delivered currents at twice the excitation threshold. This ensured effective and consistent stimulation of the cardiac tissue for the experimental protocols.

Our experimental dataset contains 52 voltage time series with several APDs with different BCL. In particular, the minimum length is 15 and the maximum is 205, but all were recorded after the BCL was left for several minutes of accomodation to that BCL. Such time series correspond to the Action Potential Duration of frog heart dynamics for different pacing rates in the interval $[50, 1000]$. In panel (A) of Figure 7 (ignore colors for now) we have the bifurcation diagram obtained with our experimental dataset (x-axis is the pacing rate BCL and y-axis is the time series). In panel (B) of this figure (ignore colors for now) we have the length (number of points) of each time series.

Remark 7 *Note that the length of the experimental time series are relatively short to use classical techniques for Lyapunov exponents (which theoretically require large data sets) and thus can not be applied successfully for chaos analysis in these datasets.*

3.2 Results

We are going to apply the DL algorithm for chaos analysis of heart time series defined in Section 2 to some experimental data. That is, once *Steps 1-3* (see Subsections 2.1-2.3) have been carried out, and a network that we expect to be suitable for chaos analysis has been obtained, we use it in the experimental data of frog heart dynamics defined in Subsection 3.1.

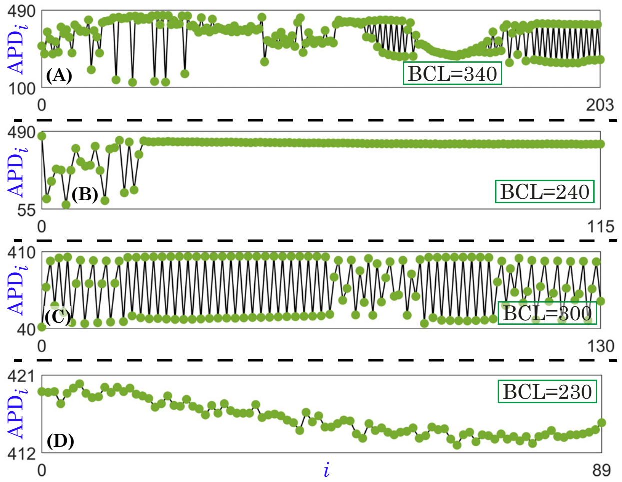


Figure 6: Time series to illustrate the rules applied when labeling experimental data manually. (A) corresponds to *Rule a*, (B) to *Rule b*, (C) to *Rule c* and (D) to *Rule d*. In green we have the points that correspond to the time series, and we have joined such points with black segments for a better visualization of the behavior.

To check if the chaos analysis has been successful, we need to label the experimental dataset manually according to expert criteria. In general, we consider the asymptotic behavior to label the data. In particular, the rules taken into account for such classification are as follows:

Rule a. If a time series seems to have long chaotic transient behavior (more than a quarter of the length) and only some regularity can be inferred at the end (see Figure 6(A) for an example), it is considered that the behavior is chaotic. In this case, we cannot ensure that the asymptotic behavior is regular as not enough information is provided.

Rule b. If a time series seems to have short chaotic transient behavior (less than a quarter of the length) and some regularity later (as in Figure 6(B)), it is classified by the expert as regular.

Rule c. If some regular windows are present in a general chaotic behavior (see Figure 6(C)), chaotic behavior is assigned.

Rule d. As we work with experimental data, the dynamics corresponding to equilibrium points are not constant values over time. We consider that a time series is an equilibrium point if when transient has ended (that is, in the last three quarters according to Rule b.), the point values have a difference less than 25 in the original range before normalization (see Figure 6(D) for an example).

Taking into account these rules, 13 of the 52 samples have chaotic behavior, and the remaining 39 are regular.

In Figure 7(A), the DL chaos analysis of experimental data of heart dynamics is depicted. In blue we have the samples that have been correctly detected as regular by the algorithm, in red those correctly classified as chaotic, and in black the incorrectly detected ones (that is, false regular and false chaotic detections). As we can see, just 5 of the 52 samples have been incorrectly classified (9.615% of the samples). That is, the DL algorithm for chaos analysis has been successful in 90.385% of the dataset (47 of 52 samples). As our dataset is not balanced (we have more regular than chaotic samples), it is important to compute the accuracy chaotic and accuracy regular. In particular, such values are 92.308% (12 of 13 samples) and 89.744% (35 of 39 time series), respectively. Notice that all the accuracies are around 90% of success, and it seems that the DL algorithm for chaos analysis of heart time series has worked properly. In Table 2 we have summarized such accuracy results.

	Experimental Results
Accuracy (%)	90.385 % (47 of 52 samples)
Accuracy Chaotic (%)	92.308 % (12 of 13 samples)
Accuracy Regular (%)	89.744 % (35 of 39 samples)

Table 2: Results of the DL chaos analysis of the experimental data (frog heart dynamics) with the network obtained in the algorithm (*Steps 1-3*).

In Figure 7(B) we have represented the length, number of data points, of each experimental time series (x-axis correspond to the BCL value of the sample, and the height of the line is the length). Colors correspond to the network classification: blue for correct regular detection, red for correct chaotic detection, and black for incorrect detection. The green dashed horizontal lines indicate length values 50, 100, 150 and 200, and have been added to facilitate visualization. As already mentioned when describing the experimental dataset, the time series have different lengths (minimum length 15 and maximum length 205). In particular, 40.385% of the samples (21 of 52) contain at most 50 time points, and 80.769% (42 of 52 samples) have at most 100. Notice that most of the samples are quite short, which can complicate the task as not enough information can be provided to the network. However, it seems that the applied DL algorithm for chaos analysis is able to detect properly most of the regular and chaotic behavior regardless of the length.

Let us analyze in detail the incorrectly detected samples (in black in Figure 7). We have represented them in

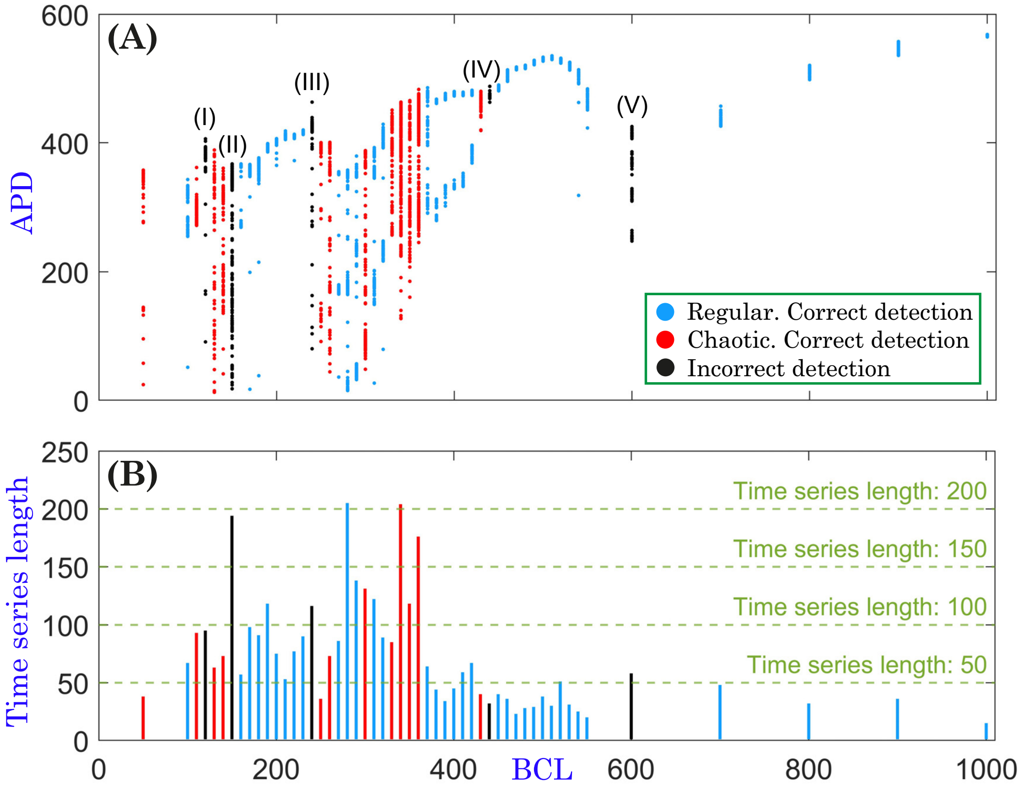


Figure 7: Chaos analysis of experimental data of frog heart dynamics using the DL algorithm of Section 2. (A) Results of the chaos analysis of experimental time series. (B) Time series length (that is, number of time points). In both panels, colors correspond to the DL classification: blue is used for correct regular detection, red for correct chaotic detection, and black for incorrect detection.

Figure 8. In green we have the points that correspond to the time series, we have connected such points with black segments for a better visualization of the behavior. Notice that not all of the samples have the same length, that is, the scale of x-axis is different for each time series. As can be seen in Figure 8, samples (I), (III) and (IV) have a similar behavior: short chaotic transient dynamics, asymptotically converging to an equilibrium point (regular). According to *Rule b* that we have used for expert classification of the experimental time series, the samples have been labeled as regular. The ANN detected them as chaotic. The whole time series are chaotic (as DL detected), but dynamically their asymptotic behavior can be considered as regular (as we labeled them). Sample (II), as can be seen in Figure 8, seems to have some periodicity at the end, with long chaotic transient. Therefore, applying *Rule a* of the expert classification, the time series has been labeled as chaotic. However, the algorithm detected it as regular. The fifth incorrect detection corresponds to sample (V). As seen in Figure 8, it can be difficult to classify such an orbit. In fact, this data seems to correspond to an orbit inside or just after a period-doubling cascade, so it can be difficult to classify it as a periodic orbit with a high period, a quasi-periodic orbit or a Feigenbaum chaotic orbit. As it seems to follow some periodicity along all the time points, we have been labeled it as regular. The network detected it as chaotic with a score of 0.625 for this class (that is, with a high uncertainty). Note that all these "incorrectly classified" time series have regular and chaotic patterns, and so any classification cannot be accurate as there are too few data points to correctly classify them. Therefore, the results of the ANN cannot really be considered as a false result.

Remark 8 *Notice that in the case of sample (V) (see Figures 7 and 8), the network assigned a score* 0.625 *for chaotic class and a score* 0.375 *for regular class. The values are closer to* 0.5 *than to* 1*, what means that the ANN is not sure about its decision. If some human supervision wants to be added to the algorithm defined in Section 2, we can add a control to Step 4. It is defined as follows: if the higher score of one sample for both categories is lower than* 2/3*, such data point should be revised by an expert as the detection score is close to the classification border (*0.5 *is the threshold to distinguish between regular and chaotic). Adding this control in our experimental dataset, just* 3.486% *of the samples (*2 *of* 52 *samples) should be revised. These two samples have been represented in Figure 9. As already indicated, one of these samples (sample (B)) corresponds to one of the samples incorrectly detected (see sample (V) of Figure 8). The other one (sample (A)) was correctly classified by*

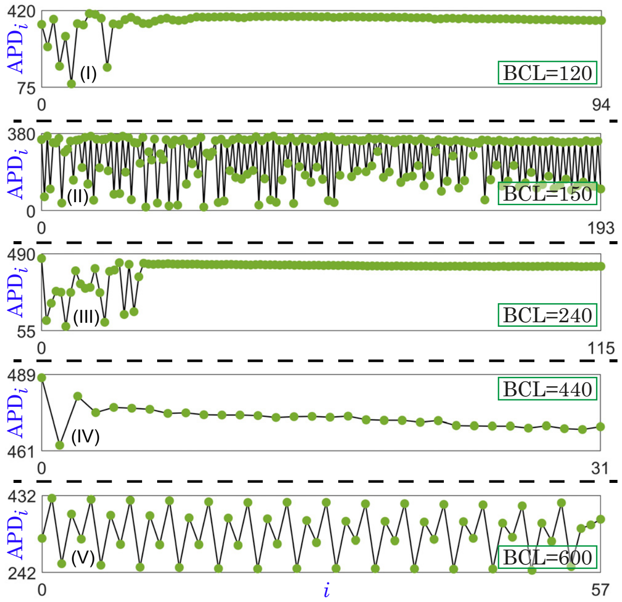


Figure 8: Time series that have been incorrectly detected. They correspond to samples (I)-(V) in black in Figure 7. In green we have the points that correspond to the time series and we have joined such points with black segments for a better visualization of the behavior.

the algorithm. The network assigned a score 0.592 *for the chaotic class and* 0.408 *for regular class. Notice that the time series seems to follow some periodicity at the beginning and the end of the recordings but in between it presents a chaotic behavior, so the doubt of the network is totally justified. After this expert supervision, the final accuracy for the whole dataset would be* 92.308% *(*48 *of the* 52 *samples), with accuracy regular increasing also to this value (*36 *of* 39 *samples).*

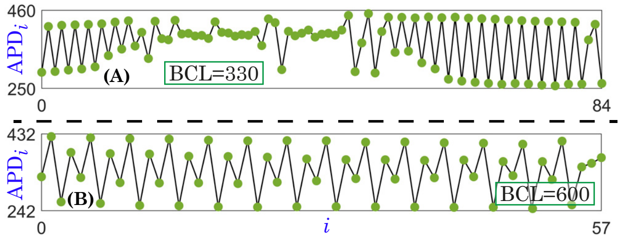


Figure 9: Time series with detection score less than 2/3 for the winning class. In green we have the dots that correspond to the time series and we have connected them with black segments for a better visualization of the behavior.

4 Conclusions

There is an increasing amount of experimental data in practice, so automatic techniques to classify them are an important task. This data often has some drawbacks, such as noisy and short recordings. In this paper, we propose a Deep Learning-based algorithm to tackle the chaos analysis of biological time series. We use an Artificial Neural Network architecture based on Long Short-Term Memory cells with a final classification layer. The algorithm is divided into different steps. The training data is taken from a basic and generic discrete dynamical system: the Logistic Map. Subsequently, once 10 ANNs have been trained, we use an APD heart map (Beeler-Reuter discrete map model) to perform a test. This analysis allows to select an Artificial Neural Network that can adequately perform a chaos analysis of biological time series of frog heart dynamics.

This point is very relevant, as it means, on the one hand, that trained generic ANNs can be used for particular problems once the most suitable one is selected using the data of the particular problem we want to address. On the other hand, this is a de facto proof that most dynamical phenomena are quite general and universal.

The technique is especially useful in the case of very short data where other techniques, such as the calculation of the maximum Lyapunov exponent, do not work properly. The algorithm is tested on a set of experimental data obtained from real frog heart dynamics in the laboratory of Professor Flavio H. Fenton, and yields very accurate results.

Acknowledgments

RB and CMC have been supported by the European Regional Development Fund and the Spanish Ministerio de Ciencia e Innovación (PID2021-122961NB-I00) and the European Regional Development Fund and Diputación General de Aragón (E24-23R). RB has been supported by the European Regional Development Fund and Diputación General de Aragón (LMP94-21). CMC has been supported by Ministerio de Universidades de España with an FPU grant (FPU20/04039).

Data Availability

Data available on request from the authors.

References

[1] H. M. Hastings, F. H. Fenton, S. J. Evans, O. Hotomaroglu, J. Geetha, K. Gittelson, J. Nilson, and A. Garfinkel, “Alternans and the onset of ventricular fibrillation,” *Physical Review E*, vol. 62, no. 3, p. 4043, 2000.

[2] Y. Xie, G. Hu, D. Sato, J. N. Weiss, A. Garfinkel, and Z. Qu, “Dispersion of refractoriness and induction of reentry due to chaos synchronization in a model of cardiac tissue,” *Physical Review Letters*, vol. 99, no. 11, p. 118101, 2007.

[3] A. Garfinkel, M. Spano, W. Ditto, and J. Weiss, "Controlling cardiac chaos," *Science (New York, N.Y.)*, vol. 257, pp. 1230–5, 09 1992.

[4] J. E. Skinner, A. L. Goldberger, G. Mayer-Kress, and R. E. Ideker, "Chaos in the heart: Implications for clinical cardiology," *Bio/technology*, vol. 8, no. 11, pp. 1018–1024, 1990.

[5] N. Thakor, "Chaos in the heart: Signals and models," in *Proceedings of the 1998 2nd International Conference Biomedical Engineering Days*, pp. 11–18, 1998.

[6] A. L. Goldberger, "Is the normal heartbeat chaotic or homeostatic?," *Physiology*, vol. 6, no. 2, pp. 87–91, 1991.

[7] C.-S. Poon and C. K. Merrill, "Decrease of cardiac chaos in congestive heart failure," *Nature*, vol. 389, no. 6650, pp. 492–495, 1997.

[8] A. L. Goldberger, V. Bhargava, B. J. West, and A. J. Mandell, "Some observations on the question: Is ventricular fibrillation "chaos"?," *Physica D: Nonlinear Phenomena*, vol. 19, no. 2, pp. 282–289, 1986.

[9] J. N. Weiss, A. Garfinkel, H. S. Karagueuzian, Z. Qu, and P.-S. Chen, "Chaos and the transition to ventricular fibrillation: a new approach to antiarrhythmic drug evaluation," *Circulation*, vol. 99, no. 21, pp. 2819–2826, 1999.

[10] D. R. Chialvo and J. Jalife, "Non-linear dynamics of cardiac excitation and impulse propagation," *Nature*, vol. 330, no. 6150, pp. 749–752, 1987.

[11] D. R. Chialvo, R. F. Gilmour Jr, and J. Jalife, "Low dimensional chaos in cardiac tissue," *Nature*, vol. 343, no. 6259, pp. 653–657, 1990.

[12] A. Gizzi, E. M. Cherry, R. F. Gilmour Jr, S. Luther, S. Filippi, and F. H. Fenton, "Effects of pacing site and stimulation history on alternans dynamics and the development of complex spatiotemporal patterns in cardiac tissue," *Frontiers in Physiology*, vol. 4, p. 71, 2013.

[13] S. Iravanian, I. Uzelac, A. D. Shah, M. J. Toye, M. S. Lloyd, M. A. Burke, M. A. Daneshmand, T. S. Attia, J. D. Vega, M. F. El-Chami, *et al.*, "Complex repolarization dynamics in ex vivo human ventricles are independent of the restitution properties," *Europace*, vol. 25, no. 12, p. euad350, 2023.

[14] J. H. Argyris, G. Faust, and M. Haase, *An Exploration of Chaos: An Introduction for Natural Scientists and Engineers.* North Holland, 1994.

[15] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, 1985.

[16] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, "A practical method for calculating largest Lyapunov exponents from small data sets," *Physica D: Nonlinear Phenomena*, vol. 65, no. 1-2, pp. 117–134, 1993.

[17] G. A. Gottwald and I. Melbourne, "The 0-1 test for chaos: A review," *Chaos Detection and Predictability*, pp. 221–247, 2016.

[18] R. Barrio and S. Serrano, "A three-parametric study of the Lorenz model," *Physica D: Nonlinear Phenomena*, vol. 229, no. 1, pp. 43–51, 2007.

[19] R. Barrio, M. Martínez, E. Pueyo, and S. Serrano, "Dynamical analysis of Early Afterdepolarization patterns in a biophysically detailed cardiac model," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 7, 2021.

[20] R. Halfar, "Dynamical properties of Beeler-Reuter cardiac cell model with respect to stimulation parameters," *International Journal of Computer Mathematics*, vol. 97, no. 1-2, pp. 498–507, 2020.

[21] M. Paluš, "Chaotic measures and real-world systems: Does the Lyapunov exponent always measure chaos?," in *Nonlinear Analysis of Physiological Data* (H. Kantz, J. Kurths, and G. Mayer-Kress, eds.), (Berlin, Heidelberg), pp. 49–66, Springer Berlin Heidelberg, 1998.

[22] A. Celletti, C. Gales, V. Rodriguez-Fernandez, and M. Vasile, "Classification of regular and chaotic motions in Hamiltonian systems with Deep Learning," *Scientific Reports*, vol. 12, no. 1, pp. 1–12, 2022.

[23] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 7, p. 073146, 2023.

[24] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using Machine Learning to replicate chaotic attractors and calculate Lyapunov exponents from data," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 12, 2017.

[25] C. Mayora-Cebollero, A. Mayora-Cebollero, Álvaro Lozano, and R. Ba-rrio, "Full Lyapunov exponents spectrum with Deep Learning from single-variable time series," *Physica D: Nonlinear Phenomena*, vol. 472, p. 134510, 2025.

[26] J. Nolasco and R. W. Dahlen, "A graphic method for the study of alternation in cardiac action potentials," *Journal of Applied Physiology*, vol. 25, no. 2, pp. 191–196, 1968.

[27] M. Guevara, G. Ward, A. Shrier, and L. Glass, "Electrical alternans and period doubling bifurcations," *Computing in Cardiology*, vol. 562, pp. 167–170, 1984.

[28] M. R. Franz, C. D. Swerdlow, L. B. Liem, J. Schaefer, *et al.*, "Cycle length dependence of human action potential duration in vivo. effects of single extrastimuli, sudden sustained rate acceleration and deceleration, and different steady-state frequencies.," *The Journal of Clinical Investigation*, vol. 82, no. 3, pp. 972–979, 1988.

[29] SZIGLIGETI, BÁNYÁSZ, MAGYAR, SZIGETI, PAPP, VARRÓ, and NÁNÁSI, "Intracellular calcium and electrical restitution in mammalian cardiac cells," *Acta Physiologica Scandinavica*, vol. 163, no. 2, pp. 139–147, 1998.

[30] M. Watanabe, N. F. Otani, and R. F. Gilmour Jr, "Biphasic restitution of action potential duration and complex dynamics in ventricular myocardium," *Circulation Research*, vol. 76, no. 5, pp. 915–921, 1995.

[31] Z. Qu, J. N. Weiss, and A. Garfinkel, "Spatiotemporal chaos in a simulated ring of cardiac cells," *Physical Review Letters*, vol. 78, no. 7, p. 1387, 1997.

[32] F. H. Fenton, E. M. Cherry, H. M. Hastings, and S. J. Evans, "Multiple mechanisms of spiral wave breakup in a model of cardiac electrical activity," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 12, no. 3, pp. 852–892, 2002.

[33] F. H. Fenton, S. J. Evans, and H. M. Hastings, "Memory in an excitable medium: A mechanism for spiral wave breakup in the low-excitability limit," *Physical Review Letters*, vol. 83, no. 19, p. 3964, 1999.

[34] E. G. Tolkacheva, D. G. Schaeffer, D. J. Gauthier, and W. Krassowska, "Condition for alternans and stability of the 1:1 response pattern in a "memory" model of paced cardiac dynamics," *Physical Review E*, vol. 67, no. 3, p. 031904, 2003.

[35] J. J. Fox, E. Bodenschatz, and R. F. Gilmour Jr, "Period-doubling instability and memory in cardiac tissue," *Physical Review Letters*, vol. 89, no. 13, p. 138101, 2002.

[36] E. M. Cherry and F. H. Fenton, "Suppression of alternans and conduction blocks despite steep APD restitution: Electrotonic, memory, and conduction velocity restitution effects," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 286, no. 6, pp. H2332–H2341, 2004.

[37] M. E. Díaz, S. C. O'Neill, and D. A. Eisner, "Sarcoplasmic reticulum calcium content fluctuation is the key to cardiac alternans," *Circulation Research*, vol. 94, no. 5, pp. 650–656, 2004.

[38] J. G. Restrepo and A. Karma, "Spatiotemporal intracellular calcium dynamics during cardiac alternans," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 19, no. 3, 2009.

[39] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A Reservoir Computing approach," *Physical Review Letters*, vol. 120, no. 2, p. 024102, 2018.

[40] B. Ramadevi and K. Bingi, "Chaotic time series forecasting approaches using Machine Learning techniques: A review," *Symmetry*, vol. 14, no. 5, p. 955, 2022.

[41] S. Shahi, F. H. Fenton, and E. M. Cherry, "A Machine-Learning approach for long-term prediction of experimental cardiac action potential time series using an Autoencoder and Echo State Networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, no. 6, 2022.

[42] S. Shahi, F. H. Fenton, and E. M. Cherry, "Prediction of chaotic time series using Recurrent Neural Networks and Reservoir Computing techniques: A comparative study," *Machine Learning with Applications*, vol. 8, p. 100300, 2022.

[43] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459–467, 1976.

[44] G. W. Beeler and H. Reuter, "Reconstruction of the action potential of ventricular myocardial fibres," *The Journal of Physiology*, vol. 268, no. 1, pp. 177–210, 1977.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance Deep Learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[46] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Data of "Deep Learning for chaos detection"," 2023.

[47] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[49] F. Liu, X. Zhou, J. Cao, Z. Wang, H. Wang, and Y. Zhang, "Arrhythmias classification by integrating stacked bidirectional LSTM and two-dimensional CNN," in *Advances in Knowledge Discovery and Data Mining: 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17,*

2019, Proceedings, Part II 23, pp. 136–149, Springer, 2019.

[50] R. Engelken, "Gradient flossing: Improving gradient descent through dynamic control of jacobians," in *Advances in Neural Information Processing Systems* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds.), vol. 36, pp. 10412–10439, Curran Associates, Inc., 2023.

[51] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[52] M. J. Feigenbaum, "Quantitative universality for a class of nonlinear transformations," *Journal of Statistical Physics*, vol. 19, no. 1, pp. 25–52, 1978.

[53] T. J. Lewis and M. R. Guevara, "Chaotic dynamics in an ionic model of the propagated cardiac action potential," *Journal of Theoretical Biology*, vol. 146, no. 3, pp. 407–432, 1990.

[54] E. M. Cherry and F. H. Fenton, "Visualization of spiral and scroll waves in simulated and experimental cardiac tissue," *New Journal of Physics*, vol. 10, no. 12, p. 125016, 2008.

[55] E. M. Cherry and F. H. Fenton, "A tale of two dogs: Analyzing two models of canine ventricular electrophysiology," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 292, no. 1, pp. H43–H55, 2007.

Graphical Preprocessing Techniques to Generalize Dynamical Behavior Analysis with Deep Learning

Carmen Mayora-Cebollero[1], Ana Mayora-Cebollero[1], Álvaro Lozano[2], Roberto Barrio[1]

[1]Departamento de Matemática Aplicada and IUMA. Computational Dynamics group. Universidad de Zaragoza. Spain
[2]Departamento de Matemáticas and IUMA. Computational Dynamics group. Universidad de Zaragoza. Spain

Abstract

One of the aims of Dynamical Systems is the behavior classification, including escape orbits, chaotic regimes, equilibrium points, and other structured patterns. Recently, Deep Learning has been applied to classify the dynamics of the system in which the network was trained. In this article, we propose to apply graphical preprocessing techniques (in concrete, time series imaging methods) to convert the time series into images used as input in a Convolutional Neural Network. With the use of these techniques, the network is able to generalize the dynamical classification to other discrete dynamical systems where the network has not been trained. That is, we train a network using data from the Logistic map and we use it to classify the dynamics of the circle map and the Hénon map.

1 Introduction

In Dynamical Systems, a key task is detecting and classifying dynamical behaviors, such as chaos, escape orbits, and regular dynamics. Some behaviors, like equilibrium points and escape orbits, can be easily detected. However, the difference between chaotic and regular (non-equilibrium points) orbits is not trivial. Classical methods, such as Lyapunov exponents (LEs) [1], fast chaos indicators [2], or the 0-1 test for chaos [3], have been used to distinguish between regular and chaotic dynamics [4, 5, 6]. However, recently some authors have proposed to use Deep Learning techniques to perform chaos detection [7, 8, 9, 10, 11]. Moreover, Deep Learning has also been used to approximate the full LEs spectrum from single-variable time series [12], or as a data-driven technique to obtain the underlying system of equations and compute LEs with classical techniques [13]. In general, with Deep Learning, good performance results are obtained while computational time is considerably reduced.

Deep Learning (DL) [14, 15] is the branch of Machine Learning devoted to use (Deep) Artificial Neural Networks to learn from data. Convolutional Neural Network (CNN) [16] is a well-known architecture of DL which was introduced in the field of computer vision and is widely used. The success of CNNs motivated the development of time series imaging techniques, that transform time series into images to be used input for CNNs, like Gramian Angular Field (GAF) [17, 18] and Markov Transition Field (MTF) [17, 18], as well as the application of existing methods like Recurrence Plots (RPs) [19]. In [20], the possible Machine Learning workflows that can be used for classification and prediction with RPs in the area of Dynamical Systems are summarized. In particular, two main procedures are possible:

1. Some features of the RP are obtained with Recurrence Quantification Analysis, and later used as input in different Machine Learning methods for classification and prediction tasks. For example, this workflow has been used in [21] for dynamical classification (periodic, chaotic, hyperchaotic, and noisy behavior).

2. The Recurrence Plot is used as input in a CNN for prediction and classification tasks.

Since the latter can be applied to other time series imaging techniques, we focus our work on it. We use GAF, MTF and RP as graphical preprocessing techniques to transform time series into images. Such images are the input of a CNN built to perform dynamical behavior analyses on discrete dynamical systems.

In particular, we train a CNN only with data from the Logistic map and then we use this trained network to study other systems. We show that the dynamical behavior analysis can be generalized through different discrete dynamical systems.

This paper is organized as follows. In Section 2 we review the time series imaging techniques that we use as graphical preprocessing techniques to transform a time series into an image. In Section 3 we explain in detail the general workflow that we use for dynamical behavior analysis. In Section 4 we perform our dynamical behavior analyses in discrete dynamical systems. Finally, in Section 5 we draw some conclusions.

All DL experiments in this article have been performed using PyTorch [22].

2 Methods

Time series imaging techniques are used to transform a time series into an image that can be used as input in a Convolutional Neural Network. To generalize the dynamical classification task we will use three different time series imaging methods as graphical preprocessing techniques: Gramian Angular Field (GAF), Markov Transition Field (MTF), and Recurrence Plot (RP). In particular, all these techniques transform time series of length L into $L \times L$ matrices, which can be huge if L is a large number. For this reason, techniques as Piecewise Aggregate Approximation (PAA) can be used to reduce the length of the time series before applying the graphical preprocessing methods.

2.1 Piecewise Aggregate Approximation (PAA)

The Piecewise Aggregate Approximation (PAA) [23] is a technique used to transform a time series of length L into one of length ℓ with $1 \leq \ell \leq L$ (for simplicity we assume that L/ℓ is an integer).

