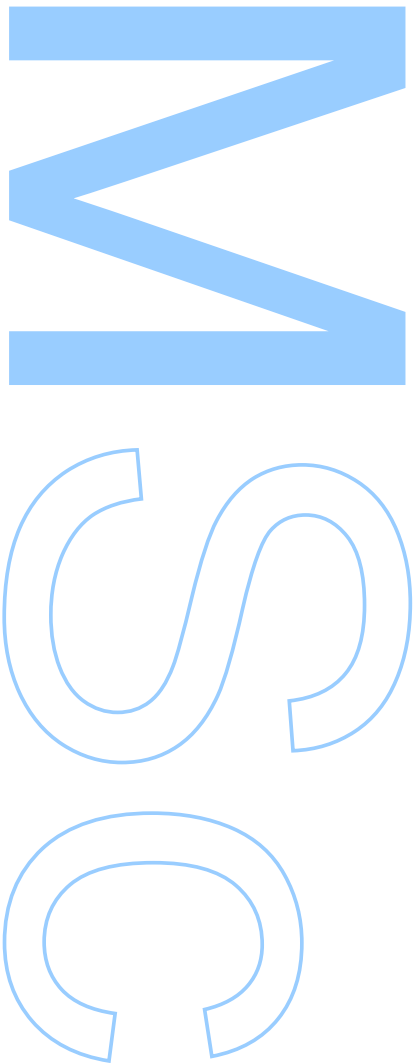
Harnessing Nonlinear Systems for Neuromorphic Computing Solutions

Juan Silva
Mestrado em Engenharia Física
Departamento de Física e Astronomia
2023

Orientador
Dr. Nuno A. Silva, Professor Auxiliar Convidado, Faculdade de Ciências

Coorientador
Dra. Catarina Dias, Professor Auxiliar, Faculdade de Ciências




U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Acknowledgements

A new stage is coming to an end in which many individuals contributed to its achievement. My deepest gratitude goes to Nuno Silva, my supervisor, for sharing his expertise and guiding me through this project. I am grateful to you for your understanding when I was not the best student and for your assistance with all the different problems that arose up. I also wish to take this opportunity to thank you for giving me the chance to participate in the working group at the Centre for Applied Photonics of INESCTEC and attempt to help with the work that is being done therem and also want to express my gratitude to the researchers that helped and advised me during this project. Last but not least, I would like to thank my co-supervisor Catarina Dias, who, even though things did not go according to plan, was there to support and help as much as possible.

The friends who have traveled with me on this journey, both those I have made here and those who are far away, deserve a special thank you. From the times spent at university, the late-night study sessions and the times I got in the way in the computer room, to the coffees, parties we attended and trips to the hospital, you made this journey more enjoyable.

Finally, I want to express my deepest appreciation to my family for their unconditional support. It was not an easy decision at first, due to a wide range of circumstances, but everyone's effort and support made this journey worthwhile. None of this would have been possible without them.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Física e Astronomia

MSc. Engineering Physics

Harnessing Nonlinear Systems for Neuromorphic Computing Solutions

by Juan SILVA

As technology continues to evolve at an astonishing rate, current computing paradigms are no longer effective in accommodating these innovations. In this sense, new computing architectures are being investigated to accommodate current needs. Neuromorphic computing, which simulates characteristics of the human brain, is an interesting candidate for this research.

Extreme Learning Machines (ELM) provide a highly versatile framework for streamlined training and deployment in the realm of neuromorphic computing applications. In essence, they can be characterized as a network of hidden neurons with randomly fixed weights and biases, producing intricate behavior in response to a given input, making the architecture appealing for physical implementations and thus suitable for neuromorphic computing solutions. In this context, this project aims to explore distinct physical systems that can be utilized as reservoir in physical implementations of ELMs.

We start by constructing a model using a nonlinear oscillator model, the Toda Lattice, with the aim of understanding the potential and innerworkings of ELMs. This is achieved by submitting the model to several numerical simulations and analyzing its performance on common machine learning tasks, such as regression and classification. Additionally, we apply metrics for the nonlinear dynamics of the system to understand the relation between the nonlinear behavior and the performance of the system.