If $x_1, x_2, \ldots, x_L$ is the original time series, the one reduced by PAA is $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_\ell$ where

$$\tilde{x}_i = \frac{\ell}{L} \sum_{j=\frac{L}{\ell}(i-1)+1}^{\frac{L}{\ell}i} x_j,$$

for $i = 1, 2, \ldots, \ell$. Notice that PAA is a dimensionality reduction technique in which the time series are divided into frames of equal size and the mean value is computed for each frame. If $\ell = L$, then the time series is not reduced at all, and if $\ell = 1$, the result of PAA is the mean of the whole time series.

This operation is in fact an average pooling of the CNNs with both kernel size and stride equal to L/ℓ.

2.2 Gramian Angular Field (GAF)

The Gramian Angular Field (GAF), introduced in [17, 18], is a time series imaging technique that transforms a one-dimensional time series into an image (Gramian matrix). In the literature we can find several applications of GAF [24, 25, 26, 27]. It has two variants known as Gramian Angular Summation Field (GASF) and Gramian Angular Difference Field (GADF). Consider a one-dimensional time series $X = x_1, x_2, \ldots, x_L$ and think on it as a row vector. Define $\hat{X}$ as the time series X normalized to the interval $[-1, 1]$ or $[0, 1]$. For each normalized value $\hat{x}_i \in \hat{X}$, define $\varphi_i = \arccos \hat{x}_i$. The angle $\varphi_i \in [0, \pi]$ or $[0, \pi/2]$, depending on whether normalization is to $[-1, 1]$ or $[0, 1]$, respectively. Then the GASF is the matrix

$$\big(\cos(\varphi_i + \varphi_j)\big)_{i,j} = \hat{X}^T \cdot \hat{X} - \sqrt{\mathbf{1} - \hat{X}^2}^{\,T} \cdot \sqrt{\mathbf{1} - \hat{X}^2},$$

and the GADF is the matrix

$$\big(\sin(\varphi_i - \varphi_j)\big)_{i,j} = \sqrt{\mathbf{1} - \hat{X}^2}^{\,T} \cdot \hat{X} - \hat{X}^T \cdot \sqrt{\mathbf{1} - \hat{X}^2},$$

where v^T is the transpose of v and $\mathbf{1} \in \mathbb{R}^L$ is the all-ones vector.

Notice that both GASF and GADF preserve temporal dependency, and contain information about the autocorrelation of the original time series. However, those matrices are usually big since they are square matrices of size L, the length of the time series. In this cases, PPA is applied to reduce the dimensionality of the matrices.

2.3 Markov Transition Field (MTF)

The Markov Transition Field (MTF) presented in [17, 18] is a time series imaging technique that transforms a one-dimensional time series into an image (the Markov Transition Field matrix). This method is inspired by that explained in [28], but in MTF the time domain information is preserved. Several applications of MTF can be found in the literature [29, 30, 31]. For a time series $X = x_1, x_2, \ldots, x_L$, the MTF matrix is obtained as follows. First, the points x_i are divided into Q quantile bins $(q_1, q_2, \ldots, q_Q)$. Then, a weighted adjacency matrix W of dimension $Q \times Q$ is built with transitions among quantile bins: the (i, j)-th element of the matrix corresponds to the number of times that an element of quantile q_i is followed by an element of q_j in the time series X. The matrix W is normalized in such a way that the sum by rows is 1. In this form, W is a transition matrix of a finite Markov chain. Finally, the (i, j)-th element of the MTF matrix is the (m, n)-th element of matrix W with $x_i \in q_m$ and $x_j \in q_n$.

As in the GAF method, the obtained image can be large. In the case of MTF, in [17, 18] the authors propose to use a blurring kernel to reduce dimensions once the MTF matrix has been obtained. However, in our experiments we do not use this reduction technique, but instead we use PAA and then we obtain the MTF matrix of the reduced time series.

2.4 Recurrence Plot (RP)

The Recurrence Plot (RP) is a method that was born in the area of Dynamical Systems to visualize and study the recurrences of a system. These recurrences can be used to obtain information of the corresponding system, and for example, in coupled systems, to study their interaction. RP was introduced in [19] and has been widely studied later [32, 33, 34, 35]. In fact, it has been extensively used [36, 37, 38, 20, 21]. The RP technique consists in transforming a time series into an image (recurrence matrix). For an n-dimensional time series $X = x_1, x_2, \ldots, x_L$ of length L, the (i, j)-th element of the recurrence matrix is given by

$$\Theta(\epsilon - \|x_i - x_j\|),$$

where $\Theta(\cdot)$ is the Heaviside function (which returns 0 for negative arguments, and 1 otherwise), ϵ is a threshold, and $\|.\|$ is a norm (usually the L_∞-norm). Notice that this matrix is symmetric with respect to the diagonal (which is filled with 1's). The RP (visual representation of the recurrence matrix) is usually depicted with two colors, for example, black for 1 matrix values and white for 0 values. Moreover, both axes are time axes with (by convention) time increasing rightwards and upwards.

As mentioned before, a threshold value ϵ is used in the formulation of the RP. Several rules can be used to establish its value (see [35] for a summary of several possibilities). One of them is to choose ϵ as the q-th percentile, where q is the density of recurrence points in the recurrence matrix (usually called the Recurrence Rate). This rule has some advantages as it preserves the Recurrence Plot density and normalization is not needed to compare the RP of different dynamical systems that can present different scales.

3 General workflow for dynamical behavior analysis

Our workflow for dynamical behavior analysis has 3 steps. First, we directly classify the equilibrium points and the escape orbits. Next, we apply the graphical preprocessing techniques (GAF, MTF or RP) to the remaining orbits. Finally, we use a DL network (trained only with data from the Logistic map) to classify the regular (non-equilibrium points) and the chaotic behavior. In what follows, for simplicity, when we talk about regular dynamics we will refer to any regular orbit except for equilibrium points.

In the first step, we identify equilibrium points and escape orbits, that can be easily detected. We consider that a time series corresponds to an escape orbit if any component of its last point is greater than 10^4. An orbit corresponds to an equilibrium point if we take the last 80 points of the time series and the Euclidean norm of the difference for all the components between two consecutive time points is less than 10^{-3}.

In the second step, we have time series of length 896 and we use the PAA method to reduce their dimension to 224 (standard dimension in image classification tasks with CNN architectures as AlexNet). Then, a graphical preprocessing technique (GAF, MTF or RP) is applied to convert each time series into an image. Notice that the GAF and MTF techniques transform one-dimensional time series into images, but with RP, time series of whatever dimension can be converted into a unique image. To perform all the analyses under the same conditions, we use one-dimensional time series for all the techniques (for dynamical systems with more than one dimension, we will choose one of the possible dimensions). In all the analyses, we use GASF version of GAF method, for MTF we consider $Q = 8$ quantile bins, and in the RP method we use the L^{∞}-norm and threshold ϵ is the 0.1-th percentile.

Finally, in the third (and last) step, the graphical representations of the time series are used as inputs in a DL architecture that classify them into regular or chaotic. In concrete, we use the AlexNet architecture trained from scratch with regular and chaotic orbits of the well-known Logistic map [39].

In Figure 1 we illustrate how a regular (panel A) and a chaotic (panel B) orbit of a discrete dynamical system (the circle map) are modified with PAA technique. The original time series of length 896 are depicted in the top part of each panel. However, for a better visualization, just the last 300 points are represented. Notice that for both examples, the time series are reduced with PAA while conserving their behavior. We also show the representation of the matrices obtained with each of

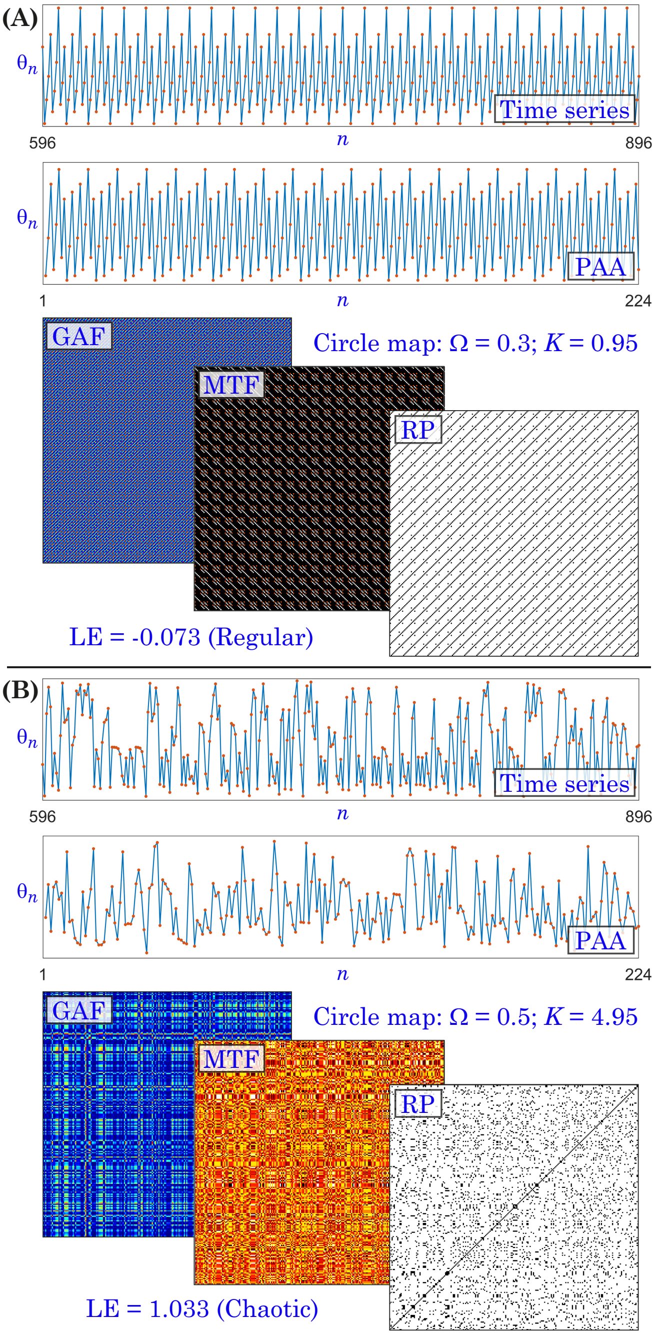


Figure 1: Example of how PAA, GAF, MTF and RP transform a regular (panel A) and a chaotic (panel B) orbit. In particular, both time series are from the circle map with initial condition 0.25. The parameter values are $(\Omega, K) = (0.3, 0.95)$ for regular sample, and $(\Omega, K) = (0.5, 4.95)$ for chaotic one.

the graphical preprocessing techniques (colormaps of each technique have been selected for proper visualization) applied to the reduced time series obtained with PAA. This illustrates the different CNN inputs for each time series depending on the imaging method. The images obtained with each graphical preprocessing technique show repeated structures in the case of regular behavior (for example, diagonal parallel lines in RP) and not any repeated pattern in the chaotic orbit.

3.1 AlexNet architecture

Several CNN standard architectures as AlexNet [40, 41], VGG [42] and ResNet [43] can be found in the literature. For our task (dynamical behavior analysis) we want to show that a not very complicated network can give amazing results. We use the AlexNet architecture.

AlexNet [40, 41] is a CNN architecture that has 5 2D convolutional layers, all of them followed by the ReLU (Rectified Linear Unit) activation function and also by a 2D max pooling layer (kernel size 3 and stride 2) in the case of the first, second and fifth layers. The number of channels on each of the 5 convolutional layers is 64, 192, 384, 384 and 256, with [kernel size-stride-padding] equal to $[11-4-2]$, $[5-1-2]$, $[3-1-1]$, $[3-1-1]$ and $[3-1-1]$, respectively. After the last max pooling layer, a 2D adaptive pooling layer with target output size 6×6 is used. Then three linear layers with 4096, 4096 and 2 neurons (we consider 2 classes: chaotic and regular behavior) are applied sequentially. Except for the last linear layer, dropout with probability $p = 0.5$ is used before each linear layer, and the ReLU activation function is applied later.

3.2 Training in the Logistic map

The AlexNet architecture is trained from scratch with regular and chaotic orbits of the Logistic map.

The Logistic map [39] is a well-known one-dimensional discrete dynamical system that describes the dynamics of animal populations. Despite its simplicity, it represents complicated behaviors as chaos. The Logistic map is given by the equation

$$x_{n+1} = \alpha\, x_n\, (1 - x_n), \tag{1}$$

where x_n corresponds with the variable in the n-th iteration and α is the bifurcation parameter. With classical techniques, the maximum Lyapunov exponent of this one-dimensional dynamical system [44] is computed as

$$\mathrm{LE} = \frac{1}{T}\sum_{i=1}^{T} \log(|\alpha(1-2x_i)|), \tag{2}$$

with $\log(\cdot)$ the natural logarithm and $[1, T]$ the interval time involved in LE computation.

To create the train, validation and test datasets needed to train the network (AlexNet) and check that learning was successful, we have proceed as follows. We create four raw datasets. For each dataset, we have fixed the initial condition x_0 ($x_0 \in \{0.5, 0.75, 0.8, 0.9\}$). Moreover, we take $12,000$ equidistant values in the interval $[0, 4)$ for the bifurcation parameter for each dataset. Each time series has a length of 896 and corresponds to the last 896 points obtained applying (1) over $12,000$ time steps. To obtain the maximum Lyapunov exponent that is used as label in the training, validation and test processes, we use $1,000$ time steps as transient, and compute the LE with the next $11,000$ time points using equation 2. Notice that these four raw datasets corresponds to those in [45] (with different length for the time series).

The raw datasets corresponding to initial condition $x_0 = 0.5$ and $x_0 = 0.9$ are devoted to create train dataset, this with $x_0 = 0.75$ is used for validation, and the one with $x_0 = 0.8$ for test. For each case, train, validation and test sets, the equilibrium points are discarded (the detection of an equilibrium point is trivial and it is not necessary to use DL) and similar samples within and through datasets are deleted. We consider similar samples those whose difference is less than 10^{-4} in infinity norm. Then, the time series within each dataset are shuffled to ensure dynamical variability. Finally, $2,000$ regular and $2,000$ chaotic time series are selected in the train set (batch size 128), $1,000$ of each class in validation (batch size 100), and 175 of each type in test set (batch size 175).

With training and validation datasets we train from scratch the AlexNet architecture (using early stopping technique, that is, the final network is the one with the lowest loss value for validation set during training) for 200 epochs. The optimizer is the Stochastic Gradient Descent with learning rate 0.001, weight decay $5\cdot 10^{-4}$ and momentum 0.9. As expected for a DL classification task, we use the Cross-Entropy loss function. Once trained, the test set is used to check that learning process was successful. In Table 1, the values of the loss and accuracy (percentage of samples that were properly classified by the network) for the training, validation and test datasets are given. The accuracy for each dynamical behavior (regular and chaotic) is indicated for the test set. All the results are in the format mean±standard deviation as we train 10 networks with weights and biases randomly initialized to show that results are independent of the initial conditions of the trainable parameters of the DL architecture. The results obtained when using GAF are in first row, those from MTF are in second row, and third row contains the values for RP. Notice that for all datasets the mean loss value is less than 0.05 and the mean accuracy is greater than 97%, which are good results. Moreover, the deviation is very small for loss and accuracy values, indicating that the performance for the 10 (randomly initialized) networks is similar. For each graphical preprocessing technique, the results of accuracy for each behavior in test set are similar, so characteristics of both type of orbits have been learned. Comparing results across the different techniques, we can conclude that there is not a significant difference between the values for

Table 1: Loss and accuracy in mean±standard deviation format for the training, validation and test sets from the Logistic map. Accuracy (Acc.) of the regular (non-equilibrium points) and chaotic samples is also indicated for test dataset.

	Train		Validation		Test			
	Loss	Accuraccy (%)	Loss	Accuraccy (%)	Loss	Accuraccy (%)	Acc. Regular (%)	Acc. Chaotic (%)
GAF	0.005 ± 0.002	99.836 ± 0.085	0.015 ± 0.002	99.635 ± 0.059	0.032 ± 0.009	99.114 ± 0.237	98.229 ± 0.475	100.000 ± 0.000
MTF	0.008 ± 0.001	99.657 ± 0.084	0.011 ± 0.001	99.485 ± 0.084	0.045 ± 0.008	98.567 ± 0.367	97.142 ± 0.731	100.000 ± 0.000
RP	0.008 ± 0.002	99.710 ± 0.101	0.010 ± 0.001	99.550 ± 0.071	0.039 ± 0.013	98.500 ± 0.582	97.753 ± 1.210	99.261 ± 0.358

GAF, MTF and RP. For all techniques the training process was successful.

4 Generalization of the dynamical behavior analysis of discrete dynamical systems

We use the workflow described in Section 3 with the networks trained with the Logistic map datasets to perform dynamical behavior analyses in different discrete dynamical systems. In particular, we show in Subsection 4.1 the performance in chaos detection task (classification of the behavior of a dynamical system) in an α-parametric line of the Logistic map, the system used to train the networks. As our main objective is to generalize the use of our trained networks to the dynamical classification of other discrete dynamical systems, in Subsection 4.2 and Subsection 4.3 we apply the workflow to perform dynamical behavior analyses of the circle map and Hénon map (systems not used during training). Notice that with this method, once we have trained the networks in the Logistic map, the model of the discrete dynamical systems is just used to obtain the time series but it is not involved in the way of classifying the time series (in the case of LEs computation used as ground truth in this paper, we have to particularize the LEs formula to each dynamical system).

The numerical results in this section are in the format mean±standard deviation and are obtained with the 10 trained networks. The graphical representations correspond to just one of the 10 trained networks.

4.1 Dynamical behavior analysis in the Logistic map

We consider a one-parameter line of the Logistic map (the one studied in [10]) with $x_0 = 0.6$ and $12,000$ equidistant points for the bifurcation parameter $\alpha \in [0, 4)$. The time series have the same number of points and are integrated under the same conditions as the datasets for training (the LEs used to compare the performance of the network results are also computed in the same way as before). As stated in Section 3, we filter out 'trivial' dynamics, we reduce the remaining time series with PAA, we transform them into images using GAF, MTF or RP. Finally, such images are used as input in the trained AlexNets to perform dynamical behavior analysis.

In Figure 2, the dynamical behavior analysis in the one-parameter line is depicted. Green color is used to identify equilibrium points, blue for other regular dynamics, and red for chaos. In panel A we have the results obtained with the classical technique of LEs (ground truth). In panels B, C and D we have the classification given by DL with GAF, MTF and RP as graphical preprocessing techniques, respectively. For a better comparison between different methods, in panels A1, B1, C1 and D1 we have performed a magnification for $\alpha \in [3.5, 4)$. Notice that the analyses performed with the different methods are similar. All the regular and chaotic regions are clearly detected, except for the first boundary between regular (non-equilibrium points) behavior and chaos, where there is a small difference between the ground truth and the results obtained with DL (see magnifications A1, B1, C1 and D1). However, as it is a boundary region it can be expected that DL fails independently of the graphical preprocessing technique.

To quantitatively evaluate the quality of the results, we have the accuracy values (in the format mean±standard deviation for the 10 randomly initialized networks) in Table 2. Notice that for all the graphical preprocessing techniques the mean accuracy is greater than 97% (with a small deviation), and the difference of the performance between regular and chaotic samples is considerably small.

Table 2: Accuracy results (in the format mean±standard deviation) for the one-parameter line of the Logistic map obtained with initial condition $x_0 = 0.6$ and $\alpha \in [0, 4)$.

	Accuraccy (%)	Acc. Regular (%)	Acc. Chaotic (%)
GAF	98.600 ± 0.207	97.895 ± 0.343	99.818 ± 0.115
MTF	98.740 ± 0.047	98.179 ± 0.095	99.709 ± 0.089
RP	99.080 ± 0.065	98.632 ± 0.115	99.855 ± 0.073

4.2 Dynamical behavior analysis in the Circle map

The Circle map [46] (also known as Sine-Circle map) is a one-dimensional discrete dynamical system that maps the circle onto itself. It is given by

$$\theta_{n+1} = \theta_n + \Omega - K \frac{\sin(2\pi\, \theta_n)}{2\pi} \mod 1,$$

where θ_n is the variable at the n-th iteration, and Ω and K are the bifurcation parameters, representing the natural frequency and the coupling strength, respectively. As

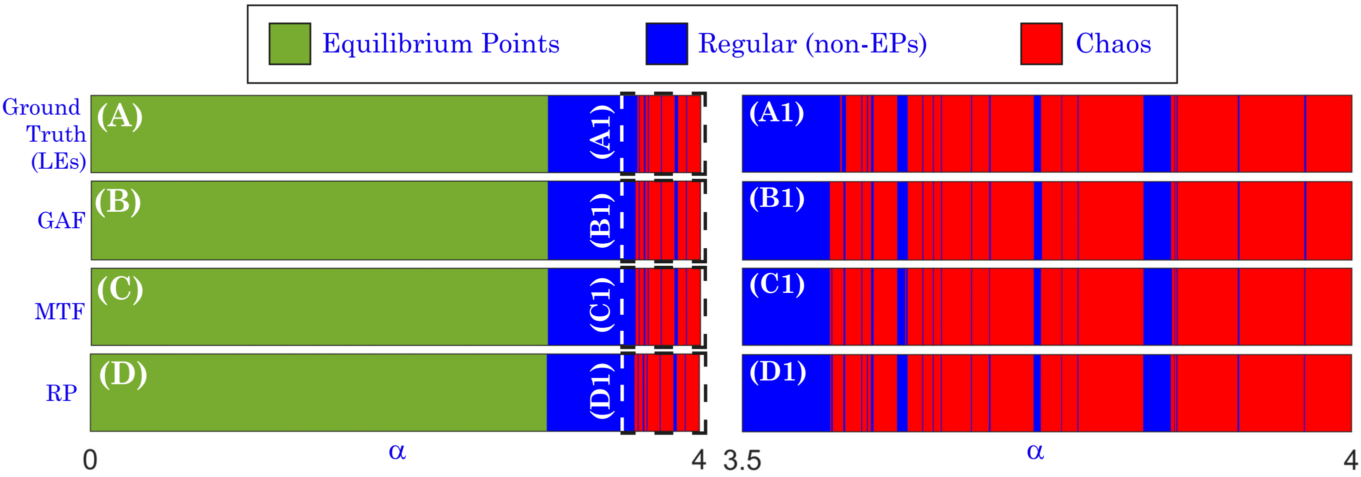


Figure 2: One-parameter analysis of the Logistic map for initial condition $x_0 = 0.6$ and $\alpha \in [0, 4)$. Panel A corresponds to dynamical classification obtained with classical method of LEs. Panels B, C and D are obtained with GAF, MTF and RP as preprocessing techniques, respectively, and the trained networks. In panels A1, B1, C1 and D1, a magnification of region $\alpha \in [3.5, 4)$ is given for each case. The label *Regular (non-EPs)* corresponds to the orbits with regular behavior that are not equilibrium points (EPs).

the circle map is a one-dimensional dynamical system, the maximum Lyapunov exponent is given by

$$\mathrm{LE} = \frac{1}{T}\sum_{i=1}^{T} \log(|1 - K\cos(2\pi\theta_i)|),$$

with $\log(\cdot)$ the natural logarithm and $[1, T]$ the interval time involved in LE computation.

We are going to use the networks trained in the Logistic map with each graphical preprocessing technique (GAF, MTF and RP) to study the different dynamical regimes in the (K, Ω)-parametric plane of the circle map. In particular, $K \in [1.25, 5]$ and $\Omega \in [0, 1]$ (biparametric plane similar to the ones in [47, 48]), and we consider $1,000$ equidistant points for each parameter (that is, 10^6 points in the whole biparametric region). The time series (and the LEs used for ground truth classification) are computed equivalently to the Logistic map case. Notice that the trained networks that are used for dynamical classification have not been trained with time series from this map, only Logistic map orbits have been used. So, in the workflow used to perform dynamical behavior analysis, just the time series of the circle map are involved and no other information of this system is used.

In Figure 3 we have depicted the dynamical behavior analysis in the aforementioned plane using classical technique of LEs (panel A1), DL with GAF (panel B1), DL with MTF (panel C1), and DL with RP (panel D1) for fixed initial condition $\theta_0 = 0.25$. The results of panel A1 (classical technique) are used as ground truth to check the performance of DL with the different graphical preprocessing techniques. Green regions correspond to equilibrium points (trivial classification), blue ones with regular (non-equilibrium) dynamics, and red ones with chaos. If we compare the plot of the ground truth (panel A1) and those obtained with DL (panels B1, C1 and D1), they are quite similar. We can see that in general, shrimp-shaped regions [49, 50, 51, 52] are properly identified. However, it should be noted that in the case of the GAF method (panel B1) some regions are not properly detected (marked with rectangles with dashed lines). In particular, there are some tiny shrimp-shaped regular regions that are not detected properly and therefore considered chaotic (compare panels A1.1 and B1.1). Moreover, for large values of K there are some samples not correctly classified (compare panels A1.2 and B1.2). These small detection errors are not committed when MTF or RP are used (see panels C1.1-D1.1 and C1.2-D1.2 and compare with ground truth panels A1.1-A1.2 and GAF panels B1.1-B1.2).

Therefore, dynamical behavior analyses seem to be successful when using DL (trained in Logistic map) and the graphical preprocessing techniques, with MTF and RP performing better than GAF. We can check this with the accuracy results that are given in Table 3 in the format mean±standard deviation for the 10 trained networks. In all cases, the mean accuracy is greater than 95% (with small standard deviation) and the accuracies for both dynamical regimes are quite similar. The lowest accuracy is obtained when GAF is used as the graphical preprocessing technique. The other two methods (MTF and RP) show a very similar performance.

Table 3: Accuracy results (in the format mean±standard deviation) for the biparametric plane ($\Omega \in [0, 1]$ and $K \in [1.25, 5]$) of the (sine) circle map obtained with the initial condition $\theta_0 = 0.25$.

	Accuracy (%)	Acc. Regular (%)	Acc. Chaotic (%)
GAF	96.993 ± 0.168	99.151 ± 0.133	95.658 ± 0.323
MTF	98.503 ± 0.055	99.908 ± 0.013	97.633 ± 0.096
RP	98.206 ± 0.049	99.976 ± 0.007	97.111 ± 0.081

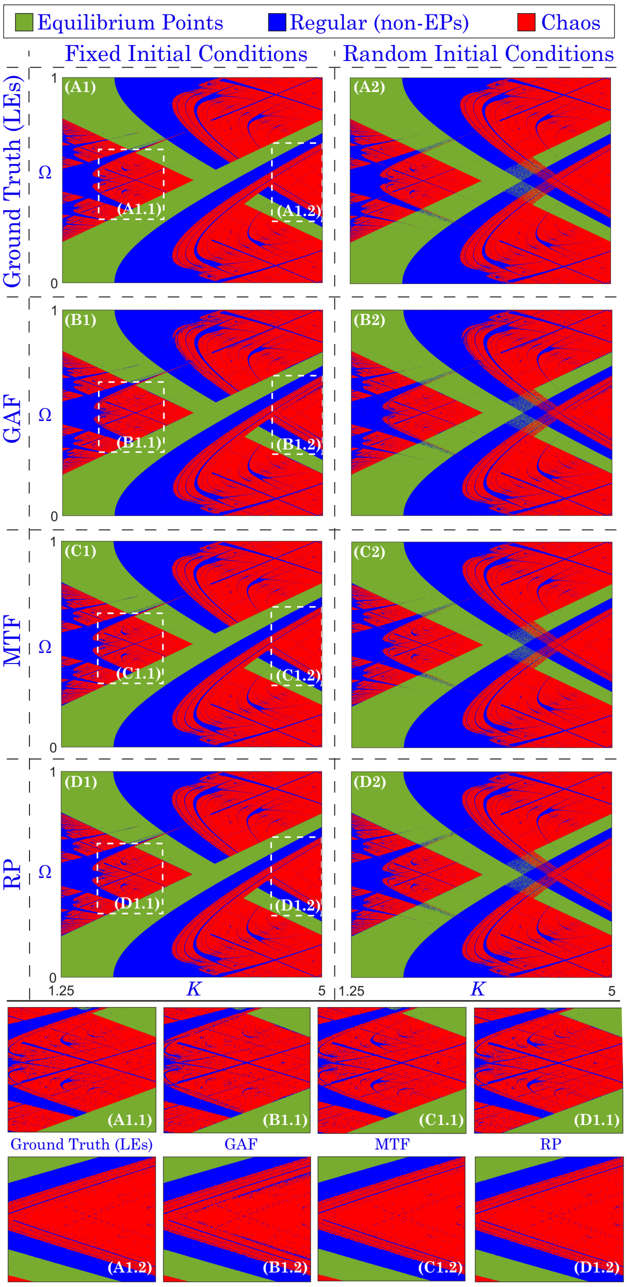


Figure 3: Analyses of the biparametric plane ($K \in [1.25, 5]$ and $\Omega \in [0, 1]$) of the (sine) circle map. Panels A1 and A2 are the studies obtained with the classical technique of LEs when fixed and random initial conditions are used. Panels B1, B2, C1, C2, D1 and D2 are the studies obtained using a DL network and GAF, MTF and RP as the graphical preprocessing techniques with fixed and random initial conditions. The fixed initial condition is $\theta_0 = 0.25$ and the random initial conditions are taken in the interval $[0, 1]$. Panels A1.1, A1.2, B1.1, B1.2, C1.1, C1.2, D1.1 and D1.2 are magnifications of the regions marked with a dashed white rectangle in panels A1, B1, C1 and D1. The label *Regular (non-EPs)* corresponds to the orbits with regular behavior that are not equilibrium points (EPs).

Now, in the same biparametric plane we consider random initial conditions (in the interval $[0, 1]$) to check if multistability can be detected with DL (and the different preprocessing techniques). The results are depicted in the right column of Figure 3. In panels A2, B2, C2 and D2, we can observe the same blurred regions that indicate the existence of multistability in the system. Therefore, DL is able to detected this dynamical phenomenon.

With these studies in the circle map, we have shown that DL with a graphical preprocessing technique can be used to classify the dynamics in the parametric space of a dynamical system not used to train the DL network. In fact, especially with MTF and RP techniques, we obtain a detailed dynamical behavior analysis (even the tiny shrimp-shaped regions have been detected). Moreover, it also allows us to detect multistability when random initial conditions are considered.

4.3 Dynamical behavior analysis in the Hénon map

The Hénon map [53] is a two-dimensional discrete dynamical system given by

$$\begin{cases} x_{n+1} = 1 + y_n - a\,x_n^2, \\ y_{n+1} = b\,x_n, \end{cases}$$

where (x_n, y_n) are the variables at discrete time n, and a and b are the parameters. It contains strange attractors for some values of its parameters. We use the algorithm in [54] to compute the two Lyapunov exponents of this two-dimensional system.

We want to proceed as in previous cases, but this map has two variables instead of one as the Logistic map or the circle map, and GAF and MTF only work with one-dimensional time series. Therefore, we restrict ourselves to the x-variable of the time series. For fair analyses, we also use RP with just one variable too (although all could be used). That is, we only use partial information of the system to classify the dynamics with the graphical preprocessing techniques and the DL network.

We will use the graphical preprocessing techniques and the DL networks trained in the Logistic map to detect the different dynamical regimes in the (a, b)-parameteric plane of the Hénon map. We consider $a \in [1.4, 1.7]$ and $b \in [0.05, 0.25]$ (same biparametric plane as the one studied in [55]) with 1,000 equidistant points for each parameter in the given ranges (we have 10^6 points in the whole biparametric plane). The time series (and the LEs used for ground truth classification) are computed equivalently to the Logistic map case. Remember that the networks that we use for the dynamical classification are only trained with orbits from the Logistic map, and no extra information of the Hénon map among the time series is used in the analyses.

In Figure 4 we have the results obtained for the dynamical behavior analysis in the (a, b)-parameteric plane with classical technique of LEs (panel A), DL with GAF (panel B), DL with MTF (panel C), and DL with RP

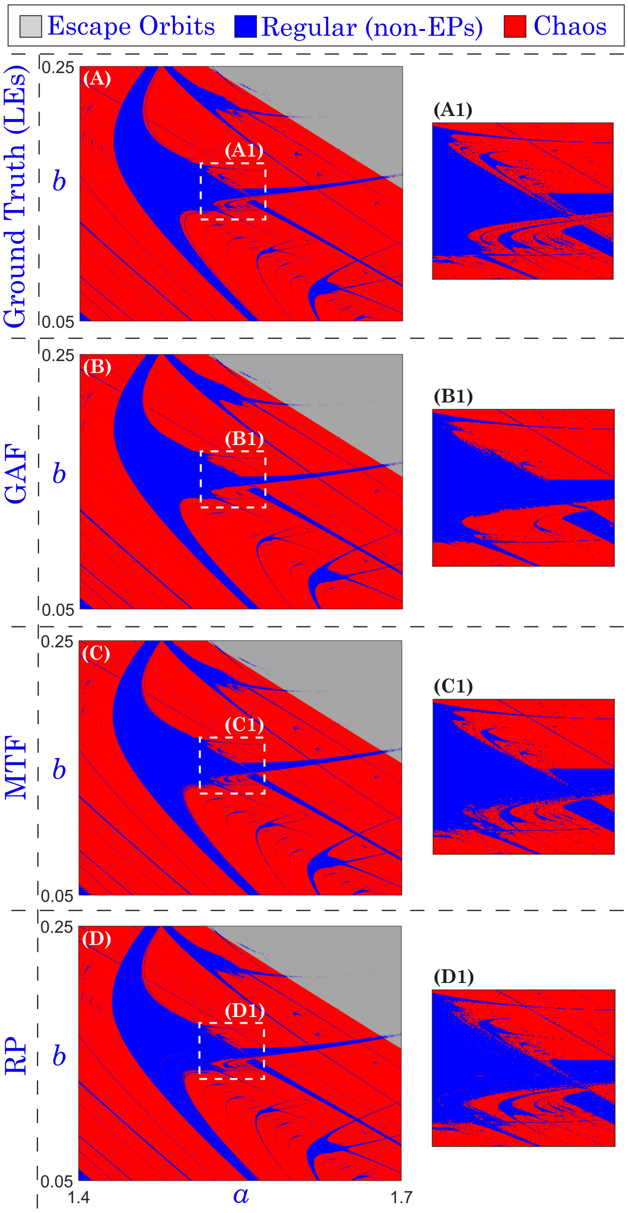


Figure 4: Analyses of the biparametric plane ($a \in [1.4, 1.7]$ and $b \in [0.05, 0.25]$) of the Hénon map. Panel A includes the study obtained with the classical technique of LEs. Panels B, C and D show the results obtained with DL and GAF, MTF and RP as graphical preprocessing techniques. The initial condition is $(x_0, y_0) = (1, 1)$ in all cases. Panels A1, B1, C1 and D1 are magnifications of the regions marked with a dashed white rectangle in panels A, B, C and D, respectively. The label *Regular (non-EPs)* corresponds to the orbits with regular behavior that are not equilibrium points (EPs).

(panel D). The initial conditions are fixed to $(x_0, y_0) = (1, 1)$ in all cases. As in previous analyses, the results of classical technique of LEs (panel A) are the ground truth used as reference to check the performance of the different graphical preprocessing techniques. Blue regions correspond to regular (non-equilibrium) dynamics, red ones to chaos, and color gray is used for escape orbits. At first sight, if we compare the ground truth (panel A) and the plots obtained with DL (panels B, C and D), all the dynamical behavior classifications are quite similar. In panels A1, B1, C1 and D1, we provide a magnification (see white dashed rectangle in panels A, B, C and D) to check the level of detail of each of the preprocessing techniques. It is clear that GAF is the technique providing less details as most of the tiny shrimp-shaped regions are not detected properly (compare panels A1 and B1). MTF and RP techniques seem to give better quality in detections and have similar performance (compare panels A1 and C1, and panels A1 and D1). With the accuracy results of Table 4 we can confirm that all the graphical preprocessing methods perform quite well (mean accuracies are greater than 95.5% for all techniques and standard deviations are relatively small). The accuracy for regular class with GAF is lower than for the other methods, which supports the idea deduced from Figure 4 (it seems to have problems to detect some regular regions as, for example, tiny shrimp-shaped zones).

Table 4: Accuracy results (in the format mean±standard deviation) for the biparametric plane ($a \in [1.4, 1.7]$ and $b \in [0.05, 0.25]$) of the Hénon map obtained with the initial conditions $(x_0, y_0) = (1, 1)$.

	Accuracy (%)	Acc. Regular (%)	Acc. Chaotic (%)
GAF	97.666 ± 0.077	95.769 ± 0.343	98.143 ± 0.164
MTF	97.819 ± 0.091	98.466 ± 0.122	97.656 ± 0.144
RP	97.674 ± 0.100	99.513 ± 0.158	97.213 ± 0.152

With these studies in the Hénon map we have shown that with a network trained with data from the Logistic map and using graphical preprocessing techniques (GAF, MTF or RP), dynamical behavior analyses can be performed properly with just information from single-variable time series. The three preprocessing techniques perform quite well, but a more accurate and detailed analysis is obtained with MTF and RP.

5 Conclusions

In this paper, we have shown that graphical preprocessing techniques can be used to generalize dynamical behavior analysis from the Logistic map to other discrete dynamical systems. In particular, we use Gramian Angular Field (GAF), Markov Transition Field (MTF) and Recurrence Plot (RP) as graphical preprocessing techniques that transform single-variable time series into images that embed their dynamics. The obtained images can be used as input to a Convolutional Neural Network (CNN) to perform dynamical classification into regular (non-equilibrium) and chaotic behavior.

We use a dataset with (a not large number of) time series from the Logistic map to train (from scratch) a well-known and not very complicated CNN architecture called AlexNet. Once trained, we follow this workflow: trivial-detected dynamics (equilibrium points and escape orbits) are classified, the remaining time series are embedded into images with a time series imaging method used as a graphical preprocessing technique (GAF, MTF or RP). Finally, the trained AlexNet is used to perform dynamical classification into regular (non-equilibrium) and chaotic behavior. This is applied to the circle map and the Hénon map (systems not used during training), obtaining good results (accuracy greater than 95%). In general, the three techniques are able to detect both behaviors and determine multistability regions. MTF and RP seem to perform better and are even able to define properly tiny shrimp-shaped regions.

With the AlexNet trained in the Logistic map and the use of graphical preprocessing techniques, we obtain a powerful tool. It is able to perform dynamical behavior analysis with good accuracy in different dynamical systems without the necessity of particularize the algorithm to the system. Notice that in the case of the classical technique of LEs (used as ground truth in this paper), the model of each system is involved in the computation of the LEs, and therefore, in the process of classification. This does not occur in our approach: once the time series is computed, the information provided by the model is not used.

Acknowledgments

RB, AL, AMC and CMC have been supported by the Spanish Research project PID2021-122961NB-I00. RB, AMC and CMC have been supported by the European Regional Development Fund and Diputación General de Aragón (E24-23R). RB has been supported by the European Regional Development Fund and Diputación General de Aragón (LMP94-21). AL has been supported by the Spanish Research project PID2022-140556OB-I00 and by the European Regional Development Fund and Diputación General de Aragón (E22-23R).

Data Availability

Data available on request from the authors.

References

[1] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, 1985.

[2] R. Barrio, "Theory and applications of the orthogonal fast Lyapunov indicator (OFLI and OFLI2) methods," *Chaos Detection and Predictability*, pp. 55–92, 2016.

[3] G. A. Gottwald and I. Melbourne, "The 0-1 test for chaos: A review," *Chaos detection and predictability*, pp. 221–247, 2016.

[4] R. Barrio and S. Serrano, "A three-parametric study of the Lorenz model," *Physica D: Nonlinear Phenomena*, vol. 229, no. 1, pp. 43–51, 2007.

[5] R. Barrio, F. Blesa, and S. Serrano, "Qualitative analysis of the Rössler equations: Bifurcations of limit cycles and chaotic attractors," *Physica D: Nonlinear Phenomena*, vol. 238, no. 13, pp. 1087–1100, 2009.

[6] R. Halfar, "Dynamical properties of Beeler–Reuter cardiac cell model with respect to stimulation parameters," *International Journal of Computer Mathematics*, vol. 97, no. 1-2, pp. 498–507, 2020.

[7] N. Boullé, V. Dallas, Y. Nakatsukasa, and D. Samaddar, "Classification of chaotic time series with Deep Learning," *Physica D: Nonlinear Phenomena*, vol. 403, p. 132261, 2020.

[8] A. Celletti, C. Gales, V. Rodriguez-Fernandez, and M. Vasile, "Classification of regular and chaotic motions in hamiltonian systems with Deep Learning," *Scientific Reports*, vol. 12, no. 1, p. 1890, 2022.

[9] A. Corbetta and T. G. de Jong, "How neural networks learn to classify chaotic time series," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 12, 2023.

[10] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 7, 2023.

[11] V. Vismaya, B. V. Nair, and S. S. Muni, "Deep Learning for prediction and classifying the dynamical behaviour of piecewise-smooth maps," *Franklin Open*, p. 100180, 2024.

[12] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, "Full Lyapunov exponents spectrum with deep learning from single-variable time series," *Physica D: Nonlinear Phenomena*, p. 134510, 2024.

[13] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using Machine Learning to replicate chaotic attractors and calculate Lyapunov exponents from data," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 12, 2017.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[15] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.* " O'Reilly Media, Inc.", 2022.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," in *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 3939–3945, 2015.

[18] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled Convolutional Neural Networks," in *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, 2015.

[19] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle, "Recurrence plots of dynamical systems," *EUROPHYSICS LETTERS*, vol. 5, no. 9, pp. 973–977, 1987.

[20] N. Marwan and K. H. Kraemer, "Trends in recurrence analysis of dynamical systems," *The European Physical Journal Special Topics*, vol. 232, no. 1, pp. 5–27, 2023.

[21] D. Thakur, A. Mohan, G. Ambika, and C. Meena, "Machine Learning approach to detect dynamical states from recurrence measures," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 34, no. 4, 2024.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[23] E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 285–289, 2000.

[24] G. Zhang, Y. Si, D. Wang, W. Yang, and Y. Sun, "Automated detection of myocardial infarction using a Gramian Angular Field and Principal Component Analysis network," *IEEE Access*, vol. 7, pp. 171570–171583, 2019.

[25] K. P. Thanaraj, B. Parvathavarthini, U. J. Tanik, V. Rajinikanth, and S. Kadry, "Implementation of Deep Neural Networks to classify EEG signals using Gramian Angular Summation Field for epilepsy diagnosis," *arXiv preprint arXiv:2003.04534*, 2020.

[26] H. Xu, J. Li, H. Yuan, Q. Liu, S. Fan, T. Li, and X. Sun, "Human activity recognition based on Gramian Angular Field and Deep Convolutional Neural Network," *IEEE Access*, vol. 8, pp. 199393–199405, 2020.

[27] A. Shankar, H. K. Khaing, S. Dandapat, and S. Barma, "Epileptic seizure classification based on Gramian Angular Field transformation and Deep Learning," in *2020 IEEE Applied Signal Processing Conference (ASPCON)*, pp. 147–151, IEEE, 2020.

[28] A. S. Campanharo, M. I. Sirer, R. D. Malmgren, F. M. Ramos, and L. A. N. Amaral, "Duality between time series and networks," *PloS one*, vol. 6, no. 8, p. e23378, 2011.

[29] R. Zhang, F. Zheng, and W. Min, "Sequential behavioral data processing using Deep Learning and the Markov Transition Field in online fraud detection," *arXiv preprint arXiv:1808.05329*, 2018.

[30] A. Shankar, S. Dandapat, and S. Barma, "Discrimination of types of seizure using brain rhythms based on Markov Transition Field and Deep Learning," *IEEE Open Journal of Instrumentation and Measurement*, vol. 1, pp. 1–8, 2022.

[31] L. Ji, Z. Wei, J. Hao, and C. Wang, "An intelligent diagnostic method of ECG signal based on Markov Transition Field and a ResNet," *Computer Methods and Programs in Biomedicine*, vol. 242, p. 107784, 2023.

[32] M. Thiel, M. C. Romano, and J. Kurths, "How much information is contained in a Recurrence Plot?," *Physics Letters A*, vol. 330, no. 5, pp. 343–349, 2004.

[33] M. C. Romano, M. Thiel, J. Kurths, and W. von Bloh, "Multivariate Recurrence Plots," *Physics letters A*, vol. 330, no. 3-4, pp. 214–223, 2004.

[34] M. Thiel, M. C. Romano, and J. Kurths, "Spurious structures in recurrence plots induced by embedding," *Nonlinear Dynamics*, vol. 44, pp. 299–305, 2006.

[35] N. Marwan, M. C. Romano, M. Thiel, and J. Kurths, "Recurrence plots for the analysis of complex systems," *Physics reports*, vol. 438, no. 5-6, pp. 237–329, 2007.

[36] J. P. Zbilut, M. Koebbe, H. Loeb, and G. Mayer-Kress, "Use of recurrence plots in the analysis of heart beat intervals," in *[1990] Proceedings Computers in Cardiology*, pp. 263–266, IEEE, 1990.

[37] D. F. Silva, V. M. De Souza, and G. E. Batista, "Time series classification using compression distance of recurrence plots," in *2013 IEEE 13th International Conference on Data Mining*, pp. 687–696, IEEE, 2013.

[38] B. M. Mathunjwa, Y.-T. Lin, C.-H. Lin, M. F. Abbod, and J.-S. Shieh, "ECG arrhythmia classification by using a Recurrence Plot and Convolutional Neural Network," *Biomedical Signal Processing and Control*, vol. 64, p. 102262, 2021.

[39] R. M. May, “Simple mathematical models with very complicated dynamics,” *Nature*, vol. 261, no. 5560, pp. 459–467, 1976.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with Deep Convolutional Neural Networks,” *Advances in neural information processing systems*, vol. 25, 2012.

[41] A. Krizhevsky, “One weird trick for parallelizing Convolutional Neural Networks,” *arXiv preprint arXiv:1404.5997*, 2014.

[42] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[43] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[44] J. H. Argyris, G. Faust, and M. Haase, *An exploration of chaos: An introduction for natural scientists and engineers.* Texts on Computational Mechanics Volume VII, 1994.

[45] R. Barrio, Á. Lozano Rojo, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, “Data of “Deep Learning for chaos detection”,” 2023.

[46] R. C. Hilborn, *Chaos and nonlinear dynamics: An introduction for scientists and engineers.* Oxford university press, 2000.

[47] J. B. de Figueiredo and C. P. Malta, “Lyapunov graph for two-parameters map: Application to the circle map,” *International Journal of Bifurcation and Chaos*, vol. 8, no. 02, pp. 281–293, 1998.

[48] N. Y. Ivankov and S. P. Kuznetsov, “Complex periodic orbits, renormalization, and scaling for quasiperiodic golden-mean transition to chaos,” *Physical Review E*, vol. 63, no. 4, p. 046210, 2001.

[49] S. Fraser and R. Kapral, “Analysis of flow hysteresis by a one-dimensional map,” *Physical Review A*, vol. 25, no. 6, p. 3223, 1982.

[50] E. A. Celarier and R. Kapral, “Bistable limit cycle oscillations in chemical systems. I. Basins of attraction,” *The Journal of chemical physics*, vol. 86, no. 6, pp. 3357–3365, 1987.

[51] E. Celarier and R. Kapral, “Bistable limit cycle oscillations in chemical systems. II. Mechanisms for noise-induced transitions,” *The Journal of chemical physics*, vol. 86, no. 6, pp. 3366–3372, 1987.

[52] P. Gaspard, R. Kapral, and G. Nicolis, “Bifurcation phenomena near homoclinic systems: A two-parameter analysis,” *Journal of Statistical Physics*, vol. 35, pp. 697–727, 1984.

[53] M. Hénon, “A two-dimensional mapping with a strange attractor,” *Communications in Mathematical Physics*, vol. 50, no. 1, pp. 69 – 77, 1976.

[54] H. F. von Bremen, F. E. Udwadia, and W. Proskurowski, “An efficient QR based method for the computation of Lyapunov exponents,” *Physica D: Nonlinear Phenomena*, vol. 101, no. 1-2, pp. 1–16, 1997.

[55] V. Dos Santos, J. D. Szezech Jr, M. S. Baptista, A. M. Batista, and I. L. Caldas, “Unstable dimension variability structure in the parameter space of coupled Hénon maps,” *Applied Mathematics and Computation*, vol. 286, pp. 23–28, 2016.

Bregman Proximal Gradient with Extrapolation to Train a Reservoir Computing Network

Carmen Mayora-Cebollero[1], Ana Mayora-Cebollero[1], Álvaro Lozano[2], Roberto Barrio[1]

[1]Departamento de Matemática Aplicada and IUMA. Computational Dynamics group. Universidad de Zaragoza. Spain
[2]Departamento de Matemáticas and IUMA. Computational Dynamics group. Universidad de Zaragoza. Spain

Abstract

This study examines the training process of Artificial Neural Networks, specifically focusing on Reservoir Computing. Training involves optimizing the network's parameters to minimize the error in a loss function, which quantifies the discrepancy between network outputs and target data. Several numerical optimizers are used in the literature. Here, we formulate the supervised learning problem using the recently introduced Bregman Proximal Gradient with extrapolation (BPGe) for non-convex optimization problems. We compare this new method with the classical Stochastic Gradient Descent (SGD), the Root Mean Square Propagation (RMSProp), and the Adaptive Moment estimation (Adam). The new approach leads to an accurate and significantly faster numerical algorithm for solving supervised learning problems on binary classification tasks in Reservoir Computing. Two test examples are presented, one based on chaotic data classification in the classical Lorenz system and the other on the Human Activity Recognition (HAR) using smartphones dataset for movement-rest classification. We show that our approach is highly competitive with existing methods in the tests performed.

1 Introduction

In recent years, Machine Learning, particularly Artificial Neural Networks (ANNs), has become essential for solving diverse problems. A critical aspect of their development is the training process. In this article, we are interested in the problem of supervised learning. Training an ANN is the process of optimizing the trainable parameters, i.e. the weights and biases, to find the lowest error in a suitable function between the outputs and the target data. Several optimizers have been used in the literature [1], like Gradient Descent (GD) [2] and its variants, Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSProp) [3] and the most currently used, Adaptive Moment estimation (Adam) [4]. But there are more families of optimizers in the literature. In particular, methods based on Proximal Gradient algorithms [5, 6] and, more specifically, on Bregman Proximal Gradient have recently been introduced to solve non-convex optimization problems. These methodologies have proven to be quite useful [7, 8] in several practical applications.

In this paper, we develop complete formulas for applying the Bregman Proximal Gradient algorithm with extrapolation (BPGe), introduced in [7], to train Reservoir Computing networks. Reservoir computing (RC) [9, 10] is a special kind of recurrent type Artificial Neural Networks whose main structure is the reservoir. The key point is that only a small fraction of the network is trained, keeping the reservoir fixed. The proposed optimizer expands the available ones, delivering greater accuracy with improved efficiency.

The paper is organized as follows. In Section 2 we introduce all the optimization algorithms. In Section 3 we present the Reservoir Computing architecture. Later, in Section 4 we adapt the Bregman Proximal Gradient with extrapolation algorithm to Reservoir Computing. In Sections 5 and 6, we detail the results of all the optimizers for a dynamical classification problem applied to the classical Lorenz chaotic system and the Human Activity Recognition (HAR) using smartphones dataset, respectively. Finally, in Section 7 we draw some conclusions.

The code of all experiments has been run on a Linux box with 2 Xeon Gold 6338 with 1Tb of DDR4-3200 RAM with an nVidia A40.

2 Minimization algorithms for ANNs

In this section, we review the four optimizers used in this study, first the classical ones and then the Bregman Proximal Gradient algorithm with extrapolation, which will be adapted to ANNs in Section 4.

2.1 Classical optimizers for ANNs

Consider an ANN as a function $\mathcal{N}_{\mathcal{W}}$ transforming any input $x \in \mathbb{R}^n$ into an output $\hat{y} \in \mathbb{R}^m$, that is, $\mathcal{N}_{\mathcal{W}}(x) = \hat{y}$, where $\mathcal{W} = (W, b)$ are the trainable parameters (weights and biases). The goal will be to obtain $\hat{y}$ as close as possible to a target value $y \in \mathbb{R}^m$. To do it, we tune $\mathcal{W}$ using data. Such data, known as training data, consists of N data points (x_j, y_j) for $j = 1, \ldots, N$, where x_j are the inputs and y_j are the corresponding labels (or desired outputs). A loss function $\mathcal{L}_{\mathcal{W}}$ that measures the difference between the target y_j and the network output $\hat{y}_j = \mathcal{N}_{\mathcal{W}}(x_j)$ is defined. Therefore, training process consists in solving the following optimization problem:

$$\min_{\mathcal{W}} \mathcal{L}_{\mathcal{W}} = \min_{\mathcal{W}} \frac{1}{N} \sum_{j=1}^{N} \ell(y_j, \mathcal{N}_{\mathcal{W}}(x_j)), \tag{1}$$

where $\ell(\cdot, \cdot)$ is the loss function applied to each data point.

Gradient Descent (GD) [2] is a classical explicit-gradient optimization algorithm. It relies on moving in the direction of descending gradient to, iteratively, find a minimum (gradient equal to 0). In the field of Deep Learning this algorithm is also known as batch GD as all the training set is used to compute the gradient at each iteration (epoch). As for large training sets the use of the whole dataset for the gradient computation can slow the optimization process, Stochastic Gradient Descent (SGD) was proposed. SGD uses just one sample (chosen randomly from training set) to approximate the gradient at each iteration. Once all the training data points are used, an epoch is completed. This definition matches with the one of Gradient Descent, where in one epoch all points contribute to the parameter update. This algorithm is faster than GD, but less regular because of its stochastic nature. To try to mitigate this effect, mini-batch SGD is applied. In mini-batch SGD, training dataset is divided into subsets of size B (known as batches) and at each iteration a subset is used to compute the gradient until an epoch is reached. That is, the update rule is

$$\mathcal{W}_i \leftarrow \mathcal{W}_{i-1} - \lambda \frac{1}{B} \sum_{j=1}^{B} \nabla_{\mathcal{W}} \ell(y_j, \mathcal{N}_{\mathcal{W}_{i-1}}(x_j)),$$

with λ the learning rate (similar to a step size). Mini-batch SGD reduces to GD when $B = N$ and to SGD when $B = 1$. For simplicity, we will use SGD to refer to mini-batch SGD from now on.

For computational purposes, we consider mini-batch configurations to compute the gradient in the explicit-gradient algorithms. To simplify the notation, we define

$$\nabla_{\mathcal{W}} \mathcal{L}^B_{\mathcal{W}_{i-1}} = B^{-1} \sum_{j=1}^{B} \nabla_{\mathcal{W}} \ell(y_j, \mathcal{N}_{\mathcal{W}_{i-1}}(x_j)).$$

Notice that in these GD-like algorithms, the learning rate λ is constant. An appropriate value for λ is crucial for a good performance of the minimization algorithm: a value that is too high can cause convergence issues, while one that is too low can slow down convergence. Moreover, when working with ANNs, different parameters can have different magnitudes for the gradients. To avoid such problems, algorithms as AdaGrad and RMSProp use adaptive learning rate (it changes over the iterations and the parameters).

Root Mean Square Propagation (RMSProp) [3] is an optimizer created by G. Hinton and T. Tieleman in 2012. It can be considered as a modification of Adaptive Gradient (AdaGrad) [11], with RMSProp performing better in general. It has two steps in each iteration. The first one consists of a moving average (exponential decay) of the gradients (main difference with AdaGrad, which uses a sum of square gradients). The second step is similar to GD-like algorithms but with adaptive learning rate as AdaGrad. This adaptation of the value of the learning rate gives more versatility to the algorithm as it can adapt to the slope of the optimization hypersurface (sharper, plateau, ...). Mathematically, each iteration is represented as

$$\begin{aligned} v_i &\leftarrow \rho v_{i-1} + (1-\rho)(\nabla_{\mathcal{W}} \mathcal{L}^B_{\mathcal{W}_{i-1}} \otimes \nabla_{\mathcal{W}} \mathcal{L}^B_{\mathcal{W}_{i-1}}), \\ \mathcal{W}_i &\leftarrow \mathcal{W}_{i-1} - \lambda \nabla_{\mathcal{W}} \mathcal{L}^B_{\mathcal{W}_{i-1}} / (\epsilon + \sqrt{v_i}), \end{aligned}$$

where ρ is a smoothing constant of the exponential decay and ϵ is a small number used for numerical stability. RMSProp was one of the preferred optimizers until the introduction of Adam.

Adaptive Moment estimation (Adam) [4] is an optimizer related to RMSProp proposed by D.P. Kingma and J. Ba in 2014. It has several advantages such as computational efficiency and requiring minimal tuning of the hyperparameters. Adam is based on the estimation of the first (mean) and second (uncentered variance) raw moments of the gradient. To estimate the first and second moments, Adam computes momentum as an exponentially decaying average of past gradients, and the exponentially decaying average of past squared gradients of RMSProp, respectively. The algorithm is given by:

$$\begin{aligned} m_i &\leftarrow \beta_1 m_{i-1} + (1-\beta_1) \nabla_{\mathcal{W}} \mathcal{L}^B_{\mathcal{W}_{i-1}}, \\ v_i &\leftarrow \beta_2 v_{i-1} + (1-\beta_2)(\nabla_{\mathcal{W}} \mathcal{L}^B_{\mathcal{W}_{i-1}} \otimes \nabla_{\mathcal{W}} \mathcal{L}^B_{\mathcal{W}_{i-1}}), \\ \hat{m}_i &\leftarrow m_i / (1 - (\beta_1)^i), \\ \hat{v}_i &\leftarrow v_i / (1 - (\beta_2)^i), \\ \mathcal{W}_i &\leftarrow \mathcal{W}_{i-1} + \lambda \hat{m}_i / (\epsilon + \sqrt{\hat{v}_i}), \end{aligned}$$

where $\beta_1, \beta_2 \in [0, 1)$, and the steps $\hat{m}_i$ and $\hat{v}_i$ are bias-corrected estimates to counteract the 0 initialization of m_0 and v_0.

2.2 Bregman Proximal Gradient algorithm with extrapolation (BPGe)

Bregman Proximal Gradient algorithm with extrapolation (BPGe) [7] is an implicit-gradient optimizer than can solve non-convex and non-smooth minimization problems. The problem statement for BPGe is the following. The function $\Psi(x)$ to minimize can be expressed as the sum of a non-convex continuously differentiable function $f(x)$ (which possibly is not necessarily globally Lipschitz gradient continuous) and a proper lower-semi-continuous convex function $g(x)$. Then, the algorithm is divided into two stages, the first one in which extrapolation is performed, and the second one, in which a proximal operator is computed. Mathematically,

$$\begin{aligned} y_{i-1} &\leftarrow x_{i-1} + \beta_k (x_{i-1} - x_{i-2}), \\ x_i &\leftarrow \arg\min_x \{ g(x) + \langle \nabla f(y_{i-1}), x - y_{i-1} \rangle + \\ &\qquad\qquad D_h(x, y_{i-1})/\lambda_k \}, \end{aligned} \tag{2}$$

where λ_k is the step size on each iteration (that is, the learning rate), β_k is the extrapolation parameter, and D_h is a Bregman distance [12]. The extrapolation parameter β_k can be determined at each iteration using a line search algorithm [7]. To apply this algorithm, we first fix $\eta \in (0,1)$, $\beta_0 \in [0,1)$ and $\rho \in (0,1)$, and then compute $C_k = 1/(1+\lambda_k \mu)$. The value of parameter μ satisfies that f is μ-weakly convex relative to h (with h the Bregman function that defines de Bregman distance D_h). Then, starting from $\beta_k = \beta_0$, the value of β_k is updated as $\beta_k = \eta \beta_k$ while $D_h(x^k, x^k + \beta_k(x^k - x^{k-1})) > \rho\, C_k\, D_h(x^{k-1}, x^k)$.

This formulation of BPGe generalizes a family of Proximal Gradient algorithms. In particular, if $D_h(x,y) = \|x-y\|^2/2$, BPGe corresponds to Proximal Gradient algorithm with extrapolation (PGe). If moreover $\beta_k = 0$, BPGe reduces to Proximal Gradient (PG) method. Finally, if just $\beta_k = 0$, it is BPG without extrapolation. Theoretical convergence results of BPGe can be found in [7].

2.3 A benchmark minimization test

In this paper, we want to adapt BPGe to an RC for binary classification. To show its performance we will compare with usual optimization algorithms used in ANNs field as (mini-batch) SGD, RMSProp and Adam. But first, we compare them in a benchmark minimization problem.

In Figure 1 we have the evolution of all the optimizers for the benchmark function known as three-hump camel function, that is given by

$$H(x,y) = 2x^2 - 1.05x^4 + x^6/6 + xy + y^2.$$

This function has two local minima and a global minimum at $(0,0)$, so it is an appropriate function to show the performance of the optimization algorithms. For the simulations, we take as initial conditions $(x_0, y_0) = (1.8, 1.8)$ and the learning rate as $\lambda = 0.01$. For SGD, RMSProp and Adam the corresponding PyTorch [13] routines have been used (with all the parameters as default except for the learning rate). For BPGe, the routine has been implemented from scratch with parameters $\beta_{k_0} = 0.99$, $\rho = 0.99$, $\eta = 0.95$ and $\mu = 6$ (following the notation used to introduce BPGe, $f = H$, $g = 0$ and $D_h(x,y) = \|x-y\|^2/2$). As stop rules, we set maximum number of iterations to $4,000$ and a tolerance value to 10^{-15} for the squared Euclidean norm between two consecutive iterations.