We then delve into a particular field, of optical ELMs, using as inspiration a previously developed optoelectronic machine to create a transparent and versatile framework focused on an all-optical version of that machine. The goal is to address some of the challenges posed by the physical implementations of these machines, particularly in the

context of studying the application of constraints to the weights in an ELM. Finally, we conclude by discussing some of the current challenges and proposing ways to continue our work.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto
Departamento de Física e Astronomia

Mestrado Integrado em Engenharia Física

Titulo da Tese em Portugês

por Juan SILVA

A medida que a tecnologia continua a evoluir a um ritmo espantoso, os paradigmas computacionais actuais deixam de ser eficazes para acomodar estas inovações. Neste sentido, novas arquitecturas de computação estão a ser investigadas para dar resposta às necessidades actuais. A computação neuromórfica, uma arquitetura que simula as características do cérebro humano, mostra-se como um candidato interessante para esta pesquisa.

As *Extreme Learning Machines* (ELM) constituem uma estrutura altamente versátil e simplificada para a formação e implementação de aplicações no domínio da computação neuromórfica. Na sua essência, podem ser caracterizadas como uma rede de neurônios ocultos com pesos e enviesamentos fixados aleatoriamente, produzindo um comportamento complexo em resposta a uma determinada entrada, tornando esta arquitetura apelativa para implementações físicas e, portanto, adequada para soluções de computação neuromórfica. Neste contexto, este projeto visa explorar distintos sistemas físicos que podem ser utilizados como reservatório em implementações físicas de ELMs.

Começamos por construir um modelo com base num sistema de osciladores não linear, a *Toda Lattice*, com o objetivo de compreender o potencial e o funcionamento interno das ELMs. Para tal, o modelo é submetido a diversas simulações numéricas e é analisado o seu desempenho em tarefas comuns de *machine learning*, como regressão e classificação. Além disso, algumas métricas relacionadas as dinâmicas não lineares de sistemas para compreender a relação entre o comportamento não linear e o desempenho do sistema nessas situações.

Em seguida, mergulhamos num domínio mais específico, o das ELM óticas, utilizando como inspiração uma máquina optoelectrónica previamente desenvolvida, de modo a

criar uma estrutura transparente e versátil centrada numa versão totalmente ótica dessa mesma máquina. O objetivo passa por abordar alguns dos desafios colocados pelas implementações físicas destas máquinas, em particular no contexto do estudo da aplicação de restrições aos pesos de uma ELM. Finalmente, concluímos discutindo alguns dos desafios actuais e propondo formas de continuar o nosso trabalho.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Throughout the course of human history, technology has played a critical role in shaping the world and advancing human progress. From the earliest tools used by our ancestors to the latest innovations in artificial intelligence, technology has constantly evolved and had a profound impact on the way we live and interact with the world. The evolution of computers is one of the most significant technological advancements of the past century. Starting from room-sized machines that required a team of experts to operate, computers have undergone a remarkable transformation. Today, most of our computers are portable devices that can fit in the palm of our hand and are used by people of all ages and backgrounds. The widespread availability of computers has revolutionized the way we communicate, work, and access information. As with many technologies, they have also become an integral part of modern life with a profound impact on society as a whole.

At the scientific and engineering level, the development of electronics, and micro and nanotechnology has been instrumental in driving the evolution of computing since the first day. These advancements have allowed for the miniaturization of components, making computers smaller, more efficient, and accessible to a wider range of people. In particular, they were also fundamental in increasing the computational power of single processors, as empirically translated into Moore's Law, which we will discuss later in this chapter.

But as the requirement for computing capacity grows in industries like data processing, robotics, and artificial intelligence, the shortcomings of these technologies become more apparent. In particular, with the pursuit of miniaturization towards higher processing speeds and efficient wider architectures, electronic chips are now rapidly approaching a mesoscopic scale with few atoms and electrons per transistor, making them less robust

to noise. A possible workaround to such a critical obstacle may reside in exploring new computing architectures and paradigms.

In recent years, the field of machine learning has risen at the vanguard of technological innovation, enabling significant advances across a wide range of domains making use of optimization paradigms to deploy data-driven solutions. Inspired by the amazing plasticity of the human brain, machine learning algorithms have transformed scientific and engineering fields such as image identification, natural language processing, and predictive analytics. However, the complexity and volume of data have continued to rise at a rate that commonly overflows the computing capacities of conventional architectures. In this context, analog computing is now making a comeback as a possible option for increased computational efficiency and scalability of such algorithms, deploying them as a unified software-hardware solution.