Notice that SGD (in black) evolves towards a local minimum, but not to the global one. All the other three algorithms find properly the global minimum, RMSProp (in light blue) and Adam (in purple) with a very similar route, and the BPGe (in red) with a different one and with a much lower number of iterations. As Supplementary material, a video with the evolution of the four optimizers is provided.

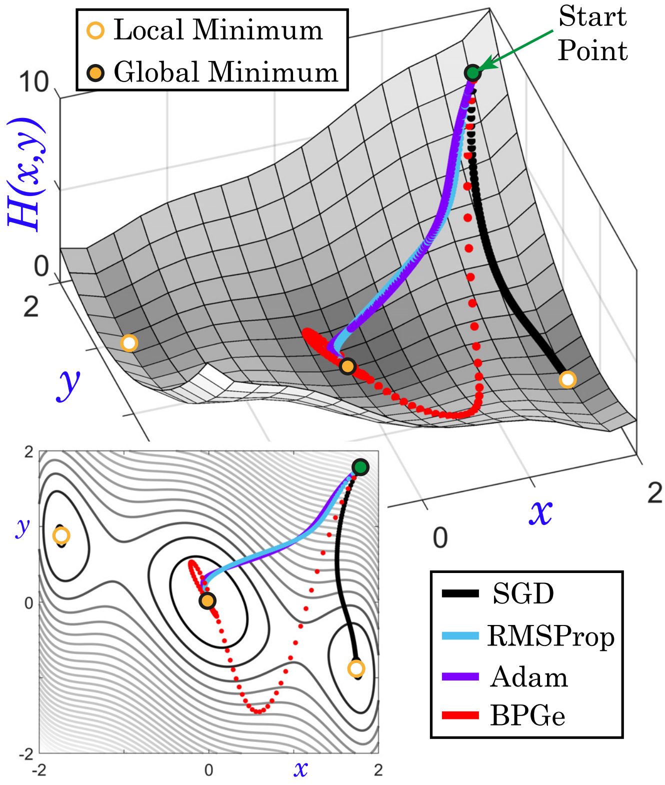


Figure 1: Evolution of SGD (in black), RMSProp (in light blue), Adam (in purple) and BPGe (in red) for the benchmark three-hump camel function.

3 Reservoir Computing

Reservoir Computing (RC) [9, 10] is a special type of recurrent-like Artificial Neural Networks whose main structure is the reservoir. The reservoir is a set of neurons whose connection weights are set randomly and not tuned during training (this simplifies considerably the training problem). RC includes three groups of set ups: Echo State Networks (ESNs) [14], Liquid State Machines (LSMs) [15], and Backpropagation Decorrelation (BPDC) learning rule [16]. In this paper, we focus on the use of ESNs for classification tasks.

Echo-State Networks (ESNs) are a type of Reservoir Computing approach that has been widely and successfully used to different applications [17, 18, 19]. The architecture of an ESN can be divided into three parts: an input layer, the reservoir, and the output layer (also known as readout layer); all of them consisting of artificial neurons. Mathematically, to perform a classification task with K classes, for a time series input $x = \{x_0, x_1, \dots, x_T\}$ (with $x_i \in \mathbb{R}^n$), the output $\hat{y}$ of the ESN (with leaky integrator neurons [20] and a reservoir

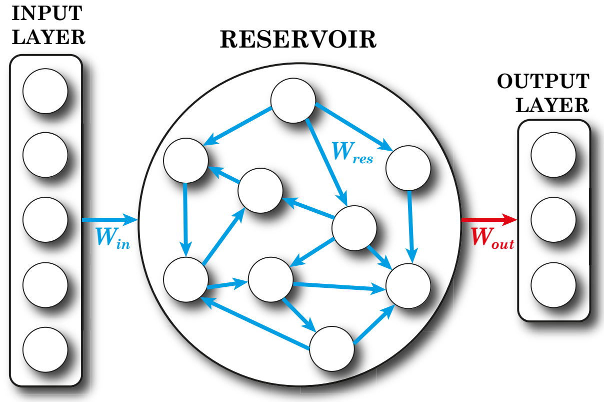


Figure 2: Graphical representation of an ESN, with an input layer of 5 neurons, a reservoir with 9 neurons, and an output layer with 3. In blue we have marked the connections whose weights are fixed (non-trained), and with red these that have to be fine-tuned during training.

with M neurons) is computed as follows:

$$\begin{aligned} h_t &= \alpha \tanh(W_{in}\, x_t + W_{res}\, h_{t-1}) + (1-\alpha) h_{t-1}, \\ \hat{y} &= \operatorname{softmax}(W_{out}\, h_T + b_{out}), \qquad t \in [1, T]. \end{aligned}$$

The first equation corresponds to the input-reservoir and reservoir-reservoir computations, that is, the non-trainable part of the network. The second equation corresponds to the readout layer and, therefore, the trainable part. In the first equation, $h_t \in \mathbb{R}^M$ is the state of the reservoir neurons at time t, $\alpha \in [0,1]$ is the leaking rate parameter (since we use leaky integrator neurons), `tanh` is the hyperbolic tangent function, $W_{in} \in \mathbb{R}^{M\times n}$ is the fixed matrix of weights between the input layer and the reservoir, and $W_{res} \in \mathbb{R}^{M\times M}$ is the fixed weight matrix of the recurrent connections of the reservoir with itself. In the second equation, `softmax` is the function applied to transform the output values into scores for each class, $h_T \in \mathbb{R}^M$ is the last state of the reservoir, $W_{out} \in \mathbb{R}^{K\times M}$ is the trainable weight matrix for the readout layer, and $b_{out} \in \mathbb{R}^K$ is the trainable bias term for such an output layer. In Figure 2 we have a graphical representation of an ESN with $n = 5$, $M = 9$ and $K = 3$ (blue color has been used for connections with non-trainable weights and red color for the trainable ones).

As indicated in [21], some guidelines can be followed when creating fixed matrices W_{in} and W_{res} to obtain a proper ESN. The reservoir matrix W_{res} is preferred to be sparse (each reservoir neuron is connected to a small number of other reservoir neurons) for speed-up purposes. The reservoir should satisfy the Echo State Property [14], that is, the state of the reservoir $h(\cdot)$ should not depend on the initial conditions for a sufficiently long input x. In general, a spectral radius of W_{res} less than 1 ensures that such property holds. In the case of W_{in}, its distribution is taken equal to that of W_{res} but with a dense format. The values of the non-trainable weights can be set using a normal distribution $\mathcal{N}(0, a)$ or a uniform distribution $\mathcal{U}(-a, a)$, with a a scaling parameter.

3.1 Architecture of the ESN for optimizers comparison

To ensure a fair comparison analysis of the performance of the four optimizers, we need to fix the structure and hyperparameters of the ANN architecture. For the reservoir, we consider a population size (number of neurons in the reservoir) $M = 500$ for Lorenz system analysis, and $M = 100$ for the Human Activity Recognition using smartphones dataset using smartphones dataset. The connection matrix between the reservoir neurons with themselves has been built as an oriented graph with no loops whose probability of connection is 0.5. For the weight matrix of the reservoir, i.e. W_{res}, we fix the weights sampling from a normal distribution $\mathcal{N}(0, 3.5)$ and such matrix is modified later in order to have spectral radius 0.9 (to ensure the echo-state property). The connection matrix between the input and reservoir neurons is non-sparse with the weights (W_{in}) sampled from a Gaussian distribution $\mathcal{N}(0, 3.5)$. The leaking rate α is set to 0.6 for Lorenz system analysis, and to 0.1 for the Human Activity Recognition using smartphones dataset. Moreover, we fix the initial values of the trainable parameters in the default values provided by PyTorch [13]. The initial value of the state (h_0) is set to 0 following the default initialization of PyTorch [13] for the states of Recurrent-like Neural Networks. The final time T corresponds to the duration of the time series and will be set for each test example. The early stopping technique [2] is used, that is, the final value of the trainable parameters is that giving the lowest loss value for the validation data set during the training process. The number of epochs is set to $4,000$. The loss function is the Cross-Entropy loss with weight decay (for L^2-penalty) equal to 10^{-5}. We remark that the hyperparameters have not been fine-tuned with the objective of not particularizing them to any optimizer and to try to obtain a fair comparison among all algorithms.

4 BPGe for Artificial Neural Networks

In this section we develop all the necessary formalism to apply BPGe to train an Echo-State Network for a binary classification.

First, note that when training a network, the problem to minimize is the one given in Equation (1). Therefore, in this case the variables with respect to which we minimize are the trainable parameters W_{out} and b_{out} with $W_{out} \in \mathbb{R}^{2\times M}$ and $b_{out} \in \mathbb{R}^2$. Let us use $\mathcal{W}$ to refer to $(W_{out}, b_{out}) \in \mathbb{R}^{2M+2}$. The function to minimize is the loss function. For a binary classification task, the usual loss function is the Cross-Entropy Loss. Moreover, we use L^2-penalty as a regularization technique to prevent overfitting. Then, using the formulation given in Subection 2.2 for BPGe algorithm, we take as f the Cross-Entropy Loss (CEL) and as g the L^2-penalty. That is,

$$f(\mathcal{W}) = \mathrm{CEL}(\mathcal{W}) = -\sum_{i=1}^{N} \log\left(\frac{\mathrm{e}^{v^i_{\{l_i\}}}}{\sum_{k=1}^{K} \mathrm{e}^{v^i_{\{k\}}}} \right),$$

where $v^i_{\{k\}} = \sum_{m=1}^{M} w_m^{\{k\}} R_m + b^{\{k\}}$, with $w_m^{\{k\}}$ the m-th weight used to compute class k, R_m the value of the neuron of the reservoir multiplied by weight $w_m^{\{k\}}$, $b^{\{k\}}$ the bias for class k, $\{l_i\}$ the correct class (label) for sample i, N the number of samples in the batch, K the number of classes, and M the number of neurons of the reservoir. Besides,

$$g(\mathcal{W}) = L^2(\mathcal{W}) = \eta \sum_{j=1}^{2M+2} w_j^2,$$

where η is the regularization parameter (for L^2- penalty) and w_j is any trainable parameter (weight or bias). Notice that these functions satisfy the required properties of the algorithm statement (see Subsection 2.2): f is convex (by composition of convex functions) and continuously differentiable (by composition); g is proper, lower semi-continuous (since it is continuous) and convex (by composition).

Notice that to apply the BPGe algorithm (see Equation (2)) we need a Bregman distance D_h that has to be defined by a Bregman function h. For this problem we set $h(\mathbf{x}) = \|\mathbf{x}\|^2/2$, getting the Bregman distance $D_h(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2/2$. With these functions the conditions required in [7] are satisfied. In particular, f is μ-weakly convex relative to h with $\mu = 0$ as f is convex.

Let us focus on adapting the algorithm to the binary classification task with Reservoir Computing. In particular, let us find the analytical expression for the second step of the algorithm (see Equation (2)), that with the new notation is rewritten as

$$\mathcal{W}^{k+1} = \arg\min_{\mathcal{W}} \{g(\mathcal{W}) + \langle \nabla f(\mathcal{V}^k), \mathcal{W} - \mathcal{V}^k \rangle \\ + D_h(\mathcal{W}, \mathcal{V}^k)/\lambda_k\}.$$

To obtain $\mathcal{W}^{k+1}$ we have to compute the expression inside the curly brackets, calculate its derivative (the gradient), obtain the value of $\mathcal{W}$ that cancels all the elements of the gradient vector, and check that the extreme is a minimum. By the moment we will not compute the expression of $\nabla f(\mathcal{V}^k)$ as it is a constant and does not affect directly the computations related to the minimum. We will compute its expression later as it is the key point of the adaptation of the algorithm to Reservoir Computing for binary classification.

The expression inside the curly brackets is

$$\{g(\mathcal{W}) + \langle \nabla f(\mathcal{V}^k), \mathcal{W} - \mathcal{V}^k \rangle + D_h(\mathcal{W}, \mathcal{V}^k)/\lambda_k\} = \\ \eta \sum_{j=1}^{2M+2} w_j^2 + \sum_{j=1}^{2M+2} (\nabla f(\mathcal{V}^k))_j (w_j - v_j^k) + \frac{1}{2\lambda_k} \|\mathcal{W} - \mathcal{V}^k\|^2.$$

The s-th component of the gradient of this expression is

$$2\eta w_s + (\nabla f(\mathcal{V}^k))_s + \frac{1}{\lambda_k}(w_s - v_s^k).$$

The value of $\mathcal{W}$ that cancels previous expression is the vector whose s-th element is

$$\frac{-(\nabla f(\mathcal{V}^k))_s + \frac{1}{\lambda_k} v_s^k}{2\eta + \frac{1}{\lambda_k}}. \tag{3}$$

We just have to check that it is a minimum. The Hessian matrix is a diagonal matrix with diagonal elements equal to $2\eta + 1/\lambda_k$, so the determinant is $(2\eta + 1/\lambda_k)^{2M+2} > 0$ ($\eta > 0$ is the regularization parameter, and $\lambda_k > 0$ is the step size, so they are greater than 0). Moreover, the first diagonal element of the matrix is $2\eta + 1/\lambda_k$ which is also positive ($\eta > 0,\ \lambda_k > 0$). Therefore, the point $\mathcal{W}$ whose s-th coordinate is given in Equation (3) is a minimum and the value $\mathcal{W}^{k+1}$ we are looking for. We just have to compute $\nabla f(\mathcal{V}^k)$ to have the expression totally defined.

Let us compute the expression of $\nabla f(\mathcal{W})$. Each element of the gradient is $\partial f / \partial w_s$ where w_s is the s-th element of $\mathcal{W} = (W, b)$. The function f is a sum of B elements, each summand is related to a sample of the batch. Each of these samples has a label. Note that according to the form of each summand, it is not the same to compute the derivative with respect to one trainable parameter involved in the computation of the value of the neuron of class l_i if the corresponding label is l_i (this weight/bias appears in the numerator and denominator of f) or it is not (this weight/bias only appears in the denominator). Therefore, we have to distinguish four possible cases:

(C1) We derive with respect to an element of $\mathcal{W}$ that corresponds to a weight involved in the computation of the output neuron corresponding to the correct class l_i (given by the label) for sample i.

(C2) We derive with respect to an element of $\mathcal{W}$ that corresponds to a weight not involved in the computation of the output neuron corresponding to the correct class l_i (given by the label) for sample i.

(C3) We derive with respect to an element of $\mathcal{W}$ that corresponds to the bias involved in the computation of the output neuron corresponding to the correct class l_i (given by the label) for sample i.

(C4) We derive with respect to an element of $\mathcal{W}$ that corresponds to the bias not involved in the computation of the output neuron corresponding to the correct class l_i (given by the label) for sample i.

For the sake of notation, in what follows we set $\mathtt{RC}^{\{l_i\}} = \exp\left(\sum_{m=1}^{M} w_m^{\{l_i\}} R_m + b^{\{l_i\}}\right)$, that is, the exponential value of the raw value (before `softmax` application) of a neuron in the output layer. With this notation, we can rewrite each of the addends of f as

$$\log\left(\frac{\mathtt{RC}^{\{l_i\}}}{\mathtt{RC}^{\{0\}} + \mathtt{RC}^{\{1\}}}\right) = \log\left(\frac{\mathtt{NUM}}{\mathtt{DEN}}\right),$$

where 0 and 1 are the possible classes and $\{l_i\} \in \{0, 1\}$, taking the value according to the class labeled to the sample. We will use $\{l_i\}^c$ for the class contrary to the label (non-correct class).

We have to compute the partial derivative of each of those addends respect to an element w_s of $\mathcal{W}$, that will have different expressions for each of the four cases, obtaining

$$\text{(C1)}\qquad \frac{\mathtt{DEN}}{\mathtt{NUM}}\,\frac{R_s\,\mathtt{NUM}\,\mathtt{DEN}-R_s\,\mathtt{NUM}^2}{\mathtt{DEN}^2}=R_s\,\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}}.$$

$$\text{(C2)}\qquad \frac{\mathtt{DEN}}{\mathtt{NUM}}\,\frac{(-\mathtt{NUM})\,R_s\,\mathtt{RC}^{\{l_i\}^c}}{\mathtt{DEN}^2}=-R_s\,\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}}.$$

$$\text{(C3)}\qquad \frac{\mathtt{DEN}}{\mathtt{NUM}}\,\frac{\mathtt{NUM}\,\mathtt{DEN}-\mathtt{NUM}^2}{\mathtt{DEN}^2}=\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}}.$$

$$\text{(C4)}\qquad \frac{\mathtt{DEN}}{\mathtt{NUM}}\,\frac{(-\mathtt{NUM})\,\mathtt{RC}^{\{l_i\}^c}}{\mathtt{DEN}^2}=-\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}}.$$

If we consider that the first M elements of $\nabla f(\mathcal{W})$ correspond to the derivative with respect to a weight w_s used to compute the value of the neuron of class 0, the next M elements are the derivative with respect to a weight w_s used to compute the value of the neuron of class 1. The penultimate derivative is with respect to the bias $b^{\{0\}}$ and the last one is with respect to the bias $b^{\{1\}}$. Therefore, the gradient is

$$\begin{pmatrix} \sum_{i=1}^{B}(-1)^{\{l_i\}}R_1\,\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}}\\ \vdots\\ \sum_{i=1}^{B}(-1)^{\{l_i\}}R_M\,\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}}\\ \sum_{i=1}^{B}(-1)^{\{l_i\}+1}R_1\,\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}}\\ \vdots\\ \sum_{i=1}^{B}(-1)^{\{l_i\}+1}R_M\,\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}}\\ \sum_{i=1}^{B}(-1)^{\{l_i\}}\,\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}}\\ \sum_{i=1}^{B}(-1)^{\{l_i\}+1}\,\frac{\mathtt{RC}^{\{l_i\}^c}}{\mathtt{RC}^{\{0\}}+\mathtt{RC}^{\{1\}}} \end{pmatrix}.$$

Finally, $\nabla f(\mathcal{V}^k)$ will be a constant vector obtained substituting $\mathcal{W}$ by $\mathcal{V}^k$ in the previous expression.

With this we have provided the complete set of formulas for applying the BPGe algorithm to the binary classification problem in RC.

5 Dynamical classification in the Lorenz System

The first problem used to study the performance of the four optimizers is the Lorenz system [22], a paradigmatic problem of chaotic dynamics. Lorenz system is a classic 3-dimensional continuous dynamical system problem given by the following equations:

$$\dot{x}=\sigma(y-x),\quad \dot{y}=x(r-z)-y,\quad \dot{z}=xy-bz,\qquad (4)$$

where the system variables are (x, y, z) and the bifurcation parameters are (σ, r, b). In particular, σ is known as Prandtl number, r is the relative Rayleigh number and b is a positive constant. In our study, $\sigma = 10$, $r \in (0, 300]$ and $b \in \{2.4, 2.8, 8/3\}$. Our goal is to classify the dynamics of this system into regular (label 0) or chaotic (label 1). To compare the results obtained with our neural network to the correct behaviour, we compute the Lyapunov exponents (positive first Lyapunov exponent means chaos, and regular otherwise) using the algorithm of Ref. [23].

To obtain the train, validation and test datasets needed to achieve the final trained network for the Lorenz system case, we use three r-parametric lines where parameter σ is fixed to 10, and $b = 2.4$ for the training dataset, $b = 2.8$ for the validation dataset, and $b = 8/3$ (classical line of the Lorenz system) for the test dataset. First, from each r-parametric line, we get $5,999$ different values of r moving uniformly in the interval $(0, 300]$ and thus we have $5,999$ time series for each dataset. These time series are of length $1,000$ and their fixed initial conditions are $(x_0, y_0, z_0) = (1, 1, 1)$. To integrate these time series the `DOPRI5` method (Runge-Kutta integrator of order 5) is used. First of all, a transient process is performed until time $t = 100,000$ with time step 0.01. Later, we integrate $100,001$ more time units with a time step of 0.001 to compute the Lyapunov exponents (used to determine the behaviour of each time series). Taking 1 out of every 100 of the last $100,000$ computed points we obtain the time series of length $1,000$ that we use as input in our neural network.

The samples obtained from the three r-parametric lines are screened to prevent repeated time series inside or through different datasets (and therefore, to ensure that the network learns correctly). First of all, we delete such repeated samples, considering repeated samples as those whose difference in infinity norm is less than 10^{-4}. Next, we shuffle the samples inside each dataset and we separate time series between regular and chaotic. From these remaining samples, we take randomly $2,260$ of each class for the training dataset ($5,520$ in total), $1,500$ of each class for the validation dataset ($3,000$ in total) and $2,990$ of each class for test set ($5,980$ in total). The batch sizes are fixed to 128, 100, and 230 for training, validation, and test sets, respectively, dropping the last incomplete batch. In the case of the test set, we will have two different versions of the dataset. One of them without repeated samples that we will use to study the loss and accuracy (to ensure the network is able to generalize correctly), and other one with all the samples of the line that we will use for the remaining experiments (in this last case, the number of samples for each class does not have to be the same one).

In Table 1, the loss and accuracy values for the final network in the train, validation and test datasets using the SGD, RMSProp, Adam and BPGe optimizers are gathered together. For each optimizer the results are shown for three different values of the learning rate λ: $\lambda_1 = 0.001$, $\lambda_2 = 0.0005$ and $\lambda_3 = 0.0001$. We have studied the accuracy of the test dataset, but we have to ensure that the network has learned correctly, that is, if it is able to classify correctly each type of behavior (regular and chaotic). In the last two columns of this table, we study the test accuracy for the regular and chaotic samples (that is, the regular samples that are correctly detected respect to the total number of regular samples, and analogously for the chaotic time series).

For the SGD and Adam optimizers, the loss increases and accuracy decreases as the learning rate decreases. Hence, the best results (lowest test loss and highest test accuracy) occur with $\lambda_1 = 0.001$. However, in the remaining two optimizers (RMSProp and BPGe), there is no clear trend in behavior. In fact, for RMSProp, in the training set, loss value increases and accuracy value decreases as the learning rate decreases, while this is not occurring for the other two datasets where the best values (lowest loss and largest accuracy) are given for $\lambda_2 = 0.0005$. In the case of BPGe, it is not easy to define a trend, but probably the appropriate value for the learning rate is $\lambda_3 = 0.0001$.

Notice that the accuracy values for both behaviors are quite similar in all cases (the difference between them is always less than 8%). Therefore, all the optimizers (for all the learning rate values) are able to detect properly regular and chaotic samples.

Comparing the results in the table across all the optimizers we can conclude that the worst results are obtained with the SGD optimizer as for the test dataset the accuracy is for the three cases less than 92% and for the other optimizers it is always greater than 93% (for all the learning rate values). Moreover, the SGD optimizer gives the largest values for the loss. The largest accuracy values for all the studied learning rates in the test dataset are those obtained with the BPGe optimizer (comparing the value of the test accuracy for each value of the learning rate). However, if we compare the accuracy for regular and chaotic samples for the test dataset, we can see that the performance of Adam and BPGe is quite similar (for example, for $\lambda_2 = 0.0005$, the accuracy for regular samples is larger for BPGe than for Adam, but the accuracy for chaotic behaviour is greater for Adam). Notice that the lowest loss value for the test dataset is obtained with $\lambda_3 = 0.0001$ for the BPGe optimizer, which corresponds with the largest test accuracy for all the optimizers and learning rates.

After studying the loss and accuracy for the best epoch (the epoch corresponding with the lowest loss value of the validation dataset, that according to early stopping provides the final network), we study the evolution of the loss and accuracy values along the epochs (for training dataset) in Figure 3. In panels (A1)-(A2), (B1)-(B2) and (C1)-(C2), these evolutions are shown for the learning rates $\lambda_1 = 0.001$, $\lambda_2 = 0.0005$ and $\lambda_3 = 0.0001$, respectively. In each plot, the four optimizers are compared (SGD in black, RMSProp in light blue, Adam in purple, and BPGe in red).

Notice that the BPGe optimizer (in red) is the noisiest optimizer for $\lambda_1 = 0.001$ (see panels (A1)-(A2)). If the value of the learning rate is decreased up to $\lambda_2 = 0.0005$, only a small part of the evolution is noisy (see panels (B1)-(B2)). When the value of the learning rate is very small ($\lambda_3 = 0.0001$), the evolution of the loss and accuracy of the BPGe is not noisy anymore (this learning rate gives us the best test accuracy as can be seen in Table 1). This could be related with the restriction $0 < \lambda_k \leq 1/L$ (for the BPGe) indicated in [7], with L the constant of the Lipschitz gradient continuity condition. We conjecture that for values of the learning rate smaller than $\lambda_2 = 0.0005$ (and not for values greater or equal than λ_1) this condition is fulfilled and therefore the noise disappears.

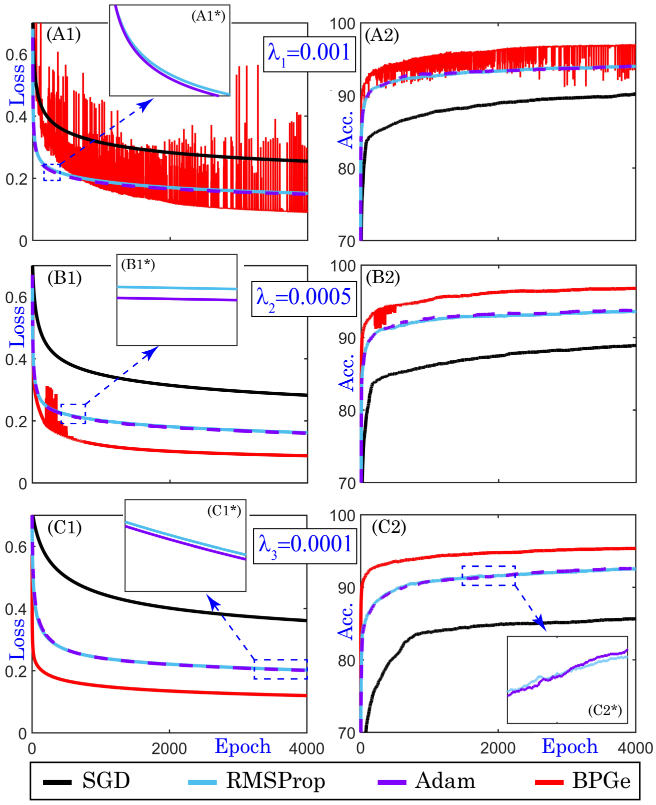


Figure 3: Study of the loss and accuracy (Acc.) evolution of the training dataset along the epochs for the Lorenz system dynamical classification. The four optimizers (SGD, RMSProp, Adam and BPGe) are compared. (A1)-(A1*)-(A2) Evolution for the learning rate $\lambda_1 = 0.001$. (B1)-(B1*)-(B2) Evolution for the learning rate $\lambda_2 = 0.0005$. (C1)-(C1*)-(C2)-(C2*) Evolution for the learning rate $\lambda_3 = 0.0001$.

For all the learning rate values, it can be seen clearly that the BPGe optimizer (in red) achieves the lowest loss values and the largest accuracy with a considerable difference. For the SGD optimizer (in black), the loss seems to be always larger and the accuracy lower than the other three optimizers. The results obtained with the RMSProp (in light blue) and Adam (in purple) are quite similar and indistinguishable at a glance, so we provide some magnifications. A zoom of panels (A1)-(B1)-(C1) is provided in subpanels (A1*)-(B1*)-(C1*), where it can be seen that loss values of both optimizers are really close but larger for the RMSProp algorithm. The accuracy for RMSProp and Adam optimizers is quite similar: some-

Table 1: Loss and accuracy (Acc.) for training, validation and test sets for the four optimizers and three different learning rate values ($\lambda_1 = 0.001$, $\lambda_2 = 0.0005$, $\lambda_3 = 0.0001$) for the dynamical classification in the Lorenz system. Accuracy regular (Acc. Reg.) and accuracy chaotic (Acc. Chaot.) are also indicated for test dataset.

		TRAIN		VALIDATION		TEST			
		Loss	Acc.(%)	Loss	Acc.(%)	Loss	Acc.(%)	Acc. Reg. (%)	Acc. Chaot. (%)
SGD	λ_1=0.001	0.255	90.201	0.225	92.533	0.242	91.656	92.319	91.111
	λ_2=0.0005	0.283	88.929	0.250	91.567	0.266	90.535	91.280	89.924
	λ_3=0.0001	0.361	85.670	0.329	88.500	0.341	87.475	88.794	86.393
RMSProp	λ_1=0.001	0.152	94.018	0.165	94.067	0.186	93.813	97.737	90.594
	λ_2=0.0005	0.162	93.594	0.162	94.300	0.182	94.013	96.994	91.568
	λ_3=0.0001	0.202	92.522	0.183	93.833	0.202	93.545	94.397	92.846
Adam	λ_1=0.001	0.149	94.018	0.135	94.800	0.148	94.716	96.067	93.607
	λ_2=0.0005	0.160	93.705	0.143	94.667	0.158	94.632	95.176	94.186
	λ_3=0.0001	0.201	92.612	0.180	93.633	0.199	93.395	93.729	93.120
BPGe	λ_1=0.001	0.108	96.652	0.178	95.433	0.212	94.900	97.069	93.120
	λ_2=0.0005	0.088	96.763	0.139	94.867	0.160	94.682	97.922	92.024
	λ_3=0.0001	0.120	95.357	0.127	95.067	0.139	94.916	97.217	93.029

times one is larger than the other and other times the opposite as can be seen in subpanel (C2*) which is a zoom of panel (C2).

To sum up, on the one hand, the best performance, that is, the lowest loss and the largest accuracy, is obtained with the BPGe optimizer (see the red line in all panels of Figure 3). In particular, the best results are the ones corresponding to $\lambda_3 = 0.0001$ (see Table 1). On the other hand, the worst performance, that is, the largest loss and the lowest accuracy, is achieved with the SGD optimizer (see the black line in all panels of Figure 3 and Table 1). The RMSProp and Adam optimizers seem to give very similar results (better than those of SGD and worst than those of BPGe optimizer). The conclusions obtained with Figure 3 coincide with those derived from the study of Table 1.

With results in Table 1 and Figure 3, we conclude that BPGe seems to work really good. However, it is also interesting to analyze in which epoch is obtained the network with the best results (lower validation loss during training process) and how much time is needed by each optimizer to train the network. Sometimes better results may not be entirely advantageous if the time needed is much longer.

For the SGD, RMSProp, and Adam optimizers, the best results occur at epoch 4,000, which is the maximum set for the process. This suggests further learning could occur with more epochs, but increased training time may not yield significant accuracy improvements, making additional epochs less worthwhile. With the maximum of $4,000$ epochs, the wall time during training process for each of the three optimizers is approximately 27 minutes. Focusing in the BPGe results, for $\lambda_1 = 0.001$ the best epoch is $3,065$, for $\lambda_2 = 0.0005$ it is $3,893$, and for $\lambda_3 = 0.0001$ it is $4,000$ (last epoch as happens for the other optimizers). With the maximum of $4,000$ epochs, the wall time during training process for BPGe optimizer is around 30 minutes. Notice that BPGe takes a bit longer to train the network, but for lower learning rates in fact a lower maximum of epochs value and consequently less wall time would be needed. For the largest learning rate value in BPGe, the best epoch is $4,000$, but although time is a bit increased respect to other optimizers, results are better.

We want to remark that in the wall time is only included the time needed to train the network once h_T has been obtained for all the samples.

The total number of epochs is a hyperparameter that could have been set to a different value. A natural question arises: At which epoch has the training accuracy surpassed a certain value? As in general we have obtained that the networks achieved the best model for the last epoch, this new analysis will allow us to understand which optimizers provide a faster learning and need less epochs to obtain a good performance. Here, we have checked that in addition to satisfy the accuracy indicated, the validation accuracy is not far away from this value and then it is not suffering from overfitting. In Table 2, we have summarized such analysis. In particular, we indicate, for each optimizer and learning rate, the first epoch in which training accuracy has surpassed the 80%, 85%, 90%, and 95%. Notice that the unique optimizer that surpasses the 95% accuracy for training is BPGe. The SGD algorithm only achieves an accuracy greater than 90% for the lower value of learning rate ($\lambda_1 = 0.001$) that we established as the best result for this optimizer according to Table 1. Moreover, the number of epochs needed by SGD to exceed whatever accuracy value is considerably larger than that for the other optimizers. The RMSProp and Adam optimizers have quite similar results as we have already concluded analyzing the training loss and training accuracy evolutions in Figure 3. Finally, it is remarkable that the BPGe, for whatever value of the learning rate, surpasses a value of 90% in the accuracy just after less than 20 epochs. Therefore, although BPGe needs more time for training process (30 min against the 27 minutes needed by the other optimizers), it uses a small portion of time to achieve a good performance (20 of the 4000 epochs, that would be approximately 10 seconds) and the remaining process is devoted to fine-tune the results to obtain a better performance. So, setting a lower number of epochs, wall time would be considerably reduced and performance would be not considerably affected. To try to support this result, we are going to check the performance of the network obtained at each epoch using an r-parametric line of the Lorenz system.

Table 2: First epoch in the training process of the dynamical classification in the Lorenz system in which the accuracy of the training set is equal to or greater than 80%, 85%, 90% and 95% for the four optimizers and the considered learning rates.

Accuracy (%)		$\geq 80\%$	$\geq 85\%$	$\geq 90\%$	$\geq 95\%$
SGD	λ_1=0.001	44	245	3750	–
	λ_2=0.0005	89	461	–	–
	λ_3=0.0001	445	2248	–	–
RMSProp	λ_1=0.001	3	13	101	–
	λ_2=0.0005	4	15	145	–
	λ_3=0.0001	12	60	578	–
Adam	λ_1=0.001	3	11	103	–
	λ_2=0.0005	4	17	154	–
	λ_3=0.0001	16	62	638	–
BPGe	λ_1=0.001	2	4	17	712
	λ_2=0.0005	2	3	14	877
	λ_3=0.0001	2	3	17	2501

Let us consider the r-parametric line with $b = 8/3$ and $\sigma = 10$ of the Lorenz system. Notice that it corresponds to the line used to create the test set, but now we use it with the $5,999$ values of parameter r. During training, we save the values of the trainable parameters (W_{out} and b_{out}) of the network obtained at each epoch. Note that as we have set to $4,000$ the maximum number of epochs, we will have $4,000$ different networks for each optimizer and each learning rate. With each network, we perform a dynamical classification in the aforementioned r-parametric line. This will allow us to have an idea of how the results would be at each moment of the training process. In Figure 4, we have such results. In particular, in panels (A1)-(A3), (B1)-(B3), (C1)-(C3) and (D1)-(D3) we have on the top the classification results obtained with the first epoch of the training process and at the bottom those provided by the last epoch (epoch $4,000$), and the results of the networks for the remaining epochs are in between. Moreover, for each optimizer and learning rate we provide the dynamical classification performed for the best epoch (see panels (A1*)-(A3*), (B1*)-(B3*), (C1*)-(C3*) and (D1*)-(D3*). The ground truth, that is, the correct classification performed by the classical technique of Lyapunov exponents is given at the top of the figure. Notice that black color is used for regular behavior and white for chaotic.

In panels of SGD (A1)-(A3), we can observe a noisy classification for any epoch (especially in panel (A3) that corresponds with the worst results according to previous analyses). In panels of RMSProp and Adam optimizers (B1)-(B3) and (C1)-(C3), respectively, we have less noisy classification in non-boundary regions, but it seems that they are more indecisive than BPGe (panels (D1)-(D3)). In BPGe panels (D1)-(D3), it is observed that in the first epochs the optimizer already correctly determines the non-boundary zones and the regular windows in the large chaotic regions; and it uses the remaining epochs to fix the behavior in the boundaries. These conclusions match with the results of Table 2 as an accuracy greater than 90% (for training set) is obtained after just 20 epochs.

If we look at the panels (A1*)-(A3*), (B1*)-(B3*), (C1*)-(C3*) and (D1*)-(D3*), we can conclude visually the best classification. It can be seen clearly that the worst performance is achieved by SGD (panels (A1*)-(A3*)) as in the left regular region (in black in the ground truth), we can observe white stripes corresponding to false chaotic detections. Moreover, we can see that the boundary zones (in the middle of the panel) are not properly defined. In the remaining panels (of RMSProp, Adam and BPGe) the results seem to be acceptable and the regions are better defined. However, in the BPGe panels (D1*)-(D3*) more regular windows in the large chaotic regions have been clearly detected.

We would like to clarify, that if we compare the best epoch results for all the optimizers and learning rates with the ground truth, one part of the most-right chaotic region is missing. This part corresponds to transient chaotic dynamics [24]. We have been seen in a previous work [25] that they are not properly detected by all DL architectures. However, this is out of the scope of this paper as the objective is to compare optimizers under the same conditions.

6 Classification in the Human Activity Recognition using Smartphones Dataset

Human Activity Recognition (HAR) using smartphones dataset [26, 27] consists in recordings from 30 subjects (aged between 19 and 48 years) performing six different Activities of Daily Living (ADL): walking, walking upstairs, walking downstairs, sitting, standing, and laying down. The 3-axial linear acceleration and 3-axial angular velocity are recorded using a waist-mounted smartphone with inertial sensors at a constant rate of 50Hz. The data obtained from the experiments is preprocessed and for each record, 3-axial total acceleration, 3-axial estimated body acceleration, 3-axial angular velocity and a vector with 561 features are given. The experimental data [26] is provided as training (70% of total samples) and test set (30%).

We work with binary classification tasks with time se-

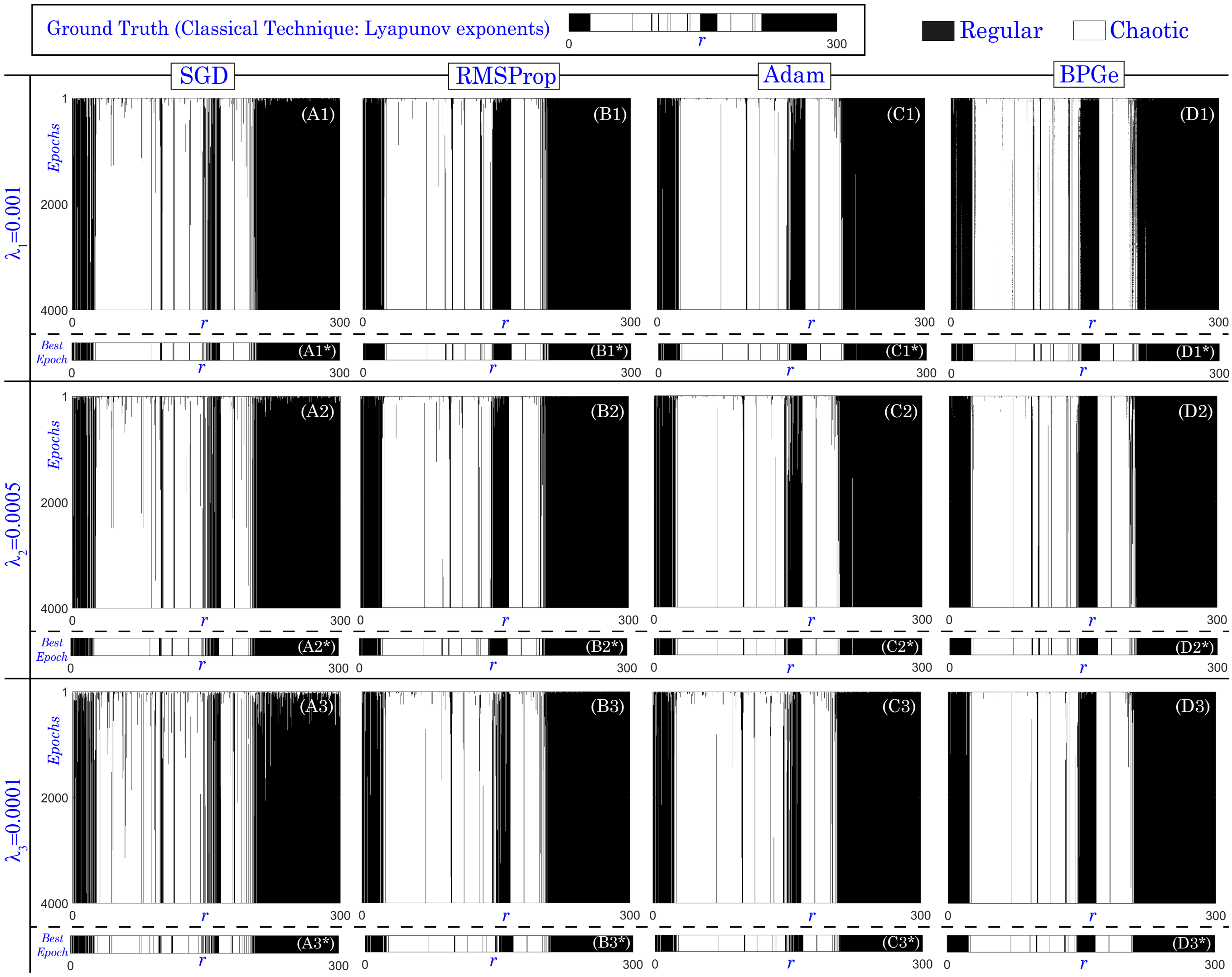


Figure 4: Results of dynamical classification in an r-parametric line of the Lorenz system obtained with each of the networks of the $4,000$ epochs obtained during training process are shown in order to study visually how the optimizers work.

ries as input. Therefore, from the HAR using smartphones dataset we consider the 3-axial estimated body acceleration, measured as a g-force, consisting in 3-dimensional time series of length 128. Moreover, we divide into two classes: rest (it includes sitting, standing and laying down) and movement (it includes walking, walking upstairs and walking downstairs). Instead of using just two datasets (train and test) as provided by this HAR dataset, we use three (train, validation and test). For this reason, we use the original test set of [26] without modifications (it includes 2947 time series, 1560 labeled as rest and 1387 as movement), but we use the original train set provided in [26] to create a train and validation set for our study. In particular, the original training set has 7352 samples, 4067 corresponds to rest class (1286 of sitting, 1374 of standing, and 1407 of laying down), and 3285 to movement (1226 of walking, 1073 of walking upstairs, and 986 of walking downstairs). We choose randomly 686 samples of each activity for training, and 300 for validation; obtaining a training dataset with 4116 time series, and a validation set of 1800. For training, validation and test sets the batch size is 128, 100, and 421, respectively.

For this new example, we perform a similar analysis to that of Lorenz system in Section 5.

In Table 3, the results of the loss and accuracy of the training, validation and test datasets are shown for the four optimizers (SGD, RMSProp, Adam and BPGe) using three different learning rate values ($\lambda_1 = 0.001$, $\lambda_2 = 0.0005$ and $\lambda_3 = 0.0001$). As for the Lorenz case, we have also computed the accuracy for each class to ensure that the network has learned to classify both type of samples (rest and movement).

Notice that the loss of the training set and the one of the test (the same for the accuracy) are very similar, therefore, we can consider that the network has not suffered overfitting for any case. It is clear that the worst optimizer is the SGD (as it occurs for the Lorenz case):

Table 3: Loss and accuracy (Acc.) for training, validation and test sets for the four optimizers and three different learning rate values ($\lambda_1 = 0.001$, $\lambda_2 = 0.0005$, $\lambda_3 = 0.0001$) for the classification between movement and rest of the HAR using smartphones database. Accuracy of rest and accuracy of movement are also indicated for test dataset.

		TRAIN		VALIDATION		TEST			
		Loss	Acc.(%)	Loss	Acc.(%)	Loss	Acc.(%)	Acc. Rest (%)	Acc. Move (%)
SGD	λ_1=0.001	0.453	84.961	0.458	84.833	0.452	85.104	91.282	78.154
	λ_2=0.0005	0.505	82.959	0.510	82.778	0.501	82.966	90.513	74.477
	λ_3=0.0001	0.595	77.808	0.598	76.944	0.596	77.537	89.231	64.384
RMSProp	λ_1=0.001	0.157	95.972	0.177	94.889	0.194	94.605	99.231	89.402
	λ_2=0.0005	0.181	95.020	0.196	94.389	0.212	94.231	98.846	89.041
	λ_3=0.0001	0.270	91.553	0.276	91.500	0.287	90.567	95.192	85.364
Adam	λ_1=0.001	0.156	95.947	0.176	94.889	0.193	94.639	99.231	89.474
	λ_2=0.0005	0.180	95.044	0.195	94.389	0.212	94.265	98.846	89.113
	λ_3=0.0001	0.270	91.577	0.276	91.444	0.287	90.601	95.192	85.436
BPGe	λ_1=0.001	0.108	97.266	0.147	96.500	0.169	95.419	99.294	91.060
	λ_2=0.0005	0.114	97.070	0.148	96.278	0.169	95.385	99.359	90.916
	λ_3=0.0001	0.150	96.069	0.172	94.944	0.188	94.774	99.295	89.690

it has the largest loss and the lowest accuracy for all datasets and independently of the value of the learning rate. Moreover, it is remarkable that the accuracy in the test dataset for rest and movement classes is very different. It seems that the characteristics of movement class have not been properly learned and the corresponding accuracy is really low (especially for λ_3 as it is smaller than 65%). RMSProp and Adam optimizers give similar results. Both performing quite well, with worsening results as the value of the learning rate is reduced. The BPGe is again the optimizer that gives the best results. It is remarkable that it provides the lowest loss and largest accuracy values for all learning rate values and datasets. Moreover, it is the unique optimizer in which an accuracy greater than 90% is obtained for the movement class (see results for λ_1 and λ_2). The learning rate that seems to perform the best is $\lambda_1 = 0.001$.

In Table 4, the epoch in which the accuracy of the training dataset (in the training process) is greater or equal than 80%, 85%, 90% and 95% for the first time is indicated for the four optimizers and the three learning rate values. Notice that in the case of the SGD most of the boxes of the tables are filled with "–" as the network does not achieve an accuracy equal to or greater than 85% for $\lambda_1 = 0.001$ and $\lambda_2 = 0.0005$, and equal to or greater than 80% for $\lambda_3 = 0.0001$. This fact is expected due to the results of Table 3.

For the remaining three optimizers, the accuracy equal or greater than 90% is achieved before the half of the total epochs ($4,000$), so good results could be obtained in half the time. Notice that, in the case of the BPGe (for all the learning rate values), an accuracy equal or greater than 95% is achieved in less than $1,450$ epochs. Therefore, although BPGe requires a few more minutes than the other optimizers to compute $4,000$ epochs, it achieves the same or better accuracy in fewer epochs, potentially reducing training time. Moreover, the best results (less loss and greater accuracy for the test, seen in Table 3) are obtained with $\lambda_1 = 0.001$ for the BPGe and an accuracy equal to or greater than 95% is obtained in epoch 120. This would allow us to obtain remarkable results in few epochs and therefore in a very short time.

Table 4: First epoch in the training process of the classification between movement and rest of the HAR using smartphones database in which the accuracy of the training set is equal to or greater than 80%, 85%, 90% and 95% for the four optimizers and the considered learning rates.

Accuracy (%)		$\geq$ 80%	$\geq$ 85%	$\geq$ 90%	$\geq$ 95%
SGD	λ_1=0.001	761	–	–	–
	λ_2=0.0005	1522	–	–	–
	λ_3=0.0001	–	–	–	–
RMSProp	λ_1=0.001	6	33	177	2006
	λ_2=0.0005	13	67	352	3981
	λ_3=0.0001	74	351	1634	–
Adam	λ_1=0.001	10	41	170	1996
	λ_2=0.0005	19	77	345	3904
	λ_3=0.0001	86	362	1729	–
BPGe	λ_1=0.001	2	4	12	120
	λ_2=0.0005	2	6	24	287
	λ_3=0.0001	5	20	106	1420

7 Conclusion

This paper deals with the training process of a Reservoir Computing algorithm. We have formulated the supervised learning problem using the recently introduced Bregman Proximal Gradient with extrapolation algorithm developed for non-convex optimization problems. We have applied the new algorithm to binary classification problems. Two test examples are presented to illustrate the effectiveness of the proposed approach, one based on chaotic data classification in the classical Lorenz system and the other on the Human Activity Recognition (HAR) using smartphones dataset for movement-rest classification. From tests and comparisons with Stochastic Gradient Descent (SGD), Root Mean Square Propagation(RMSProp) and Adaptive Moment estimation (Adam), it appears that the new methodology is highly competitive in terms of speed of convergence, quality and accuracy of the predicted models. It is remarkable that just a few epochs permit an accuracy greater than 90% (for training set). In particular, for the Lorenz test problem, the performance of the BPGe is comparable to widely used optimizers for ANNs as Adam (even improving the results a little). In the HAR using smartphones dataset, it can be seen that BPGe improves considerably the results, accelerating also the training process. Part of our current research is to extend this approach to more generic ANNs.

Acknowledgments

This work was supported by the Agencia Estatal de Investigación (grant number PID2021-122961NB-I00); the European Regional Development Fund and the Diputación General de Aragón (grant number E24-23R).

Data Availability

Data available on request from the authors.

References

[1] A. Srivastava, B. S. Rawat, G. Singh, V. Bhatnagar, P. K. Saini, and S. A. Dhondiyal, “A review of optimization algorithms for training neural networks,” in *2023 International Conference on Sustainable Emerging Innovations in Engineering and Technology (ICSEIET)*, pp. 886–890, 2023.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[3] G. Hinton and T. Tieleman, “Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, pp. 26–31, 2012.

[4] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[5] N. Parikh, S. Boyd, *et al.*, “Proximal algorithms,” *Foundations and trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[6] N. Polson, J. G. Scott, and B. T. Willard, “Proximal algorithms in statistics and machine learning,” *Statistical science*, vol. 30, no. 4, pp. 559–581, 2015.

[7] X. Zhang, R. Barrio, M. A. Martínez, H. Jiang, and L. Cheng, “Bregman proximal gradient algorithm with extrapolation for a class of nonconvex nonsmooth minimization problems,” *IEEE Access*, vol. 7, pp. 126515–126529, 2019.

[8] P. Jain and P. Kar, “Non-convex optimization for Machine Learning,” *Foundations and Trends in Machine Learning*, vol. 10, no. 3-4, pp. 142–363, 2017.

[9] D. Verstraeten, B. Schrauwen, M. d’Haene, and D. Stroobandt, “An experimental unification of Reservoir Computing methods,” *Neural networks*, vol. 20, no. 3, pp. 391–403, 2007.

[10] M. Cucchi, S. Abreu, G. Ciccone, D. Brunner, and H. Kleemann, “Hands-on Reservoir Computing: a tutorial for practical implementation,” *Neuromorphic Computing and Engineering*, vol. 2, no. 3, p. 032002, 2022.

[11] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization.,” *Journal of machine learning research*, vol. 12, no. 7, 2011.

[12] J. Bolte, S. Sabach, M. Teboulle, and Y. Vaisbourd, “First order methods beyond convexity and Lipschitz gradient continuity with applications to quadratic inverse problems,” *SIAM Journal on Optimization*, vol. 28, no. 3, pp. 2131–2151, 2018.

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32, NeurIPS 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019.

[14] H. Jaeger, “The “echo state” approach to analysing and training recurrent neural networks-with an erratum note,” *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[15] W. Maass, T. Natschläger, and H. Markram, “Real-time computing without stable states: A new framework for neural computation based on perturbations,” *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[16] J. J. Steil, “Backpropagation-decorrelation: Online recurrent learning with O(N) complexity,” in *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, vol. 2, pp. 843–848, 2004.

[17] S. Shahi, F. H. Fenton, and E. M. Cherry, “Prediction of chaotic time series using Recurrent Neural

Networks and Reservoir Computing techniques: A comparative study," *Machine learning with applications*, vol. 8, p. 100300, 2022.

[18] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 12, 2017.

[19] D. Verstraeten, B. Schrauwen, and D. Stroobandt, "Reservoir-based techniques for speech recognition," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pp. 1050–1053, 2006.

[20] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural networks*, vol. 20, no. 3, pp. 335–352, 2007.

[21] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural Networks: Tricks of the Trade: Second Edition. Lecture Notes in Computer Science (LNCS, volume 7700). Springer Berlin, Heidelberg*, pp. 659–686, 2012.

[22] E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.*, vol. 20, pp. 130–141, 1963.

[23] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, 1985.

[24] T. Tél and Y.-C. Lai, "Chaotic transients in spatially extended systems," *Physics Reports*, vol. 460, no. 6, pp. 245–275, 2008.

[25] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 7, p. 073146, 2023.

[26] J. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, and X. Parra, "Human Activity Recognition Using Smartphones." UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C54S4K.

[27] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *The European Symposium on Artificial Neural Networks*, 2013.

Discussion of Unpublished Articles

In this chapter we discuss the unpublished articles. Section 5.1 is devoted to the preprint **Deep Learning for analyzing chaotic dynamics in biological time series: Insights from frog heart signals** [5]. Section 5.2 focuses on the preprint **Graphical preprocessing techniques to generalize dynamical behavior analysis with Deep Learning** [6]. Finally, in Section 5.3 the preprint **Bregman Proximal Gradient with extrapolation to train a Reservoir Computing network** [7] is discussed.

5.1 Deep Learning for Analyzing Chaotic Dynamics in Biological Time Series: Insights from Frog Heart Signals [5]

*Our objective in the preprint **Deep Learning for analyzing chaotic dynamics in biological time series: Insights from frog heart signals** [5] is to study if Deep Learning (DL) can be used to analyze chaotic dynamics in biological time series, a task that is not always feasible with classical techniques. We combine the use of Deep Learning techniques, the universality of chaotic dynamics, and heart-like dynamical information to construct a new automatic algorithm (human supervision is not needed) with 4 steps that is able to detect chaos in experimental heart time series. To show the performance of this algorithm we use frog heart dynamics data. The DL part is carried out using PyTorch [107].*

When using Deep Learning, one of the key points is the training process, during which data is used to enable the network to learn from it. Normally, when we work with experimental recordings, there is not enough data to address the complete training process, so other alternatives have to be used. Numerous dynamical phenomena are universal, so we consider the Logistic map [68] (equation 3.1), a simple one-dimensional discrete dynamical system with great dynamical richness (see Figure 3 in [5]), to train 10 randomly initialized Long Short-Term Memory (LSTM) networks with the same architecture as the network used for the chaos detection task in [2] (*Step 1* of the automatic algorithm proposed in [5]).

Notice that to train the networks we have used generic data, but our final goal is to perform chaos analysis in experimental heart time series. Therefore, at this point of the algorithm, we use some heart-like dynamical information to particularize the method to the nature of our experimental data.

Recall that the action potentials are the electrical signals generated as a consequence of the changes in the membrane potential. The time interval between the onset of the cell depolarization and the completion of its repolarization is known as the action potential duration (see Figure 1 in [5]). The action potential duration restitution curve (that represents the dynamics of a single cardiomyocyte) fitted to the kinetics of the Beeler-Reuter model [112] results in the discrete map [69]

$$\mathrm{APD}_{i+1} = 258 + 125\exp(-0.068(n\mathrm{BCL} - \mathrm{APD}_i - 43.54)) - 350\exp(-0.028(n\mathrm{BCL} - \mathrm{APD}_i - 43.54)),$$

with APD_i the action potential duration corresponding to the i-th stimulus, BCL the time interval between two consecutive pacing stimuli (we use it as bifurcation parameter), and n the parameter block. In Figure 4 of [5] we have the bifurcation diagram and LEs analysis of a one-parameter line of such discrete model.

The 10 LSTM networks trained with the Logistic map data in the *Step 1* of the algorithm are used to perform a chaos analysis in a one-parameter line of the previous heart map model (*Step 2*). In particular, such dynamical behavior study is performed using the original time series, as well as a noisy version obtained by adding component-wise Gaussian noise with two different noise strengths. The measures used to describe the Deep Learning results are accuracy-like measures (to ensure that the chaos detection task has been successful) and noise indicators (to ensure that the detections are robust against noise).

Some criteria (based on the accuracy-like measures and the noise indicators) are imposed to choose the network that has been able to perform properly a chaos analysis in the one-parameter line of the heart map model (*Step 3* of our algorithm). In Table 1 of [5] we show the results for the network that has met all the criteria. The values corresponding to the percentage of chaotic samples classified correctly by the DL network (accuracy chaotic), although meeting the criteria, are not really good. Most of these misclassified time series are in boundary regions between different dynamical regimes and have a behavior that is not expected to be present in the Logistic map used for training (see Figure 4 in [5]). Notice that the unique information available for the DL network is the time series, so the task is very difficult and we can expect that the network will fail in some cases. Finally, the network that meets the criteria of *Step 3* of the algorithm is the one used to perform the chaos analysis of experimental time series (*Step 4*).

The whole Deep Learning algorithm for chaos analysis of heart time series is summarized in Algorithm 1 of [5].

The experimental dataset in which we want to perform a chaos analysis consists of 52 time series (13 with chaotic and 39 with regular behavior) of the action potential duration of a frog cardiomyocyte under different pacing rates. These samples have different lengths (number of time points), the shortest one has length 15, and the longest has length 205. Notice that they are very short time series. In fact, about 80% of the samples have at most 100 points, which makes their chaotic classification difficult with classical techniques. Applying the explained algorithm (that is, using this experimental data during *Step 4*), we obtain the chaos analysis summarized in Figure 7 and Table 2 of [5]. We obtain an accuracy greater than 90% in the dynamical classification, with a good balance between the percentage of samples with each behavior (regular and chaotic) that have been correctly detected. If we analyze in detail the samples that have been misclassified (see Figure 8 in [5]), we can see that they have chaotic transient or seem to correspond to orbits in a period-doubling cascade, which can difficult their classification because of their short length.

From this paper [5], it can be concluded that Deep Learning is a good technique to address experimental time series analyses. We have shown that Artificial Neural Networks trained in a generic one-dimensional discrete dynamical system (the Logistic map) can be used to perform chaos analyses in particular problems once the most suitable one has been selected. Moreover, our study clearly demonstrate the universality of some dynamical phenomena.

5.2 Graphical Preprocessing Techniques to Generalize Dynamical Behavior Analysis with Deep Learning [6]

Our objective in the preprint ***Graphical preprocessing techniques to generalize dynamical behavior analysis with Deep Learning*** *[6] is to study if combining graphical preprocessing techniques (time series imaging methods) and Deep Learning, we can generalize dynamical behavior analysis through different discrete dynamical systems. We train from scratch the well-known AlexNet (Convolutional Neural Network with a not very complicated architecture) in the Logistic map, and we use such trained network to perform dynamical behavior analyses in other discrete dynamical systems (the Circle map and the Hénon map), generalizing the chaos detection task. To achieve this, a proper preprocessing of the time series of the dynamical systems is needed. In particular, we consider different time series imaging methods (Gramian Angular Field, Markov Transition Field, and Recurrence Plot) as graphical preprocessing techniques and we compare their performance. All the DL experiments are performed using PyTorch [107].*

Time series imaging techniques are used to transform time series into images, benefiting from the advantages of Convolutional Neural Networks that can process them. One of these techniques is the

Gramian Angular Field (GAF) [73, 74], which is based on the transformation of the time series into polar coordinates. In the case of the Markov Transition Field (MTF) [73, 74], the core of the method is the use of Markov transition matrices. On the other hand, the Recurrence Plot (RP) [71], that was born in the field of Dynamical Systems, uses the distance between the time series points to define the image. In Figure 1 of [6], we show a couple of examples of how these graphical preprocessing techniques seem to maintain the dynamical properties of the time series (repeated structures for regular behavior, and unclear patterns for chaos).

The workflow used to generalize dynamical behavior analyses using Deep Learning consists of three main steps. First of all, we identify dynamical regimes, such as equilibrium points and escape orbits, that can be trivially classified. Then, we use Piecewise Aggregate Approximation (PAA) [113] to reduce the remaining time series into the usual dimension for image classification tasks (length 224). One of the three graphical preprocessing techniques (GAF, MTF or RP) is applied to the reduced time series. Finally, the AlexNet [114, 115] trained in the Logistic map [68] (equation 3.1) is used to distinguish orbits with regular (non-equilibrium) and chaotic behavior.

The Circle map [75] is a one-dimensional discrete dynamical system given by

$$\theta_{n+1} = \theta_n + \Omega - K\frac{\sin(2\pi\theta_n)}{2\pi} \qquad \text{mod } 1,$$

with θ_n the variable at discrete time n, Ω the natural frequency parameter, and K the coupling strength parameter. Notice that this system maps the circle to itself.