Overall, this thesis is motivated by these pressing challenges and aims to seek solutions by exploring possible hardware implementations of extreme learning machines, one of the most promising and versatile approaches to machine learning. First, we will investigate the complexities of machine learning in order to gain a comprehensive understanding of its inner workings. Then, we will focus on extreme learning machines to investigate the potential of some physical nonlinear systems as a viable platform for neuromorphic computing, offering enhanced computational efficiency and scalability. By harnessing these physical systems, we aspire to present novel opportunities for the field of computing and unlock novel trends for faster and energy-efficient neuromorphic computing solutions.

1.1 Current Computing Paradigms and Limitations

Over the last 70 years, the field of computing has witnessed remarkable advancements that have propelled human progress to unprecedented heights. The exponential growth of data generation, coupled with the increasing demands for processing power, has highlighted the need for novel computing paradigms. Traditional computing architectures, such as the widely adopted Von Neumann paradigm have reached a computational plateau and face significant challenges in meeting such requirements. But to properly understand the constraints of the existing computing landscape and explore potential alternatives, it is necessary to delve into the history and evolution of general-purpose computers and understand what has allowed the current paradigms to endure for so long.

The year 1945 was a breakthrough point in computer history. The introduction of the Electronic Numerical Integrator and Computer (ENIAC) [1], a pioneering programmable computer, stunned the world with its enormous proportions and astounding capabilities. However, physical adjustments were necessary to program the machine, limiting its ability to move quickly and effectively.

Recognizing the need for more streamlined and efficient operations, John Von Neumann introduced a groundbreaking computer architecture in the same year, during the development of the Electronic Discrete Variable Automatic Computer (EDVAC), a subsequent project to ENIAC. This architectural framework [2, 3] again revolutionized the field of computing by optimizing the interplay between the central processing unit (CPU) and memory. This concept of storing both program instructions and data in a shared memory unit became the foundation for modern general-purpose computers, establishing it as the dominant computing paradigm over the last seven decades. Note that at that point in history, alternative computing paradigms, such as analog computing, were already known to offer advantages in terms of efficiency, speed and capacity, but struggled to match the Von Neumann paradigm capabilities and versatility in general-purpose computing. Ultimately, they were unable to keep pace with the rapid advancements of this architectural paradigm.

The superiority was further solidified with the fast evolution of electronics. First, the creation of the transistor in 1947 [4] and the subsequent emergence of the integrated circuits in 1949 [5], have taken the prevailing architecture to new heights. The transistor revolutionized electronic devices, offering greater reliability, miniaturization, and power efficiency. Integrated circuits, on the other hand, combined multiple transistors into a single chip, enabling even more compact and powerful computing systems. These advancements not only solidified its position but also paved the way for widespread adoption across diverse computing applications.

The remarkable trajectory that electronics was following led to a famous observation of Intel cofounder Gordon Moore in 1965. Later known as the Moore's law, Moore noted that the number of transistors on an integrated circuit doubled every two years [6, 7]. The reduction of transistor size has been one of the key pillars supporting this rule: transistors grown cheaper, quicker, and more energy-efficient as they got smaller.

Nevertheless, manufacturing becomes increasingly challenging as transistors get smaller and more densely packed. The cost of building a facility to create and develop microchips

has thus risen exponentially, and problems like power consumption, heat dissipation, and even quantum effects have become more significant. Thus, we cannot rely on the law of Moore to hold true in the near term [8–10]. Besides, several issues with the current architecture have come to light as a result of this problem [11]. For example, the *big data* era has reemerged the old data scalability issues in the Von Neumann architecture. Indeed, the Von Neumann bottleneck [12] is again a limiting factor, as the reliance on a shared memory unit for both data and program instructions limits the pace at which data could be transmitted between the CPU and memory and thus the total system performance.

These restrictions have spurred a resurgence of interest in previously discarded alternative computer platforms, with neuromorphic computing being one of the more promising concepts.