The aforementioned workflow that combines graphical preprocessing techniques and Deep Learning is used to study the dynamical behavior of the (K,Ω)-parametric plane of the Circle map. In Figure 3 of [6] we have depicted the analyses obtained by each technique (including the ground truth given by the classical technique of Lyapunov exponents [77]) when fixed and random initial conditions are used. With the three graphical preprocessing techniques, a good general overview of the different dynamical regimes is obtained, and with random initial conditions, multistability is easily detected. However, if the obtained plots are studied in detail it can be seen that MTF and RP (that perform similarly) are capable of detecting some tiny shrimp-shaped regular regions [116, 117, 118, 119] that with GAF method are misclassified as chaotic. As confirmed by the accuracy results (percentage of samples properly classified) in Table 3 of [6], the MTF and the RP methods perform better.

The Hénon map [76] is a two-dimensional discrete dynamical system given by

$$\begin{cases} x_{n+1} &= 1 + y_n - a x_n^2, \\ y_{n+1} &= b x_n, \end{cases}$$

with (x_n, y_n) the variables at the n-th iteration, and (a,b) the bifurcation parameters. This map has strange attractors for some values of its parameters.

We use the method that applies Deep Learning with graphical preprocessing techniques to perform a dynamical behavior analysis of the (a,b)-parametric plane of the Hénon map. In particular, just the x variable of the system is used for the classification (GAF and MTF methods work with only single-variable time series). The analyses obtained with Deep Learning and each graphical preprocessing technique (as well as the results given by the classical technique of Lyapunov exponents [60] that are used as ground truth) are depicted in Figure 4 of [6]. It can be seen that with all the preprocessing methods we can obtain good studies (this is also supported by accuracy values given in Table 4 of [6]). However, as occurred in the Circle map, the GAF method fails in some regions (for example, some tiny shrimp-shaped zones have not been detected properly).

From this paper [6], it can be concluded that, although Deep Learning is only trained with time series from the Logistic map, thanks to graphical preprocessing techniques (GAF, MTF and RP), it can be used to perform dynamical behavior analyses in other discrete dynamical systems that have not been involved in the training process. In fact, notice that when using for example classical techniques as Lyapunov exponents [60, 77] for classification, the algorithm is particularized to each system, but with this new workflow in which Deep Learning is involved, it is not necessary. Moreover, as seen in the

Circle map, some phenomena as multistability can be detected easily; or as checked with the Hénon map, just partial information (only one variable of the system) can be used obtaining good results. The three graphical preprocessing techniques perform properly, however MTF and RP seem to be better.

5.3 Bregman Proximal Gradient with Extrapolation to Train a Reservoir Computing Network [7]

Our objective in the preprint ***Bregman Proximal Gradient with extrapolation to train a Reservoir Computing network*** *[7] is to adapt the Bregman Proximal Gradient with extrapolation algorithm to train a (supervised) Reservoir Computing network (a special type of Recurrent Neural Network) for binary classification tasks. We compare its performance with widely used training optimizers in Deep Learning (SGD, RMSProp, and Adam), showing that it is highly competitive. The two test examples that we have considered are the dynamical behavior classification (chaos and regular) for the Lorenz system, and the movement and rest classification in the Human Activity Recognition (HAR) using smartphones dataset. The routines of SGD, RMSProp and Adam provided by PyTorch [107] have been used.*

One of the key points of Deep Learning is the training of Artificial Neural Networks. It consists of finding the value of the (trainable) weights and biases that minimize the loss function for the given data. Such function measures the error between the network output and the corresponding label (expected output). Explicit-gradient optimization algorithms as SGD (Stochastic Gradient Descent) [83], RMSProp (Root Mean Square Propagation) [85] and Adam (Adaptive Moment estimation) [84] are commonly used for this training task. The Bregman Proximal Gradient with extrapolation (BPGe) [88] is an implicit-gradient optimizer that generalizes a family of Proximal Gradient algorithms. It can be applied to non-convex optimization. The formulation of this optimizer is

$$\begin{aligned} y_{i-1} &\leftarrow x_{i-1} + \beta_k (x_{i-1} - x_{i-2}), \\ x_i &\leftarrow \arg\min_x \{ g(x) + \langle \nabla f(y_{i-1}), x - y_{i-1} \rangle + D_h(x, y_{i-1})/\lambda_k \}, \end{aligned}$$

where $f + g$ is the function that has to be minimized, λ_k is the learning rate, β_k is the extrapolation parameter (that can be obtained with a line search algorithm [88]), and D_h is a Bregman distance [120]. As a first approach, if we compare SGD, RMSProp, Adam and BPGe in a benchmark minimization function (the three-hump camel function), we obtain that BPGe uses a different route to find the global minimum and with a lower number of iterations it is already in the neighborhood of such minimum (see Figure 1 in [7]).

The Echo State Network (Reservoir Computing architecture) consists of an input layer, a reservoir (set of neurons with recurrent connections) and an output (classification) layer. All the parameters are fixed except for the weights and biases of the output layer that have to be trained. This simplifies the reformulation of BPGe to the training process. However, as we consider the Cross-Entropy loss with L^2-penalty, it is not trivial to adapt the BPGe to Reservoir Computing. In particular, the key point is to obtain the gradient of the Cross-Entropy loss, whose computation has to be separated in different cases according to the following conditions: if the element of the gradient is the derivative with respect to a weight or a bias, and if this trainable parameter is or not involved in the computation of the network score for the labeled class.

The Lorenz system [63] (equation 3.2) is a paradigmatic continuous dynamical system that exhibits chaotic dynamics. We use the four optimizers and three different learning rate values ($\lambda_1 = 0.001$, $\lambda_2 = 0.0005$, $\lambda_3 = 0.0001$) to compare performance for the chaos detection task in this system.

In Table I of [7] we have gathered together the loss and accuracy values for the final network (the one chosen by early stopping technique [83]) in training, validation and test sets. For all the optimizers and learning rates combinations, the network seems to have learned properly. Comparing the results, the SGD seems to be the one that works the worst, and BPGe with learning rate $\lambda_3 = 0.0001$ gives the best results for the test set.

To continue the analysis, in Figure 3 of [7] we have depicted the loss and accuracy evolution for the training dataset throughout all the training process for all the optimizers and learning rate values. On the one hand, it highlights that the BPGe always outperforms the other optimizers, the SGD is always left behind, and RMSProp and Adam evolves similarly. On the other hand, it is remarkable that BPGe has a noisy behavior for $\lambda_1 = 0.001$ with this effect being mitigated for $\lambda_2 = 0.0005$, and totally disappeared for $\lambda_3 = 0.0001$. We conjecture that this phenomenon is related to a bound imposed in the learning rate in [88]. Comparing wall time for training process for each optimizer, we obtain that widely used SGD, RMSProp and Adam need around 27 minutes while BPGe needs around 30 minutes.

An interesting (and necessary) study is to check in which epoch of training process each optimizer with each learning rate has surpassed a certain training accuracy value. This would allow us to check which optimizer is faster at learning. Table II of [7] summarizes such analyses (for these results validation accuracy has also been controlled to confirm that the network does not suffer from overfitting). Only the BPGe optimizer achieves an accuracy greater than or equal to 95%. In fact, the SGD algorithm only surpasses the 90% for one learning rate value. On the other hand, RMSProp and Adam perform quite similarly. It is surprising that BPGe needs less than 20 epochs to reach an accuracy greater than 90% (approximately 10 seconds taking into account wall time for training), devoting the remaining epochs to fine-tune the results. This is confirmed in Figure 4 of [7], where we depict the chaos detection studies obtained in a one-parameter line of the Lorenz system using the networks with the value of the trainable parameters obtained throughout the epochs during training process. Moreover, this figure also illustrates the final analysis performed with the trained network with each optimizer: although RMSProp, Adam and BPGe results are similar, BPGe is able to detect more regular windows in the large chaotic regions (providing a more detailed analysis).

The Human Activity Recognition (HAR) using smartphones dataset [121, 122] is a database with recordings from 30 subjects performing several activities that can be classified as rest or movement. In particular, we use the 3-axial estimated body acceleration (three-dimensional time series of length 128) as input for the network, and we expect its output to be the correct rest vs movement classification. As before, we consider the four optimizers and the three different learning rate values ($\lambda_1 = 0.001$, $\lambda_2 = 0.0005$, $\lambda_3 = 0.0001$).

In Table III of [7], we summarize the loss and accuracy values for training, validation and test datasets for all the possible combinations between optimizers and learning rates. The SGD is the algorithm that gives worst results, RMSProp and Adam perform similarly, and BPGe seems to be the best option (lowest loss and largest accuracy for all learning rate values). In fact, as can be seen in Table IV of [7], the accuracy in the training set surpasses the 95% in less than 1450 epochs for all learning values when BPGe is used.

From this paper [7], it can be concluded that the Bregman Proximal Gradient with extrapolation algorithm is a promising optimizer for training Reservoir Computing networks. In particular, we have reformulated it for binary classification tasks. We have tested its performance in the chaos detection task of a classical dynamical system (the Lorenz model), where BPGe is comparable to commonly used optimizers as Adam. We have also used it in the Human Activity Recognition using smartphones dataset to distinguish between movement and rest activities, where BPGe is considerably better. Although BPGe slightly increases the wall time for the training process by a few minutes compared to the other optimizers, it achieves good accuracy quickly, providing promising results.

Posters

This chapter includes some posters presented at conferences by the PhD candidate Carmen Mayora Cebollero during her doctoral thesis:

- “Chaos detection using Deep Learning”, Dynamics Days US 2023 (Online), 09/01/2023-11/01/2023
- “Deep Learning for chaos detection”, 2023 SIAM Conference on Dynamical Systems (Portland, Oregon, US), 14/05/2023-18/05/2023
- “Deep Learning chaoticity analysis of biological time series: Frog heart dynamics”, 13th European Conference on Mathematical and Theoretical Biology (Toledo, Spain), 22/07/2024-26/07/2024
- “Dominance of tripod gait in bipartite networks”, 2023 International Physics of Living Systems (iPoLS) Network Annual Meeting (Atlanta, Georgia, US), 01/08/2023-04/08/2023

Chaos Detection Using Deep Learning

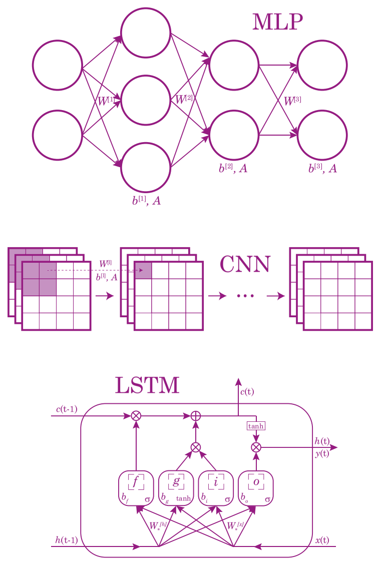
Carmen Mayora-Cebollero, Roberto Barrio, Álvaro Lozano, Ana Mayora-Cebollero, Antonio Miguel, Alfonso Ortega, Sergio Serrano, Rubén Vigara

Universidad de Zaragoza, Spain

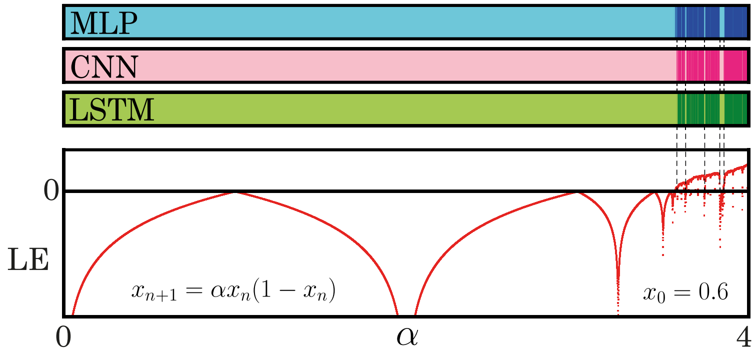
Chaos detection is one of the problems that we have to consider when the behaviour of a dynamical system is studied. Usually to decide if a dynamical system has regular or chaotic behaviour, classical techniques as **Lyapunov Exponents (LEs)** [1] are used [2]. However, recently, some authors have proposed to use Deep Learning to deal with this problem [3].

Deep Learning (DL) includes all the techniques that use architectures based on layers of artificial neurons to learn from data with several levels of abstraction. Nowadays there are many DL architectures in the literature. To carry out the chaos detection task in a dynamical system [4], we have chosen three of the best known examples: MLP, CNN and LSTM.

DL ARCHITECTURES

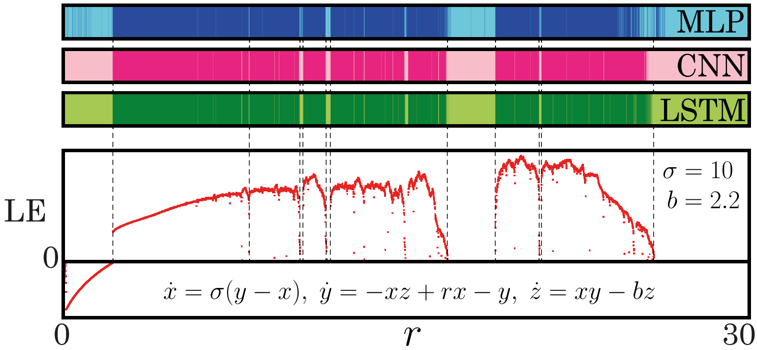


DISCRETE DYNAMICAL SYSTEM. LOGISTIC MAP



MLP		CNN		LSTM	
ACCURACY REGULAR (%)	ACCURACY CHAOTIC (%)	ACCURACY REGULAR (%)	ACCURACY CHAOTIC (%)	ACCURACY REGULAR (%)	ACCURACY CHAOTIC (%)
99.211±0.284	98.551±0.222	99.853±0.029	98.995±0.499	99.388±0.402	98.188±0.729

CONTINUOUS DYNAMICAL SYSTEM. LORENZ SYSTEM



MLP		CNN		LSTM	
ACCURACY REGULAR (%)	ACCURACY CHAOTIC (%)	ACCURACY REGULAR (%)	ACCURACY CHAOTIC (%)	ACCURACY REGULAR (%)	ACCURACY CHAOTIC (%)
91.015±0.104	96.172±0.174	95.652±1.489	97.638±0.644	96.864±1.020	98.314±0.322

REFERENCES

[1] A. Wolf, J.B. Swift, H.L. Swinney, J.A. Vastano, *Determining Lyapunov Exponents from a Time Series*, Physica D, **16**, 285-317 (1985).

[2] R. Barrio, S. Serrano, *A Three-Parametric Study of the Lorenz Model*, Physica D, **229**, 43-51 (2007).

[3] N. Boullé, V. Dallas, Y. Nakatsukasa, D. Samaddar, *Classification of Chaotic Time Series with Deep Learning*, Physica D, **403**, 132261 (10 pages) (2020).

[4] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, R. Vigara, *Deep Learning for Chaos Detection*, preprint (2023).

CONCLUSIONS

- In the case of the Logistic map (the discrete dynamical system test problem), we can see that all the networks give similar results and all of them are able to detect the boundary between the two dynamical regimes and the regular windows in the chaotic region.
- In the case of the Lorenz system (the continuous dynamical system test problem), the DL networks that work best are the CNN and the LSTM. This is an expected result because these two networks are able to process spatial or temporal information, and the MLP is not. Moreover, the network that seems to be the most accurate detecting the boundary between both dynamical regimes is the LSTM. This is also an expected fact since the LSTM is specific for time series tasks.
- We conclude that DL can be used to perform the chaos detection task. However, a deeper study is necessary to know how far we can go using these new techniques to detect chaos in a dynamical system.

cmayora@unizar.es

DEEP LEARNING FOR CHAOS DETECTION

Carmen Mayora-Cebollero, Roberto Barrio, Álvaro Lozano, Ana Mayora-Cebollero,
Antonio Miguel, Alfonso Ortega, Sergio Serrano, Rubén Vigara

Computational Dynamics (CoDy) group, Universidad de Zaragoza, Spain

Introduction

One of the main problems in the area of Dynamical Systems is the **detection of chaotic regions in the parametric space**. In the literature, classical techniques as **Lyapunov Exponents** (LEs) have been used to detect chaos [1, 2]. Recently some authors have applied new techniques as **Deep Learning** (DL) to this task [3, 4]. Our goal is to use DL to study the dynamical behaviour (chaotic or regular) of the **Lorenz system** [5].

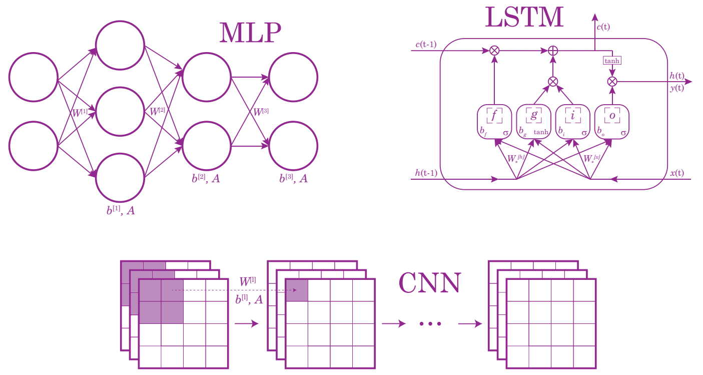
Lorenz System

$$\dot{x} = \sigma(y - x), \;\; \dot{y} = -xz + rx - y, \;\; \dot{z} = xy - bz,$$

where (x, y, z) are the variables and (σ, r, b) are the bifurcation parameters.

Deep Learning

Deep Learning includes all the techniques that use **(Deep) Artificial Neural Networks** to learn from data with several levels of abstraction. Artificial Neural Networks are formed of artificial neurons (loosely inspired by their biological counterparts) organized in layers. Among all DL architectures, we have chosen three of the best known: **MLP** (Multi-Layer Perceptron), **CNN** (Convolutional Neural Network), **LSTM** (Long Short-Term Memory).



How Do We Measure the Performance of DL Networks?

$$\textbf{Accuracy}\,(\%) = \frac{T_C + T_R}{T_C + F_R + T_R + F_C} \times 100$$

$$\textbf{Accuracy Regular}\,(\%) = \frac{T_R}{T_R + F_C} \times 100, \;\; \textbf{Accuracy Chaotic}\,(\%) = \frac{T_C}{T_C + F_R} \times 100$$

$T_C = \text{True Chaotic}$, $T_R = \text{True Regular}$, $F_R = \text{False Regular}$, $F_C = \text{False Chaotic}$

Conclusions

- Three well-known Deep Learning networks (MLP, CNN and LSTM) have been built and trained to carry out the chaos detection task in the Lorenz system.
- In the one-parameter study the three networks have given good results (accuracy greater than 90% and correct boundary detection). The LSTM seems to provide the best results.
- The LSTM network has been used to perform biparametric and triparametric analyses of the system with good results.
- Up to the knowledge of the authors, the dense triparametric study of the Lorenz system had not been obtained before with any kind of technique.
- A deeper study is necessary to know how far we can go using DL to detect chaos in a dynamical system.

References

[1] R. Barrio, S. Serrano, *A Three-Parametric Study of the Lorenz Model*, Physica D, 229(1), 43-51 (2007).

[2] R. Barrio, S. Serrano, *Bounds for the Chaotic Region in the Lorenz Model*, Physica D, 238(16), 1615-1624 (2009).

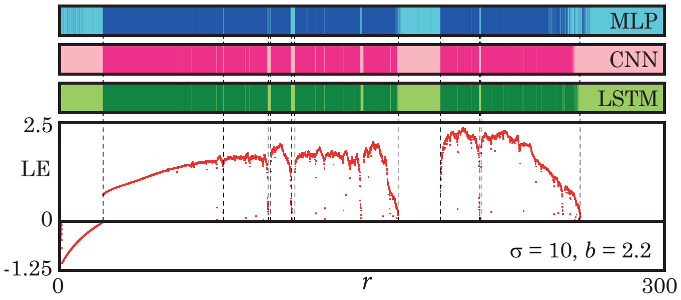
[3] N. Boullé, V. Dallas, Y. Nakatsukasa, D. Samaddar, *Classification of Chaotic Time Series with Deep Learning*, Physica D, 403, 132261 (10 pages) (2020).

[4] A. Celletti, C. Gales, V. Rodriguez-Fernandez, M. Vasile, *Classification of Regular and Chaotic Motions in Hamiltonian Systems with Deep Learning*, Scientific Reports, 12(1), 1890 (12 pages) (2022).

[5] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, R. Vigara, *Deep Learning for Chaos Detection*, Preprint (2023).

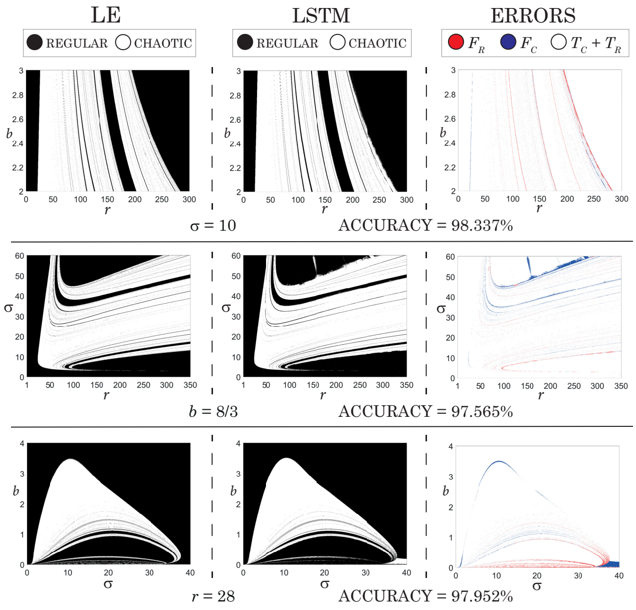
Computational Dynamics (CoDy) group ⟶ https://cody.unizar.es/

One-Parameter Study of the Lorenz System

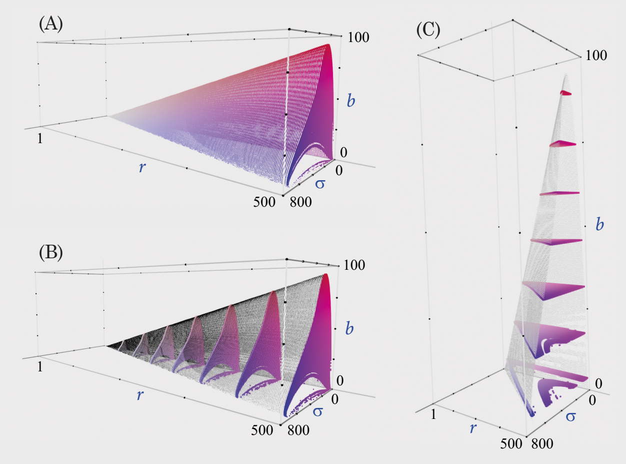


MLP		CNN		LSTM	
ACCURACY REGULAR (%)	ACCURACY CHAOTIC (%)	ACCURACY REGULAR (%)	ACCURACY CHAOTIC (%)	ACCURACY REGULAR (%)	ACCURACY CHAOTIC (%)
91.015±0.104	96.172±0.174	95.652±1.489	97.638±0.644	96.864±1.020	98.314±0.322

Biparametric Study of the Lorenz System



Triparametric Study of the Lorenz System



These representations have been obtained using the results of the dense triparametric analysis performed with the LSTM network. Panel A. Boundary between both dynamical regimes and chaotic region for $r = 500$. Panels B and C. Several 2D planes (shaded in black, the boundary between both dynamical regimes).

cmayora@unizar.es

DEEP LEARNING CHAOTICITY ANALYSIS OF BIOLOGICAL TIME SERIES: FROG HEART DYNAMICS

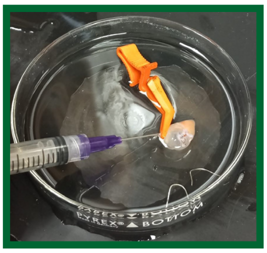
Carmen Mayora-Cebollero[1], Roberto Barrio[1], Flavio H. Fenton[2], Mikael J. Toye[2]

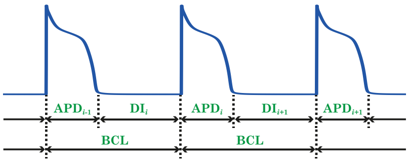
[1] Universidad de Zaragoza, Spain [2] Georgia Institute of Technology, USA

Introduction

Classical techniques as Lyapunov Exponents are often used to perform chaoticity analysis of dynamical systems. However, these techniques sometimes present problems when used on real data as recordings are generally short and noisy. When dealing with large experimental datasets it is necessary to have an automatic algorithm for chaoticity analysis since human intervention on the entire dataset is not feasible. Recently, some authors have used Deep Learning (Artificial Neural Networks) to detect chaos in a dynamical system [1]. **Could Deep Learning (DL) be applied to perform a chaoticity analysis in an experimental dataset?** [2]

Our **experimental dataset** contains 52 time series with different lengths (minimum length 15 and maximum length 205). Such time series correspond to the **Action Potential Duration (APD) of frog heart dynamics for different pacing rates (Basic Cycle Length, BCL)**.





$$\boxed{APD_i + DI_{i+1} = BCL}$$

APD = Action Potential Duration
DI = Diastolic Interval
BCL = Basic Cycle Length

Algorithm for DL Chaoticity Analysis of Biological Time Series

Step 1: Artificial Neural Networks Framework. Train 10 randomly initialized recurrence-like Artificial Neural Networks with time series from the Logistic Map.

Remark. Deep Learning includes all the Artificial Intelligence techniques that use Artificial Neural Networks to learn from data with several levels of abstraction. Recurrence-like Artificial Neural Networks are commonly used for sequential processing since they retain information from past times.

Step 2: DL Chaoticity Analysis of an APD Heart Model. Perform a test analysis in an APD heart model with each Artificial Neural Network trained in *Step 1*.

Step 3: Select one Artificial Neural Network. Establish some criteria on the results of *Step 2* to choose automatically one network of *Step 1*.

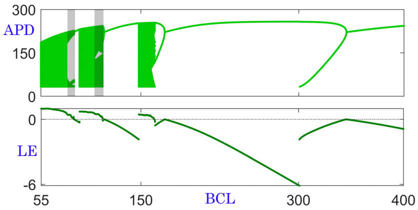
Step 4: DL Chaoticity Analysis of Biological Time Series: Frog Heart Dynamics. Use the network chosen in *Step 3* to perform chaoticity analysis in experimental data.

DL Chaoticity Analysis of an APD Heart Model

APD restitution curve (that describes the dynamics of a single cell) fitted to the kinetics of Beeler-Reuter model [3] gives rise to the discrete equation

$$\begin{aligned}\mathrm{APD}_{i+1} = 258 + 125\exp(-0.068(n\,\mathrm{BCL} - \mathrm{APD}_i - 43.54))\\ -350\exp(-0.028(n\,\mathrm{BCL} - \mathrm{APD}_i - 43.54)),\end{aligned}$$

where n is the parameter block (lower $n \in \mathbb{N}$ such that $n\,\mathrm{BCL} - \mathrm{APD}_i \geq \mathrm{DI}_{\min}$ with $\mathrm{DI}_{\min}$ the minimum DI set to 43.54ms).



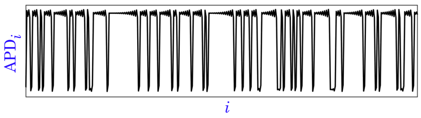
80.292% of the samples have regular behaviour (according to LE value) and the remaining 19.708% are chaotic. 79.803% of regular samples are equilibrium points and the other 20.197% present periodic behaviour.

A DL chaoticity analysis of this APD heart model is performed without noise (no noise), adding Gaussian noise with strength 0.5 ($+0.5\mathcal{N}(0,1)$), and same type of noise with strength 1.0 ($+1.0\mathcal{N}(0,1)$). The following table collects the results for the network chosen in *Step 3*:

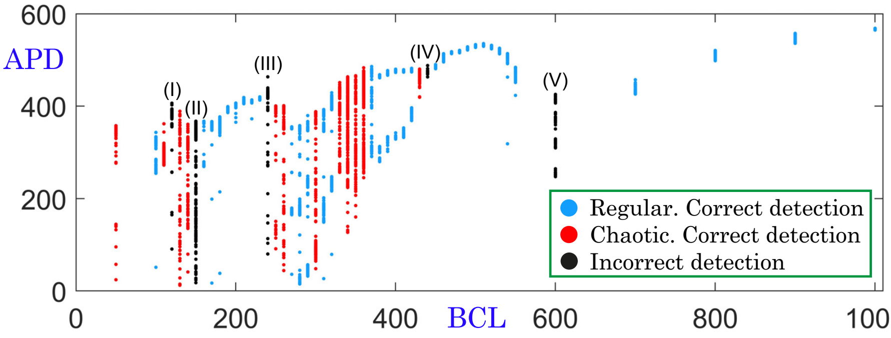
	No noise	$+0.5\mathcal{N}(0,1)$	$+1.0\mathcal{N}(0,1)$
Accuracy	95.092 %	94.825 %	94.750 %
Accuracy Chaotic	76.195 %⋆⋆	75.983 %	75.729 %
Accuracy Regular	99.730 %	99.450 %	99.419 %
Accuracy EPs	100 %	99.649 %	99.649 %
Accuracy POs	98.666 %	98.666 %	98.512 %

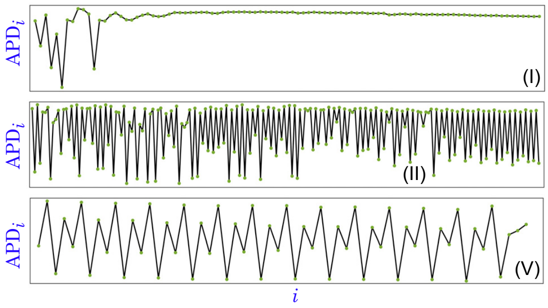
Remark. Accuracy refers to the percentage of samples correctly detected with DL. Accuracy regular (resp. chaotic) corresponds to the percentage of regular (resp. chaotic) samples correctly classified by DL. Accuracy EPs (resp. POs) is the percentage of equilibrium points (resp. periodic orbits) correctly detected as regular with DL.