1.2 The brain as a computing machine

Neuromorphic computing has a unique architectural style, which is the source of its particular attraction. This computer architecture has structure and function inspired by the brain and has been designed to work in a way that mimics the functioning of neurons and synapses in the brain [13], allowing it to process and store information in a much more efficient and effective way than traditional computers.

To put things into perspective, the human brain is perhaps the most sophisticated and complex computing device on the planet [14–16]. It can perform a wide range of tasks, from simple reflexes to complex processes such as decision-making and problem-solving, using only 20 watts of power, which is equivalent to the energy consumption of a small lightbulb. For reference, modern high-performance computers require hundreds of watts to run. Another point that draws a lot of attention is its rapid learning capacity, that is, the brain needs a few examples to learn the most diverse tasks, as well as being able to store all the knowledge it gains. This is possible because the brain has the ability to change and reorganize itself in order to adapt and learn new information, known as neuroplasticity, which also helps to recover from injury or disease. It is made up of approximately 86 billion neurons interconnected by 10^{15} synapses. To have a notion, each neuron can have approximately 10^4 connections to other neurons through synapses [15–17]. These connections allow information to be transmitted rapidly and accurately through the brain, enabling it to perform complex processes such as decision-making, pattern recognition

and problem-solving. Given these (and many other) extraordinary talents, it is not surprising that researchers have been for long motivated to create a computer machine that can mimic the human brain.

Thus, as seen in figure 1.1 this novel design differs significantly from the conventional paradigm in a variety of ways. Neuromorphic computers are artificial neural networks constructed from artificial synapses and neurons. The organization and parameters of these components dictate the programming of the computer and govern both its memory and processing capabilities. In contrast to traditional computing systems, this paradigm integrates processing and memory elements on a unified architecture, operating in a distributed fashion without a clear distinction between them [18, 19]. This flexibility enables each processing part to function as a memory element and vice versa, resulting in high parallel processing. In addition, and unlike the sequential processing in Von Neumann architecture, neuromorphic systems allow synapses and neurons to operate simultaneously.

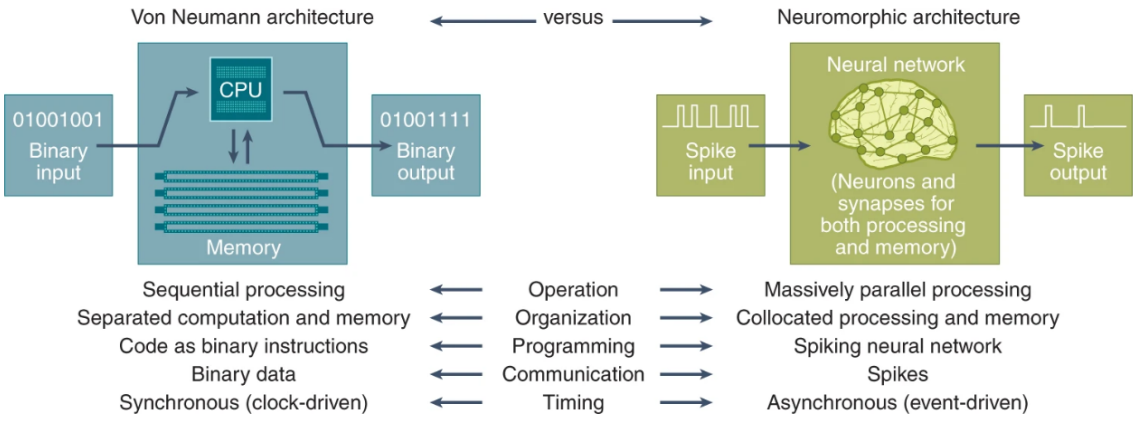


FIGURE 1.1: Contrast between Von Neumann paradigm and neuromorphic computing. Figure reproduced from [18].

From a distinct perspective, the integration of processing and memory elements also enables the creation of compact and lightweight computing systems by integrating them into a single physical device, overcoming the limitations of the Von Neumann bottleneck. This reduction in components not only improves energy efficiency but also holds the potential to reduce the carbon footprint. These inherent features make neuromorphic computing intrinsically scalable, allowing the combination of individual neuromorphic chips to form larger networks capable of handling more complex tasks.