⋆⋆ Most incorrect DL detections of chaotic samples occur in the grey shaded areas of the bifurcation diagram. This time series is an example of these samples:



DL Chaoticity Analysis of Biological Time Series: Frog Heart Dynamics





	Experimental Results
Accuracy	90.385 % (47 of 52 samples)
Accuracy Chaotic	92.308 % (12 of 13 samples)
Accuracy Regular	89.744 % (35 of 39 samples)

- Samples (I)-(III)-(IV) have been detected as chaotic by the Artificial Neural Network. All these samples have a similar behaviour: short chaotic transient dynamics, asymptotically converging to an equilibrium point (regular). Therefore, the whole time series is chaotic (as DL detected), but dynamically its asymptotic behaviour can be considered as regular (and the samples were labeled under this consideration).
- Sample (II) is detected as regular. The final part of the time series seems to have some periodicity, so it can be considered as regular with a long chaotic transient (in this case the network detection is correct). However, since the behaviour is chaotic most of the time, it was labeled as chaotic.
- Sample (V) is a quasi-periodic orbit detected as chaotic. The score ('probability') for chaotic class given by the Artificial Neural Network is 0.625 (and 0.375 for regular class), so it doubts on the detection.

Conclusions

- Experimental data has some drawbacks as noisy and short recordings. An algorithm based on Deep Learning (Artificial Neural Networks) is proposed to deal with the chaoticity analysis of biological time series.
- A test of the robustness of Artificial Neural Networks when noise is present is carried out in an APD heart model (Beeler-Reuter).
- This analysis allows to select an Artificial Neural Network that can properly perform a chaoticity analysis of biological time series of frog heart dynamics.
- The proposed algorithm does not require human supervision, so it can be considered as an automatic technique for chaoticity analysis of large datasets of biological time series.

References

[1] 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).

[2] R. Barrio, F.H. Fenton, C. Mayora-Cebollero, M.J. Toye, *Deep Learning Chaoticity Analysis of Biological Time Series: Frog Heart Dynamics*, Preprint (2024).

[3] H.M. Hastings, F.H. Fenton, S.J. Evans, O. Hotomaroglu, J. Geetha, K. Gittelson, J. Nilson, A. Garfinkel, *Alternans and the Onset of Ventricular Fibrillation*, Physical Review E, 62(3), 4043 (2000).

Computational Dynamics (CoDy) group ⟶ https://cody.unizar.es/

cmayora@unizar.es

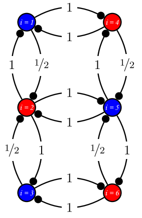
DOMINANCE OF TRIPOD GAIT IN BIPARTITE NETWORKS

Carmen Mayora-Cebollero, Roberto Barrio, Álvaro Lozano, Ana Mayora-Cebollero, Sergio Serrano, Rubén Vigara

Computational Dynamics (CoDy) group (https://cody.unizar.es/), Universidad de Zaragoza, Spain

Central Pattern Generators

Central Pattern Generators (CPGs) are groups of neurons that when activated can generate rhythmic motor patterns even in absence of sensory input. Many coordinated and rhythmic behaviors in organisms as chewing, respiration or walking are driven by CPGs.
CPGs of six coupled neurons following the model of Ghigliazza and Holmes [1] for motoneurons in cockroaches (GH network):

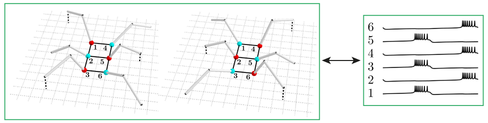


$$\begin{cases} C\dot{v}_i = -\left(I_{Ca} + I_K + I_L + I_{KS}\right) + I_{ext} - (I_{syn})_i & \tau_m = \operatorname{sech}\left(k_{0_K}(v - v_K^{th})/2\right) \\ \dot{m}_i = \epsilon\left(m_\infty - m_i\right)/\tau_m & \tau_w = \operatorname{sech}\left(k_{0_{KS}}(v - v_{KS}^{th})/2\right) \\ \dot{w}_i = \delta\left(w_\infty - w_i\right)/\tau_w & m_\infty = \left(1 + \exp(-2k_{0_K}(v - v_K^{th}))\right)^{-1} \\ \dot{s}_i = \alpha s_\infty(1 - s_i) - \beta s_i & w_\infty = \left(1 + \exp(-2k_{0_{KS}}(v - v_{KS}^{th}))\right)^{-1} \end{cases}$$

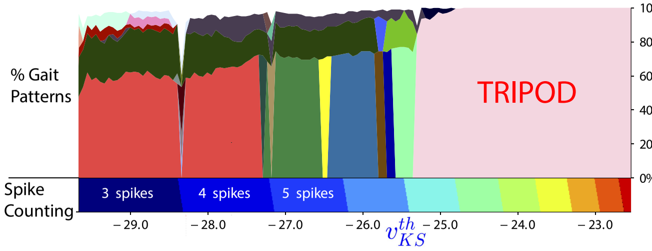
where v_i are voltages, m_i and w_i are gating variables, and s_i are synapses variables $(i = 1, \dots, 6)$. τ_m, τ_w, m_∞ and w_∞ are time scales and steady state gating variables. For more details see [2, 3].

Tripod Gait

Three legs move at a time while the other three remain stationary (three legs are always on the ground).



Study Patterns. Quasi-Monte Carlo Sweeping Technique [2, 3]



Top: **Percentage of the different gaits** in a v_{KS}^{th} -parametric line applying the **quasi-Monte Carlo sweeping technique** [2] with 200 initial conditions for each parametric value. Bottom: Spike-counting sweeping.

Dominance of Tripod Gait. Generalization [4]

As seen with the quasi-Monte Carlo sweeping technique, **the tripod gait is the dominant pattern in a wide region of the parameter space.** The underlying graph of the GH network is bipartite and the tripod gait is a bipartite pattern. **Is the dominance of tripod gait a consequence of network bipartiteness?** GH network is extended to general graphs and quasi-Monte Carlo sweeping technique is applied to all bipartite networks with up to 9 neurons (982 different graphs).

Networks and Graphs

- To identify a neuron network with a graph G, we consider that the vertices of G correspond to the neurons, and the edges represent the (inhibitory) synapses.
- For $q \in \mathbb{N}$, a graph G is **q-colorable** if the set of vertices can be divided into q disjoint non-empty subsets in such a way that adjacent vertices lie on different subsets of the partition. Such decomposition is a **q-coloring** of G.
- A graph G is **bipartite** if it is 2-colorable.
- A synchronization pattern is a **q-color pattern** if the partition defined by the pattern is a q-coloring of the graph.
- **The tripod gait of the GH network is a bipartite pattern (that is, a 2-color pattern).**

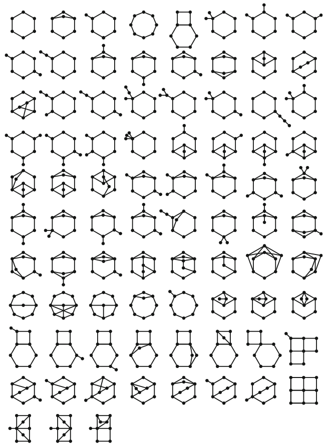
Bipartite Patterns [4]

Evident **dominance of bipartite patterns** (tripod-like patterns) in our study:

- **100/70 rule**: In 100% of the graphs, the bipartite pattern appears in more than 70% of the simulations. In about 70% of the graphs the bipartite pattern appears in 100% of the simulations.
- **95/90 rule**: For around 95% of the graphs, the bipartite pattern appears in at least 90% of the simulations. For nearly 90% of the graphs, the bipartite pattern appears in at least 95% of the simulations.

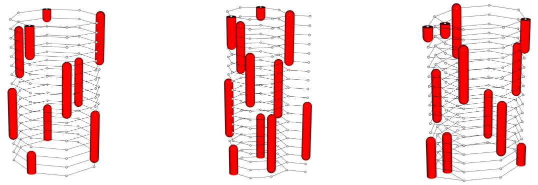
Non-Bipartite Patterns [4]

In 83 of the networks there is at least a **non-bipartite pattern** that appears in more than 5% of the simulations:

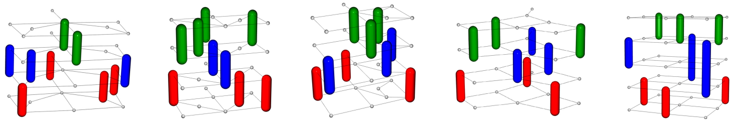


There are 95 non-bipartite patterns: 87 are **3-color patterns** and 8 are **non-color patterns**. The last ones can be divided into two classes:

In 3 patterns, connected neurons do not fire at the same time, but they are in the same subset of the partition because of partial overlappings of bursting intervals (trivial partition).



In 5 patterns, there are connected neurons firing at the same time, which seems to contradict that synapses in our networks are inhibitory. Therefore, they deserve a further investigation.



THE TRIPOD GAIT IS THE DOMINANT PATTERN IN BIPARTITE NETWORKS.

References

[1] R.M. Ghigliazza, P. Holmes. *A Minimal Model of a Central Pattern Generator and Motoneurons for Insect Locomotion*, SIAM J. Appl. Dyn. Syst., 3(4):671–700 (2004).

[2] R. Barrio, Á. Lozano, M. Rodríguez, S. Serrano. *Numerical Detection of Patterns in CPGs: Gait Patterns in Insect Movement*, Commun. Nonlinear Sci. and Numer. Simul., 82:105047 (2020).

[3] R. Barrio, Á. Lozano, M.A. Martínez, M. Rodríguez, S. Serrano. *Routes to Tripod Gait Movement in Hexapods*, Neurocomputing, 461:679–695 (2021).

[4] Á. Lozano, R. Vigara, C. Mayora-Cebollero, R. Barrio. *Ubiquitous Tripod Gait in Bipartite Networks*, Preprint (2023).

cmayora@unizar.es

Conclusions and Future Work

In this doctoral thesis we have demonstrated that standard techniques for dynamical systems are essential for understanding certain phenomena in excitable media. However, as some of these standard methods can be highly computationally expensive, we have applied Deep Learning to deal with dynamical systems studies, which has confirmed that it is a promising and powerful tool. Finally, we have studied how Deep Learning can benefit from dynamical systems and mathematics. In what follows, we draw some conclusions.

- Simple mathematical models such as the Hindmarsh-Rose model that replicates the usual behavior of excitable cells permit us to study in detail some dynamical phenomena. Moreover, in cardiac dynamics, the use of the low-dimensional 3D Luo-Rudy model allows us to verify some conjectures obtained from the study of a higher-dimensional system (Sato model).
- The spike-counting sweeping technique and the numerical continuation of bifurcations permit us to visualize and explain the spike-adding phenomenon in the Hindmarsh-Rose model (for values of the small parameter ε in its standard range): the pencils of the bifurcations which produce the addition of one extra spike are generated in codimension-two bifurcation points contained in the isolas of homoclinic bifurcations.
- In the Hindmarsh-Rose model, geometric bifurcations permit us to understand the changes observed in the spike-adding cascade and the disappearance of codimension-two bifurcation points when the value of the small parameter ε is varied beyond its standard range.
- It is conjectured that the templates of the chaotic attractors of the Hindmarsh-Rose model correspond to subtemplates of the Smale horseshoe, which is progressively filled in as ε decreases to zero.
- The spike-adding phenomenon is also present in realistic processes. On the one hand, there is a clear relationship between spike-adding process and some insect movement pattern transitions. One the other hand, spike-adding corresponds to the creation of Early Afterdepolarizations in cardiac muscle cells that under some circumstances can produce dangerous dynamics as arrhythmias.
- Deep Learning can be used to perform chaos detection in a dynamical system. In fact, it allows us to obtain a dense triparametric analysis of the Lorenz system. To the best of the authors' knowledge, it has not been achieved before with any technique.
- Three Deep Learning architectures (the Multi-Layer Perceptron, the Convolutional Neural Network, and the Long Short-Term Memory cell) are compared for the chaos detection task. Although all of them provide good results, the Convolutional Neural Network and the Long Short-Term Memory perform better (which makes sense as they take into account spatial and temporal information, respectively).
- We have described a new algorithm to perform chaos analysis of experimental time series. It is based on training Artificial Neural Networks in the Logistic map and checking their validity using

a mathematical model of the same nature as the experimental data. This algorithm allows us to study some experimental data obtained from a frog cardiomyocyte.

- The success of the Deep Learning chaos analysis algorithm in the experimental dataset shows the universality of the Logistic map and some dynamical phenomena.

- The combination of graphical preprocessing techniques (time series imaging) and Deep Learning allows us to generalize chaos detection analyses. That is, training an Artificial Neural Network only with data from the Logistic map, we can obtain detailed dynamical behavior analyses (detection of shrimp-shaped regions and multistability zones) in other discrete dynamical systems.

- When we use the Markov Transition Field and the Recurrence Plot as graphical preprocessing techniques, we obtain more accurate results than when the Gramian Angular Field method is applied.

- Deep Learning allows us to obtain the full Lyapunov exponents spectrum of a dynamical system using just short single-variable time series and no additional dynamical information. To the best of the authors' knowledge, this is not possible with any other technique.

- Although using Deep Learning to approximate Lyapunov exponents gives good results with both non-random and random training data, the latter approach performs better as more dynamical variability is provided during training.

- A detailed dynamical classification (tori, hyperchaos and chaos) biparametric analysis can be obtained using the Lyapunov exponents approximations given by Deep Learning.

- Deep Learning provides good results in dynamical systems studies. However, it fails in regions with complex dynamics as the boundaries between different dynamical regimes, and the areas with transient chaos. For example, this occurs for the chaos detection task and the Lyapunov exponents approximation.

- One of the main advantages of using Deep Learning for dynamical systems analyses is time savings. For instance, Deep Learning needs less than 10% of the time required by classical techniques to perform a triparametric chaos detection study or to approximate all the Lyapunov exponents in a biparametric plane.

- We have particularized the Bregman Proximal Gradient with extrapolation (BPGe) algorithm to train a Reservoir Computing network for binary classification tasks. Although it slightly increases wall time (by a few minutes) compared to classical optimizers (SGD, RMSProp and Adam), it provides better results and faster convergence.

- The generalized tripod gait pattern (bipartite pattern) is the dominant one for all the Central Pattern Generator models of 6 to 9 neurons with bipartite connectivity (bipartite network). This demonstrates a correlation between the network topology and its dynamics.

- Dynamical systems, and mathematics in general, can be used to improve Deep Learning. This shows that Deep Learning can benefit from mathematics and dynamical systems, and vice versa.

As shown with the conclusions, in this doctoral thesis we have obtained promising results, however some questions still remain open. In the following we summarize some possible future research topics derived from this work.

- Which dynamical structures of a single Hindmarsh-Rose neuron are maintained when we couple two Hindmarsh-Rose neurons?

- Deep Learning architectures capable of dealing with longer sequences can be useful to obtain better results in regions with complex dynamics.

- Could Deep Learning be used to obtain the Lyapunov exponents spectrum of experimental time series?

- Could the combination of graphical preprocessing techniques and Deep Learning be used to generalized dynamical behavior analysis in continuous dynamical systems?

- We have approximated the full Lyapunov exponents spectrum with just single-variable time series. It would be interesting to analyze how the use of all the system variables affects the results.

- How far can we go using Deep Learning to perform dynamical systems analyses?

- All the advantages provided by the Bregman Proximal Gradient with extrapolation algorithm suggest that it would be interesting to reformulate it to other tasks and Artificial Neural Networks.

- Could the training process of an Artificial Neural Network be improved using other families of optimizers?

- When studying the possible correlation between network topology and its dynamics, some patterns with connected neurons firing at the same time have been found. This seems to contradict the use of inhibitory synapses, so such patterns need further investigation.

- A brief study of non-bipartite 3-color networks has shown that the 3-color pattern is the predominant one. However, the dominance is weaker than the one of bipartite patterns in bipartite networks. Why is the predominance weaker in the former case?

- After studying the interplay between topology and dynamics in networks up to 9 neurons, it seems natural to study networks with a larger number of neurons.

- An interesting topic is the application of dynamical systems techniques to analyze the relationship between the architecture of Artificial Neural Networks and how they work.

Conclusiones y Trabajo Futuro

En esta tesis doctoral hemos mostrado que las técnicas estándar para el análisis de sistemas dinámicos son esenciales para entender ciertos fenómenos en medios excitables. Sin embargo, como algunas de estas técnicas estándar pueden ser computacionalmente caras, hemos aplicado el aprendizaje profundo (*Deep Learning*) para llevar a cabo estudios en diversos sistemas dinámicos, lo que ha confirmado que es una herramienta muy prometedora. Finalmente, hemos estudiado cómo el aprendizaje profundo puede beneficiarse del campo de los sistemas dinámicos y las matemáticas. A continuación mostramos algunas conclusiones.

- Modelos matemáticos simples como el de Hindmarsh-Rose que replican el comportamiento habitual de células excitables nos permiten estudiar en detalle algunos fenómenos dinámicos. Además, en lo referente a dinámica cardiaca, el uso del modelo de baja dimensión conocido como modelo 3D de Luo-Rudy nos permite verificar conjeturas obtenidas del estudio de un sistema con más dimensiones (modelo de Sato).

- La técnica de barrido de conteo de picos (*spike-counting sweeping technique*) y la continuación numérica de bifurcaciones nos permiten visualizar y explicar el fenómeno de *spike-adding* en el modelo de Hindmarsh-Rose (para valores estándar del pequeño parámetro ε): los haces de las bifurcaciones involucradas en el proceso de añadir un pico extra se generan en puntos de bifurcación de codimensión 2 contenidos en las isolas de bifurcaciones homoclínicas.

- En el modelo de Hindmarsh-Rose las bifurcaciones geométricas permiten entender los cambios observados en las cascadas de *spike-adding* y la desaparición de puntos de bifurcación de codimensión 2 cuando el valor del pequeño parámetro ε varía más allá de sus valores estándar.

- Se conjetura que las plantillas (*templates*) de los atractores caóticos del modelo de Hindmarsh-Rose corresponden a subplantillas de la herradura de Smale (*Smale horseshoe*), que se va llenando progresivamente a medida que ε tiende a cero.

- El fenómeno de *spike-adding* también está presente en procesos realistas. Por un lado, hay una clara relación entre el proceso de *spike-adding* y algunas transiciones entre los patrones de movimiento de insectos. Por otro lado, el *spike-adding* equivale a la creación de *Early Afterdepolarizations* en los cardiomiocitos, que bajo ciertas circunstancias pueden dar lugar a dinámicas peligrosas como las arritmias.

- El aprendizaje profundo puede ser utilizado para llevar a cabo detección de caos en un sistema dinámico. De hecho nos ha permitido obtener un análisis triparamétrico denso del sistema de Lorenz. Hasta donde saben los autores, esto no había sido conseguido antes con ninguna otra técnica.

- Comparamos el funcionamiento de tres arquitecturas de aprendizaje profundo (el perceptrón multicapa, la red neuronal convolucional, y la célula de tipo *Long Short-Term Memory*) para la tarea de detección de caos. Aunque todas ellas dan buenos resultados, la red neuronal convolucional y la *Long Short-Term Memory* parecen funcionar mejor (esto tiene sentido ya que tienen en cuenta información espacial y temporal, respectivamente).

- Hemos descrito un nuevo algoritmo para llevar a cabo análisis de caoticidad de series temporales experimentales. Se basa en el entrenamiento de redes neuronales artificiales (*Artificial Neural Networks*) en el mapa logístico y la comprobación de su validez usando un modelo matemático de la misma naturaleza que los datos experimentales. Este algoritmo nos permite estudiar datos experimentales obtenidos de un cardiomiocito de rana.

- El éxito del algoritmo para el análisis de caoticidad con aprendizaje profundo en el conjunto de datos experimentales muestra la universalidad del mapa logístico y de algunos fenómenos dinámicos.

- La combinación del aprendizaje profundo con técnicas de preprocesado gráfico (*time series imaging*) nos permite generalizar los análisis de detección de caos. Es decir, entrenando una red neuronal artificial únicamente con datos del mapa logístico, podemos obtener análisis detallados de comportamiento dinámico (detección de regiones *shrimp-shaped* y zonas con multiestabilidad) en otros sistemas dinámicos discretos.

- Al utilizar los métodos *Markov Transition Field* y *Recurrence Plot* como técnicas de preprocesado gráfico obtenemos mejores resultados que al aplicar el método *Gramian Angular Field*.

- El aprendizaje profundo nos permite obtener el espectro completo de exponentes de Lyapunov de un sistema dinámico utilizando sólo series temporales cortas de una variable y ninguna otra información dinámica adicional. Hasta donde saben los autores, esto no es posible con ninguna otra técnica.

- Aunque usar aprendizaje profundo para aproximar exponentes de Lyapunov nos da buenos resultados tanto al utilizar datos no aleatorios como aleatorios para entrenar, la segunda estrategia funciona mejor ya que se proporciona mayor variabilidad dinámica en el entrenamiento.

- Se puede obtener un análisis biparamétrico detallado de clasificación dinámica (toros, hipercaos y caos) utilizando las aproximaciones de exponentes de Lyapunov proporcionadas por el aprendizaje profundo.

- El aprendizaje profundo proporciona buenos resultados en los análisis de sistemas dinámicos. Sin embargo, falla en regiones con dinámica compleja como las fronteras entre distintos regímenes dinámicos y las zonas con caos transitorio. Por ejemplo, esto ocurre en la detección de caos y en la aproximación de exponentes de Lyapunov.

- Una de las principales ventajas del uso del aprendizaje profundo para realizar análisis de sistemas dinámicos es el ahorro de tiempo. Por ejemplo, el aprendizaje profundo necesita menos de un 10% del tiempo utilizado por las técnicas clásicas para llevar a cabo un estudio triparamétrico de detección de caos o aproximar todos los exponentes de Lyapunov en un plano biparamétrico.

- Hemos particularizado el algoritmo BPGe (*Bregman Proximal Gradient with extrapolation*) para entrenar una red de tipo *Reservoir Computing* para clasificación binaria. Aunque incrementa ligeramente el tiempo real (*wall time*) unos minutos respecto a los optimizadores clásicos (SGD, RMSProp y Adam), proporciona mejores resultados y una convergencia más rápida.

- El patrón generalizado de tipo *tripod gait* (patrón bipartito) es el dominante para todos las redes de generadores centrales de patrones (*Central Pattern Generators*) de 6 a 9 neuronas con conectividad bipartita (red bipartita). Esto muestra una correlación entre la topología de la red y su dinámica.

- Los sistemas dinámicos, y en general las matemáticas, pueden ser utilizados para mejorar el aprendizaje profundo. Esto muestra que el aprendizaje profundo se puede beneficiar de las matemáticas y los sistemas dinámicos, y viceversa.

Como se ha mostrado en las conclusiones, en esta tesis doctoral se han obtenido resultados prometedores, sin embargo algunas cuestiones aún están sin resolver. A continuación resumimos algunos nuevos posibles temas de investigación derivados de este trabajo.

- ¿Qué estructuras dinámicas de una única neurona de tipo Hindmarsh-Rose se mantienen cuando acoplamos dos neuronas de este tipo?
- Arquitecturas de aprendizaje profundo capaces de tratar secuencias largas pueden ser útiles para obtener mejores resultados en regiones con dinámica compleja.
- ¿Se puede utilizar el aprendizaje profundo para obtener el espectro de exponentes de Lyapunov de series temporales experimentales?
- ¿Se pueden combinar técnicas de preprocesado gráfico y aprendizaje profundo para generalizar los análisis de comportamiento dinámico en sistemas dinámicos continuos?
- Hemos aproximado el espectro completo de exponentes de Lyapunov utilizando únicamente series temporales de una variable. Sería interesante analizar cómo afecta a los resultados considerar todas las variables del sistema.
- ¿Hasta dónde podemos llegar utilizando aprendizaje profundo para llevar a cabo análisis de sistemas dinámicos?
- Todas las ventajas del algoritmo BPGe sugieren que sería interesante adaptarlo a otras tareas y a otras arquitecturas de redes neuronales artificiales.
- ¿Se podría mejorar el entrenamiento de una red neuronal artificial usando otras familias de optimizadores?
- Al estudiar la posible correlación entre la topología de la red y su dinámica han aparecido algunos patrones con neuronas conectadas que están activadas al mismo tiempo. Esto parece contradecir el uso de sinapsis inhibitorias, por lo que es necesario profundizar en el estudio de dichos patrones.
- Un breve estudio de redes 3 coloreables no bipartitas ha mostrado que el patrón 3 coloreable es el predominante. Sin embargo, la dominancia es más débil que en el caso de los patrones bipartitos en redes bipartitas. ¿Por qué es más débil la predominancia en el primer caso?
- Después de estudiar la relación entre topología y dinámica en redes de hasta 9 neuronas parece natural estudiar redes con un mayor número de neuronas.
- Un tema interesante es la aplicación de técnicas de sistemas dinámicos para analizar la relación entre la arquitectura de las redes neuronales artificiales y cómo funcionan.

Bibliography

[1] R. Barrio, S. Ibáñez, J. A. Jover-Galtier, Á. Lozano, M. Á. Martínez, A. Mayora-Cebollero, C. Mayora-Cebollero, L. Pérez, S. Serrano, and R. Vigara, "Dynamics of excitable cells: Spike-adding phenomena in action," *SeMA Journal*, vol. 81, no. 1, pp. 113–146, 2024.

[2] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 7, p. 073146, 2023.

[3] C. Mayora-Cebollero, A. Mayora-Cebollero, Álvaro Lozano, and R. Barrio, "Full Lyapunov exponents spectrum with Deep Learning from single-variable time series," *Physica D: Nonlinear Phenomena*, vol. 472, p. 134510, 2025.

[4] Á. Lozano, R. Vigara, C. Mayora-Cebollero, and R. Barrio, "Dominant patterns in small directed bipartite networks: Ubiquitous generalized tripod gait," *Nonlinear Dynamics*, vol. 112, pp. 15549–15565, 2024.

[5] C. Mayora-Cebollero, F. H. Fenton, M. Halprin, C. Herndon, M. J. Toye, and R. Barrio, "Deep Learning for analyzing chaotic dynamics in biological time series: Insights from frog heart signals," *Preprint*, 2024.

[6] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, "Graphical preprocessing techniques to generalize dynamical behavior analysis with Deep Learning," *Preprint*, 2024.

[7] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, "Bregman Proximal Gradient with extrapolation to train a Reservoir Computing network," *Preprint*, 2024.

[8] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[9] J. L. Hindmarsh and R. Rose, "A model of neuronal bursting using three coupled first order differential equations," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 221, no. 1222, pp. 87–102, 1984.

[10] M. Storace, D. Linaro, and E. de Lange, "The Hindmarsh-Rose neuron model: Bifurcation analysis and piecewise-linear approximations," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 18, no. 3, 2008.

[11] A. Shilnikov and M. Kolomiets, "Methods of the qualitative theory for the Hindmarsh-Rose model: A case study – A tutorial," *International Journal of Bifurcation and Chaos*, vol. 18, no. 08, pp. 2141–2168, 2008.

[12] R. Barrio and A. Shilnikov, "Parameter-sweeping techniques for temporal dynamics of neuronal systems: Case study of Hindmarsh-Rose model," *The Journal of Mathematical Neuroscience*, vol. 1, pp. 1–22, 2011.

[13] R. Barrio, M. Á. Martínez, S. Serrano, and A. Shilnikov, “Macro- and micro-chaotic structures in the Hindmarsh-Rose model of bursting neurons,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 24, no. 2, 2014.

[14] E. M. Izhikevich, *Dynamical systems in neuroscience*. MIT Press, 2007.

[15] D. Terman, “Chaotic spikes arising from a model of bursting in excitable membranes,” *SIAM Journal on Applied Mathematics*, vol. 51, no. 5, pp. 1418–1450, 1991.

[16] H. Korn and P. Faure, “Is there chaos in the brain? II. Experimental evidence and related models,” *Comptes rendus biologies*, vol. 326, no. 9, pp. 787–840, 2003.

[17] Y. Hirata, M. Oku, and K. Aihara, “Chaos in neurons and its application: Perspective of chaos engineering,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 4, 2012.

[18] J. N. Weiss, A. Garfinkel, H. S. Karagueuzian, P.-S. Chen, and Z. Qu, “Early Afterdepolarizations and cardiac arrhythmias,” *Heart Rhythm*, vol. 7, no. 12, pp. 1891–1899, 2010.

[19] R. Barrio, S. Ibáñez, and L. Pérez, “Hindmarsh-Rose model: Close and far to the singular limit,” *Physics Letters A*, vol. 381, no. 6, pp. 597–603, 2017.

[20] R. Barrio, S. Ibáñez, and L. Pérez, “Homoclinic organization in fold/hom bursters: The Hindmarsh-Rose model,” *Chaos*, vol. 30, no. 5, p. 053132, 2019.

[21] R. Barrio, S. Ibáñez, L. Pérez, and S. Serrano, “Spike-adding structure in fold/hom bursters,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 83, p. 105100, 2020.

[22] R. Barrio, S. Ibáñez, and L. Pérez, “Exploring the geometry of the bifurcation sets in parameter space,” *Scientific Reports*, vol. 14, no. 1, p. 10900, 2024.

[23] R. Barrio, S. Ibáñez, and L. Pérez, “Homoclinic organization in the Hindmarsh-Rose model: A three parameter study,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 5, 2020.

[24] R. Gilmore and M. Lefranc, *The topology of chaos: Alice in Stretch and Squeezeland*. John Wiley & Sons, 2012.

[25] S. Serrano, M. Á. Martínez, and R. Barrio, “Order in chaos: Structure of chaotic invariant sets of square-wave neuron models,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 4, 2021.

[26] R. M. Ghigliazza and P. Holmes, “Minimal models of bursting neurons: How multiple currents, conductances, and timescales affect bifurcation diagrams,” *SIAM Journal on Applied Dynamical Systems*, vol. 3, no. 4, pp. 636–670, 2004.

[27] R. M. Ghigliazza and P. Holmes, “A minimal model of a Central Pattern Generator and motoneurons for insect locomotion,” *SIAM Journal on Applied Dynamical Systems*, vol. 3, no. 4, pp. 671–700, 2004.

[28] R. Barrio, Á. Lozano, M. Á. Martínez, M. Rodríguez, and S. Serrano, “Routes to tripod gait movement in hexapods,” *Neurocomputing*, vol. 461, pp. 679–695, 2021.

[29] R. Barrio, M. Á. Martínez, E. Pueyo, and S. Serrano, “Dynamical analysis of Early Afterdepolarization patterns in a biophysically detailed cardiac model,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 7, 2021.

[30] R. Barrio, M. Á. Martínez, S. Serrano, and E. Pueyo, "Dynamical mechanism for generation of arrhythmogenic Early Afterdepolarizations in cardiac myocytes: Insights from in silico electrophysiological models," *Physical Review E*, vol. 106, no. 2, p. 024402, 2022.

[31] F. Liu, X. Zhou, J. Cao, Z. Wang, H. Wang, and Y. Zhang, "Arrhythmias classification by integrating stacked bidirectional LSTM and two-dimensional CNN," in *Advances in Knowledge Discovery and Data Mining: 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part II 23*, pp. 136–149, Springer, 2019.

[32] G. Yan, S. Liang, Y. Zhang, and F. Liu, "Fusing Transformer model with temporal features for ECG heartbeat classification," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 898–905, IEEE, 2019.

[33] W. S. Lee and S. Flach, "Deep Learning of chaos classification," *Machine Learning: Science and Technology*, vol. 1, no. 4, p. 045019, 2020.

[34] N. Boullé, V. Dallas, Y. Nakatsukasa, and D. Samaddar, "Classification of chaotic time series with Deep Learning," *Physica D: Nonlinear Phenomena*, vol. 403, p. 132261, 2020.

[35] A. Celletti, C. Gales, V. Rodriguez-Fernandez, and M. Vasile, "Classification of regular and chaotic motions in Hamiltonian systems with Deep Learning," *Scientific Reports*, vol. 12, no. 1, p. 1890, 2022.

[36] 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*, pp. 95–98, IEEE, 2003.

[37] Y. Savitsky and 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*, pp. 298–301, IEEE, 2015.

[38] L. A. Dmitrieva, Y. A. Kuperin, N. M. Smetanin, and G. A. Chernykh, "Method of calculating Lyapunov exponents for time series using Artificial Neural Networks committees," in *2016 Days on Diffraction (DD)*, pp. 127–132, IEEE, 2016.

[39] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using Machine Learning to replicate chaotic attractors and calculate Lyapunov exponents from data," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 12, 2017.

[40] 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)*, pp. 1–4, IEEE, 2018.

[41] S. Bompas, B. Georgeot, and D. Guéry-Odelin, "Accuracy of neural networks for the simulation of chaotic dynamics: Precision of training data vs precision of the algorithm," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 11, 2020.

[42] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A Reservoir Computing approach," *Physical Review Letters*, vol. 120, no. 2, p. 024102, 2018.

[43] P.-R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics," *Neural Networks*, vol. 126, pp. 191–217, 2020.

[44] S. Shahi, C. D. Marcotte, C. J. Herndon, F. H. Fenton, Y. Shiferaw, and E. M. Cherry, “Long-time prediction of arrhythmic cardiac action potentials using Recurrent Neural Networks and Reservoir Computing,” *Frontiers in Physiology*, vol. 12, p. 734178, 2021.

[45] R. Chandra, S. Goyal, and R. Gupta, “Evaluation of Deep Learning models for multi-step ahead time series prediction,” *IEEE Access*, vol. 9, pp. 83105–83123, 2021.

[46] J. Shena, K. Kaloudis, C. Merkatas, and M. A. Sanjuán, “On the approximation of basins of attraction using Deep Neural Networks,” *arXiv preprint arXiv:2109.06564*, 2021.

[47] D. Valle, A. Wagemakers, A. Daza, and M. A. Sanjuán, “Characterization of fractal basins using deep Convolutional Neural Networks,” *International Journal of Bifurcation and Chaos*, vol. 32, no. 13, p. 2250200, 2022.

[48] M. Lellep, J. Prexl, M. Linkmann, and B. Eckhardt, “Using Machine Learning to predict extreme events in the Hénon map,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 1, 2020.

[49] S. H. Lim, L. T. Giorgini, W. Moon, and J. S. Wettlaufer, “Predicting critical transitions in multiscale dynamical systems using Reservoir Computing,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 12, 2020.

[50] M. Kalia, S. L. Brunton, H. G. Meijer, C. Brune, and J. N. Kutz, “Learning normal form Autoencoders for data-driven discovery of universal, parameter-dependent governing equations,” *arXiv preprint arXiv:2106.05102*, 2021.

[51] L.-W. Kong, H.-W. Fan, C. Grebogi, and Y.-C. Lai, “Machine Learning prediction of critical transition and system collapse,” *Physical Review Research*, vol. 3, no. 1, p. 013090, 2021.

[52] M. Ganaie, S. Ghosh, N. Mendola, M. Tanveer, and S. Jalan, “Identification of chimera using Machine Learning,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 6, 2020.

[53] S. T. da Silva, R. L. Viana, C. Batista, and A. M. Batista, “Prediction of chimera in coupled map networks by means of Deep Learning,” *Physica A: Statistical Mechanics and its Applications*, vol. 610, p. 128394, 2023.

[54] R. Barrio and S. Serrano, “A three-parametric study of the Lorenz model,” *Physica D: Nonlinear Phenomena*, vol. 229, no. 1, pp. 43–51, 2007.

[55] R. Barrio and S. Serrano, “Bounds for the chaotic region in the Lorenz model,” *Physica D: Nonlinear Phenomena*, vol. 238, no. 16, pp. 1615–1624, 2009.

[56] M. R. Gallas, M. R. Gallas, and J. A. Gallas, “Distribution of chaos and periodic spikes in a three-cell population model of cancer,” *The European Physical Journal Special Topics*, vol. 223, no. 11, pp. 2131–2144, 2014.

[57] R. Halfar, “Dynamical properties of Beeler-Reuter cardiac cell model with respect to stimulation parameters,” *International Journal of Computer Mathematics*, vol. 97, no. 1-2, pp. 498–507, 2020.

[58] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, “Determining Lyapunov exponents from a time series,” *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, 1985.

[59] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, “A practical method for calculating largest Lyapunov exponents from small data sets,” *Physica D: Nonlinear Phenomena*, vol. 65, no. 1, pp. 117–134, 1993.

[60] H. F. von Bremen, F. E. Udwadia, and W. Proskurowski, "An efficient QR based method for the computation of Lyapunov exponents," *Physica D: Nonlinear Phenomena*, vol. 101, no. 1, pp. 1–16, 1997.

[61] R. Barrio, "Theory and applications of the orthogonal fast Lyapunov indicator (OFLI and OFLI2) methods," *Chaos Detection and Predictability*, pp. 55–92, 2016.

[62] G. A. Gottwald and I. Melbourne, "The 0-1 test for chaos: A review," *Chaos Detection and Predictability*, pp. 221–247, 2016.

[63] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[64] M. Paluš, "Chaotic measures and real-world systems: Does the Lyapunov exponent always measure chaos?," in *Nonlinear Analysis of Physiological Data* (H. Kantz, J. Kurths, and G. Mayer-Kress, eds.), (Berlin, Heidelberg), pp. 49–66, Springer Berlin Heidelberg, 1998.

[65] J. E. Skinner, A. L. Goldberger, G. Mayer-Kress, and R. E. Ideker, "Chaos in the heart: Implications for clinical cardiology," *Bio/technology*, vol. 8, no. 11, pp. 1018–1024, 1990.

[66] A. Garfinkel, M. Spano, W. Ditto, and J. Weiss, "Controlling cardiac chaos," *Science (New York, N.Y.)*, vol. 257, pp. 1230–1235, 1992.

[67] N. Thakor, "Chaos in the heart: Signals and models," in *Proceedings of the 1998 2nd International Conference Biomedical Engineering Days*, pp. 11–18, 1998.

[68] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459–467, 1976.

[69] H. M. Hastings, F. H. Fenton, S. J. Evans, O. Hotomaroglu, J. Geetha, K. Gittelson, J. Nilson, and A. Garfinkel, "Alternans and the onset of ventricular fibrillation," *Physical Review E*, vol. 62, no. 3, pp. 4043–4048, 2000.

[70] N. Marwan and K. H. Kraemer, "Trends in recurrence analysis of dynamical systems," *The European Physical Journal Special Topics*, vol. 232, no. 1, pp. 5–27, 2023.

[71] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle, "Recurrence Plots of dynamical systems," *Europhysics Letters*, vol. 4, no. 9, pp. 973–977, 1987.

[72] D. Thakur, A. Mohan, G. Ambika, and C. Meena, "Machine Learning approach to detect dynamical states from recurrence measures," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 34, no. 4, 2024.

[73] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," in *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 3939–3945, 2015.

[74] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled Convolutional Neural Networks," in *Workshops at the twenty-ninth AAAI Conference on Artificial Intelligence*, 2015.

[75] R. C. Hilborn, *Chaos and nonlinear dynamics: An introduction for scientists and engineers*. Oxford University Press, 2000.

[76] M. Hénon, "A two-dimensional mapping with a strange attractor," *The Theory of Chaotic Attractors*, pp. 94–102, 2004.

[77] J. H. Argyris, G. Faust, and M. Haase, *An exploration of chaos: An introduction for natural scientists and engineers*. Texts on Computational Mechanics Volume VII, 1994.

[78] G. Grassi, F. L. Severance, and D. A. Miller, "Multi-wing hyperchaotic attractors from coupled Lorenz systems," *Chaos, Solitons & Fractals*, vol. 41, no. 1, pp. 284–291, 2009.

[79] T. Laurent and J. von Brecht, "A Recurrent Neural Network without chaos," in *International Conference on Learning Representations*, 2022.

[80] H. Banki-Koshki and S. A. Seyyedsalehi, "Complexity emerging from simplicity: Bifurcation analysis of the weights time series in a feedforward neural network," *Communications in Nonlinear Science and Numerical Simulation*, vol. 118, p. 107044, 2023.

[81] C. Curto and K. Morrison, "Graph rules for Recurrent Neural Network dynamics," *Notices of the American Mathematical Society*, vol. 70, no. 4, pp. 536–551, 2023.

[82] R. Engelken, "Gradient flossing: Improving gradient descent through dynamic control of jacobians," *Advances in Neural Information Processing Systems*, vol. 36, pp. 10412–10439, 2023.

[83] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[84] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[85] G. Hinton and T. Tieleman, "Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude," *Coursera: Neural Networks for Machine Learning*, vol. 4, pp. 26–31, 2012.

[86] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[87] N. Polson, J. G. Scott, and B. T. Willard, "Proximal algorithms in statistics and Machine Learning," *Statistical Science*, vol. 30, no. 4, pp. 559–581, 2015.

[88] X. Zhang, R. Barrio, M. Á. Martínez, H. Jiang, and L. Cheng, "Bregman Proximal Gradient algorithm with extrapolation for a class of nonconvex nonsmooth minimization problems," *IEEE Access*, vol. 7, pp. 126515–126529, 2019.

[89] R. Barrio, Á. Lozano, M. Rodríguez, and S. Serrano, "Numerical detection of patterns in CPGs: Gait patterns in insect movement," *Communications in Nonlinear Science and Numerical Simulation*, vol. 82, p. 105047, 2020.

[90] G. A. Pavlopoulos, P. I. Kontou, A. Pavlopoulou, C. Bouyioukos, E. Markou, and P. G. Bagos, "Bipartite graphs in systems biology and medicine: A survey of methods and applications," *GigaScience*, vol. 7, no. 4, p. giy014, 2018.

[91] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2022.

[92] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[93] F. Rosenblatt, "The Perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[94] D. H. Hubel, "Single unit activity in striate cortex of unrestrained cats," *The Journal of Physiology*, vol. 147, no. 2, pp. 226–238, 1959.

[95] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurons in the cat's striate cortex," *The Journal of Physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[96] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[97] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[98] J. Schmidhuber and S. Hochreiter, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[99] H. Jaeger, "The "echo state" approach to analysing and training Recurrent Neural Networks - with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, 2001.

[100] M. Cucchi, S. Abreu, G. Ciccone, D. Brunner, and H. Kleemann, "Hands-on Reservoir Computing: A tutorial for practical implementation," *Neuromorphic Computing and Engineering*, vol. 2, no. 3, p. 032002, 2022.

[101] M. Lukoševičius, "A practical guide to applying Echo State Networks," in *Neural Networks: Tricks of the Trade: Second Edition. Lecture Notes in Computer Science (LNCS, volume 7700). Springer Berlin, Heidelberg*, pp. 659–686, 2012.

[102] E. J. Doedel, "AUTO: A program for the automatic bifurcation analysis of autonomous systems," *Congressus Numerantium*, vol. 30, pp. 265–284, 1981.

[103] E. J. Doedel, R. Paffenroth, A. R. Champneys, T. F. Fairgrieve, Y. A. Kuznetsov, B. E. Oldeman, B. Sandstede, and X. J. Wang, "AUTO2000," *http://cmvl.cs.concordia.ca/auto*.

[104] D. Sato, L.-H. Xie, A. A. Sovari, D. X. Tran, N. Morita, F. Xie, H. Karagueuzian, A. Garfinkel, J. N. Weiss, and Z. Qu, "Synchronization of chaotic Early Afterdepolarizations in the genesis of cardiac arrhythmias," *Proceedings of the National Academy of Sciences*, vol. 106, no. 9, pp. 2983–2988, 2009.

[105] C.-h. Luo and Y. Rudy, "A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interaction.," *Circulation Research*, vol. 68, no. 6, pp. 1501–1526, 1991.

[106] D. Sato, L.-H. Xie, T. P. Nguyen, J. N. Weiss, and Z. Qu, "Irregularly appearing Early Afterdepolarizations in cardiac myocytes: Random fluctuations or dynamical chaos?," *Biophysical Journal*, vol. 99, no. 3, pp. 765–773, 2010.

[107] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance Deep Learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[108] T. Tél and Y.-C. Lai, "Chaotic transients in spatially extended systems," *Physics Reports*, vol. 460, no. 6, pp. 245–275, 2008.

[109] E. J. Doedel, B. Krauskopf, and H. M. Osinga, "Global invariant manifolds in the transition to preturbulence in the Lorenz system," *Indagationes Mathematicae*, vol. 22, no. 3, pp. 222–240, 2011.

[110] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[111] J. H. Halton, "Algorithm 247: Radical-inverse quasi-random point sequence," *Communications of the ACM*, vol. 7, no. 12, pp. 701–702, 1964.

[112] G. W. Beeler and H. Reuter, "Reconstruction of the action potential of ventricular myocardial fibres," *The Journal of Physiology*, vol. 268, no. 1, pp. 177–210, 1977.

[113] E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," in *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 285–289, 2000.

[114] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[115] A. Krizhevsky, "One weird trick for parallelizing Convolutional Neural Networks," *arXiv preprint arXiv:1404.5997*, 2014.

[116] S. Fraser and R. Kapral, "Analysis of flow hysteresis by a one-dimensional map," *Physical Review A*, vol. 25, no. 6, pp. 3223–3233, 1982.

[117] E. A. Celarier and R. Kapral, "Bistable limit cycle oscillations in chemical systems. I. Basins of attraction," *The Journal of Chemical Physics*, vol. 86, no. 6, pp. 3357–3365, 1987.

[118] E. Celarier and R. Kapral, "Bistable limit cycle oscillations in chemical systems. II. Mechanisms for noise-induced transitions," *The Journal of Chemical Physics*, vol. 86, no. 6, pp. 3366–3372, 1987.

[119] P. Gaspard, R. Kapral, and G. Nicolis, "Bifurcation phenomena near homoclinic systems: A two-parameter analysis," *Journal of Statistical Physics*, vol. 35, pp. 697–727, 1984.

[120] J. Bolte, S. Sabach, M. Teboulle, and Y. Vaisbourd, "First order methods beyond convexity and Lipschitz gradient continuity with applications to quadratic inverse problems," *SIAM Journal on Optimization*, vol. 28, no. 3, pp. 2131–2151, 2018.

[121] J. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, and X. Parra, "Human Activity Recognition using smartphones." UCI Machine Learning Repository, 2013. DOI: 10.24432/C54S4K.

[122] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for Human Activity Recognition using smartphones," in *The European Symposium on Artificial Neural Networks*, 2013.

Appendix

I. Published Articles: Impact Factors of the Journals and Thematic Areas

[1] R. Barrio, S. Ibáñez, J.A. Jover-Galtier, Á. Lozano, M.Á. Martínez, A. Mayora-Cebollero, C. Mayora-Cebollero, L. Pérez, S. Serrano, and R. Vigara, "Dynamics of excitable cells: Spike-adding phenomena in action", *SeMA Journal*, vol. 81, no. 1, pp. 113-146, 2024.
`doi:10.1007/s40324-023-00328-2`

- CiteScore 2023 (Scopus): 2.8
- Thematic area of the article: Applied Mathematics 248/635 (Q2)

[2] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 7, p. 073146, 2023.
`doi:10.1063/5.0143876`

- ⋆ Featured Article
- JCR 2023. Impact Factor: 2.7
- Thematic area of the article:
  - ○ Applied Mathematics 25/332 (Q1)
  - ○ Mathematical Physics 5/60 (Q1)

[3] C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, "Full Lyapunov exponents spectrum with Deep Learning from single-variable time series", *Physica D: Nonlinear Phenomena*, vol. 472, p. 134510, 2025.
`doi:10.1016/j.physd.2024.134510`

- JCR 2023. Impact Factor: 2.7
- Thematic area of the article: Applied Mathematics 25/332 (Q1)

[4] Á. Lozano, R. Vigara, C. Mayora-Cebollero, and R. Barrio, "Dominant patterns in small directed bipartite networks: Ubiquitous generalized tripod gait", *Nonlinear Dynamics*, vol. 112, pp. 15549-15565, 2024.
`doi:10.1007/s11071-024-09830-2`

- JCR 2023. Impact Factor: 5.2
- Thematic area of the article: Mechanical Engineering 22/183 (Q1)

II. Published Articles: The PhD Candidate's Contribution

In the published article **Dynamics of excitable cells: Spike-adding phenomena in action** (*SeMA Journal*) [1], the contributions of the PhD candidate have been to contextualize the work and select or design the appropriate figures to illustrate all the studied phenomena. The PhD candidate has also participated in the writing of the original draft and in its subsequent revisions.

The PhD candidate's contribution in the published article **Deep Learning for chaos detection** (*Chaos: An Interdisciplinary Journal of Nonlinear Science*) [2] includes the creation and selection of the data, the design of the architecture of the Deep Learning networks, the programming of the code used to train and test these networks, and the extraction of the results. Moreover, the PhD candidate has participated in the writing of the original draft and its subsequent revisions, as well as in the design of the figures.

In the published article **Full Lyapunov exponents spectrum with Deep Learning from single-variable time series** (*Physica D: Nonlinear Phenomena*) [3], the contributions of the PhD candidate are the creation and selection of the data, the programming of the code used to train and test the Deep Learning networks, and the extraction of the results. The PhD candidate has also participated in the writing of the original draft, in the review and editing process, and in the design of the visual representations.

The PhD candidate's contribution in the published article **Dominant patterns in small directed bipartite networks: Ubiquitous generalized tripod gait** (*Nonlinear Dynamics*) [4] has been to perform the analyses required to classify the obtained patterns into different equivalence classes and to extract the results. Moreover, the PhD candidate has participated in the revision and editing process.

Furthermore, the PhD candidate has had to learn to use PyTorch in order to implement the Deep Learning techniques. It has also had to learn how to perform spiking-counting sweeps and compute Lyapunov exponents with standard techniques.

III. Unpublished Articles: The PhD Candidate's Contribution

The PhD candidate's contribution in the unpublished article **Deep Learning for analyzing chaotic dynamics in biological time series: Insights from frog heart signals** (*Preprint*) [5] includes the design of the algorithm as well as its programming. Moreover, the PhD candidate has applied the algorithm to the experimental data, and has participated in the writing of the original draft and in the design of the visual representations.

In the unpublished article **Graphical preprocessing techniques to generalize dynamical behavior analysis with Deep Learning** (*Preprint*) [6], the contributions of the PhD candidate are the creation of the data, the design of the analyses, and the programming of the time series imaging methods and the code used to train and test the Deep Learning networks. The PhD candidate has also participated in the writing of the original draft and in the design of the figures.

The PhD candidate's contribution in the unpublished article **Bregman Proximal Gradient with extrapolation to train a Reservoir Computing network** (*Preprint*) [7] includes the reformulation of the Bregman Proximal Gradient with extrapolation algorithm and its implementation, the data selection, the programming of the Deep Learning code, and the extraction of the results. In addition, the PhD candidate has participated in the writing of the original draft and in the design of the figures.

IV. Curriculum Vitae of the PhD Candidate

Published Articles

- C. Mayora-Cebollero, A. Mayora-Cebollero, Á. Lozano, and R. Barrio, "Full Lyapunov exponents spectrum with Deep Learning from single-variable time series", *Physica D: Nonlinear Phenomena*, vol. 472, p. 134510, 2025.
doi: 10.1016/j.physd.2024.134510

- A. Mayora-Cebollero, J.A. Jover-Galtier, F. Drubi, S. Ibáñez, Á. Lozano, C. Mayora-Cebollero, and R. Barrio, "Almost synchronization phenomena in the two and three coupled Brusselator systems", *Physica D: Nonlinear Phenomena*, vol. 472, p. 134457, 2025.
doi: 10.1016/j.physd.2024.134457

- Á. Lozano, R. Vigara, C. Mayora-Cebollero, and R. Barrio, "Dominant patterns in small directed bipartite networks: Ubiquitous generalized tripod gait", *Nonlinear Dynamics*, vol. 112, pp. 15549-15565, 2024.
doi: 10.1007/s11071-024-09830-2

- R. Barrio, J.A. Jover-Galtier, A. Mayora-Cebollero, C. Mayora-Cebollero, and S. Serrano, "Synaptic dependence of dynamic regimes when coupling neural populations", *Physical Review E*, vol. 109, no. 1, p. 014301, 2024.
doi: 10.1103/PhysRevE.109.014301

- R. Barrio, S. Ibáñez, J.A. Jover-Galtier, Á. Lozano, M.Á. Martínez, A. Mayora-Cebollero, C. Mayora-Cebollero, L. Pérez, S. Serrano, and R. Vigara, "Dynamics of excitable cells: Spike-adding phenomena in action", *SeMA Journal*, vol. 81, pp. 113-146, 2024.
doi: 10.1007/s40324-023-00328-2

- R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep Learning for chaos detection", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 7, p. 073146, 2023.
doi: 10.1063/5.0143876

- F. Drubi, A. Mayora-Cebollero, C. Mayora-Cebollero, S. Ibáñez, J.A. Jover-Galtier, Á. Lozano, L. Pérez, and R. Barrio, "Connecting chaotic regions in the Coupled Brusselator System", *Chaos, Solitons & Fractals*, vol. 169, p. 113240, 2023.
doi: 10.1016/j.chaos.2023.113240

Grants

- Grant for research staff in training financed by Ministerio de Universidades de España. *Beneficiaria de una ayuda del Ministerio de Universidades para la Formación de Profesorado Universitario (FPU) correspondiente al año 2020*, Universidad de Zaragoza (Spain), 01/12/2021-19/09/2023

- Grant to participate in the Sixteenth International Conference Zaragoza-Pau on Mathematics and its Applications (Jaca, Spain), Universidad de Zaragoza (Spain), 07/09/2022-09/09/2022

Research Stays and Visits

- Research stay in Georgia Institute of Technology (Georgia Tech) under the supervision of Professor Flavio H. Fenton ("Artificial Intelligence and Machine Learning for Biological Applications"), Atlanta (Georgia, US), 10/05/2023-07/08/2023

- Research visit to the group of Professor Valeriy Makarov ("Acoplamiento de neuronas y posibles aplicaciones"), Madrid (Spain), 13/06/2022-20/06/2022

Conferences and Other Research Activities

- "Deep Learning to Quantify Chaos in a Dynamical System" (Invited talk), Seventeenth International Conference Zaragoza-Pau on Mathematics and its Applications (Jaca, Spain), 04/09/2024-06/09/2024
- Organizer with Jorge A. Jover-Galtier of the minisymposium "Dynamical Systems: Techniques and Applications", Seventeenth International Conference Zaragoza-Pau on Mathematics and its Applications (Jaca, Spain), 04/09/2024-06/09/2024
- "Deep Learning Chaoticity Analysis of Biological Time Series: Frog Heart Dynamics" (Poster), 13th European Conference on Mathematical and Theoretical Biology (Toledo, Spain), 22/07/2024-26/07/2024
- "Full Lyapunov Exponents Spectrum with Deep Learning from Single-Variable Time Series" (Contributed talk), Recent Advances in Dynamical Systems - GDM2024 (Santiago de Compostela, Spain), 08/07/2024-12/07/2024
- "Adaptive Coupling in Mean-Field Models of Neural Populations" (Poster), 2024 International Conference on Mathematical Neuroscience (Dublin, Ireland), 11/06/2024-14/06/2024
- "Las Matemáticas en la Vida Real" (Invited talk for scientific dissemination with Ana Mayora Cebollero), II Ciclo de Conferencias CIENCIA en la UNED 2024 (Fraga, Spain), 28/02/2024
- "Deep Learning for Chaos Detection in a Dynamical System" (Poster), DDays 2023 (Oviedo), 03/10/2023-06/10/2023
- "Dominance of Tripod Gait in Bipartite Networks" (Poster), 2023 International Physics of Living Systems (iPoLS) Network Annual Meeting (Atlanta, Georgia, US), 01/08/2023-04/08/2023
- Member of the Organizing Committee of 3rd International Workshop on Neurodynamics (Castro-Urdiales, Spain), 14/06/2023-17/06/2023
- "Deep Learning for Chaos Detection" (Poster), SIAM Conference on Applications of Dynamical Systems (Portland, Oregon, US), 14/05/2023-18/05/2023
- "Detectando Caos con Deep Learning" (Contributed talk), Seminario de Doctorado Rubio de Francia (Zaragoza, Spain), 31/01/2023
- "Chaos Detection using Deep Learning" (Poster), Dynamics Days US 2023 (Online), 09/01/2023-11/01/2023
- "Deep Learning, Sistemas Dinámicos y Aplicaciones" (Contributed talk with Sergio Serrano), Jornada IUIs-UZ Transformación Digital (Zaragoza, Spain), 02/12/2022
- "Deep Learning for Chaos Detection" (Contributed talk), Sixteenth International Conference Zaragoza-Pau on Mathematics and its Applications (Jaca, Spain), 07/09/2022-09/09/2022
- "Chaos Detection: From Lyapunov Exponents to Deep Learning" (Poster), XXVII Congreso de Ecuaciones Diferenciales y Aplicaciones (CEDYA) / XVII Congreso de Matemática Aplicada (CMA) (Zaragoza, Spain), 18/07/2022-22/07/2022
- "Using Deep Learning to Solve a Dynamical Systems Problem: Chaos Detection" (Creation and Teaching of Research Content in the Master in Mathematical Engineering of Universidad Complutense de Madrid with Roberto Barrio, Álvaro Lozano, Ana Mayora-Cebollero, Sergio Serrano, and Rubén Vigara), XVI Modelling Week (Madrid, Spain), 13/06/2022-17/06/2022

Attendance at Courses

- Introducción al Tratamiento y Representación de Datos con Origin Pro para Publicaciones Científicas, Escuela de Doctorado de la Universidad de Zaragoza (Online), 15/11/2024
- Cómo Preparar Materiales Docentes Adecuados para la Enseñanza no Presencial, CIFICE / Universidad de Zaragoza (Zaragoza, Spain), 18/10/2024-01/11/2024
- Inteligencia Artificial y Grandes Modelos de Lenguaje: Funcionamiento, Componentes Clave y Aplicaciones, Cursos Extraordinarios Universidad de Zaragoza (Zaragoza, Spain), 03/07/2024-05/07/2024
- EBRAINS Brain Simulation Workshop 2024, Basque Center for Applied Mathematics (Bilbao, Spain), 03/06/2024-07/06/2024
- Aspectos Básicos de ADD/Moodle, CIFICE / Universidad de Zaragoza (Online), 08/11/2023-22/11/2023
- Fitting Data with Dynamical Models: Ten Lessons on Mathematical Field Work, Centre de Recerca Matemàtica (Online), 24/10/2023-05/12/2023
- Introduction to AI with MATLAB, MathWorks (Online), 27/07/2023
- Using MATLAB with Python, MathWorks (Online), 25/05/2023
- Taller sobre la Elaboración del Plan de Investigación y Documento de Actividades (DAD) en Sigma y Gestión de la Producción Científica, Iberus Connect (Zaragoza, Spain), 10/02/2023
- G9 Live: Open Science Open Data, Charlesworth Knowledge (Online), 02/12/2022
- Las Funciones del y la Docente en el Contexto Universitario, CIFICE / Universidad de Zaragoza (Zaragoza, Spain), 17/10/2022-26/10/2022
- G9 Live: Planning for an Academic Conference, Charlesworth Knowledge (Online), 20/04/2022
- G9 Live: Effective Public Speaking, Charlesworth Knowledge (Online), 06/04/2022
- LMS Invited Lectures on the Mathematics of Deep Learning, Isaac Newton Institute for Mathematical Sciences - Cambridge (Online), 28/02/2022-04/03/2022
- Fundamentals of Accelerated Computing with CUDA C/C++, NVIDIA Deep Learning Institute (Online), 24/02/2022-25/02/2022
- Introducción a la Programación en MATLAB, Escuela de Doctorado de la Universidad de Zaragoza (Zaragoza, Spain), 17/01/2022-21/01/2022
- Intelligent Systems Research Centre Computational Neuroscience, Neurotechnology and Neuro-Inspired Artificial Intelligence Autumn School, Ulster University (Online), 25/10/2021-29/10/2021
- Advanced Course on Mathematical Aspects of Algorithmic Learning and Deep Neural Networks, Centre de Recerca Matemàtica / BGSMath (Online), 05/10/2021-18/11/2021

Teaching Support Tasks (Required Activity of the FPU Grant)

- Matemáticas (20h), Grado en Geología, Faculty of Sciences, Universidad de Zaragoza (Zaragoza, Spain), 2022-2023

- Modelización Matemática (40h), Grado en Matemáticas / Grado en Matemáticas-Ingeniería Informática / Grado en Física-Matemáticas, Faculty of Sciences, Universidad de Zaragoza (Zaragoza, Spain), 2022-2023

University Teaching

- Epidemiología y Bioestadística (68h), Grado en Veterinaria, Faculty of Veterinary, Universidad de Zaragoza (Zaragoza, Spain), 2024-2025

- Ciencias Básicas para Veterinaria (100.6h), Grado en Veterinaria, Faculty of Veterinary, Universidad de Zaragoza (Zaragoza, Spain), 2024-2025

- Matemáticas III (48h), Grado en Ingeniería Electrónica y Automática, School of Engineering and Architecture, Universidad de Zaragoza (Zaragoza, Spain), 2023-2024

- Epidemiología y Bioestadística (62h), Grado en Veterinaria, Faculty of Veterinary, Universidad de Zaragoza (Zaragoza, Spain), 2023-2024

- Ciencias Básicas para Veterinaria (84h), Grado en Veterinaria, Faculty of Veterinary, Universidad de Zaragoza (Zaragoza, Spain), 2023-2024