All of these characteristics, as well as the way this computing paradigm is structured, make it a perfect foundation for many of the cutting-edge artificial intelligence and machine learning applications that are in great demand today.

1.3 Machine Learning as a computing paradigm

Machine learning (ML) is a sub-field of artificial intelligence (AI) that focuses on developing models and algorithms that let computers learn and anticipate the future or make decisions on their own [20]. In conceptual terms, it gives computers the ability to unravel and analyze enormous volumes of data, spot trends, and derive insightful knowledge, which enhances decision-making, automation, and problem-solving abilities.

The roots of machine learning can be traced back to the 1950s when researchers began exploring ways to create computational models that could mimic human intelligence. One influential contribution was the perceptron model by Frank Rosenblatt [21], which can be considered the precursor of nowadays neural networks. Nevertheless, despite considerable advancements in the years that followed, lack of computational power, inadequate datasets, and challenges in algorithmic optimization led to a period of relative obscurity for machine learning, now commonly referred to as the "AI winter" [22]. It was at the end of the 1990s that this field started to experience radical growth. Innovations in technology made the computational power more affordable and more diverse datasets facilitated the training of models. This subsequently led to an increase in performance and deployment of models capable of handling more complex tasks. From models capable of winning chess matches [23], to bots capable of generate images from simple text prompts [24, 25], significant milestones were achieved during the years, attracting more attention to the field and expanding its reach to various areas of study.

Over the years, researchers empirically realized that the performance of the models was closely related to their density and complexity and began developing larger and more robust models, requiring larger datasets for training. This observation played a significant role in motivating and understanding models with multiple layers of interconnected neurons [26, 27], which now we call deep learning.

An important breakthrough appeared in 2012, when a group of researchers from Google Brain won the ImageNet Large-Scale Visual Recognition Challenge [28]. Their network architecture, consisting of eight layers and approximately 62 million learnable parameters

trained on 1.2 million samples, demonstrated unprecedented accuracy in image detection, highlighting the power of deep learning in extracting meaningful features from vast amounts of data. As a result, a multitude of different and innovative works developed [29, 30], surpassing prior restrictions and extending the potential in this sector. The exponential expansion of model parameters has reached unprecedented magnitudes, as seen in the figure 1.2.

FIGURE 1.2: Trend in the number of parameters in machine learning models. Figure from [31].

But nowadays, we are immersed in the era of large-scale models, where size and complexity have grown to an almost impracticable point. For example, GPT-3 [32] was released in 2020 with 175 billion trainable parameters, and GPT-4 [33] in 2023 as an estimated 170 trillion parameters. In spite of their performance, important computational obstacles have also become clearer in this process, with concerns regarding power and memory being particularly relevant.

Some technological developments have arisen in an attempt to soften these challenges. The adoption of new kinds of hardware such as Graphical Processing Units (GPUs), enabled parallel computation, lowering processing costs for deeper models. Pre-trained models have gained popularity as a way to reduce computational strain by utilizing already-existing information. Additionally, cloud computing has become a useful tool for easing memory-related difficulties.

But perhaps more important towards the long term, a large amount of effort has also focused on the physical implementation of large scale models as a possible workaround to these bottlenecks. Neverthless, we shall note that the most typical models such as deep

learning ones, are unsuitable for this approach. Indeed, fine-tuning a physical model with millions or billions of parameters requires fabrication and versatility capacities which are usually unavailable.

In this context, two types of models have recently emerged as powerful solutions: extreme learning machines (ELM) [34–37] and reservoir computing (RC) [38–41], with respective architectures represented in figure 1.3. These methods share some similarities in structure and operation, as both rely on the projection of the information on the input space onto an intermediate space of higher dimensionality, using a single hidden layer commonly referred as the reservoir. In both, the training process occurs exclusively at the end of the reservoir layer, which can be performed by a simple linear optimization procedure using a given loss function. The parameters of the models are randomly initialized and remain fixed throughout training. This strongly simplifies the learning process while also contributes to computational efficiency, being possible to demonstrate that they can achieve similar results in generalization performance as other kind of models. Distinguishing them, we shall note that while ELMs belong to the family of feed-forward neural networks [42], which means the information only flows in one direction, RC falls into the category of recurrent neural networks [43], making them able to retain memory of past inputs and exhibit dynamic behavior over time.

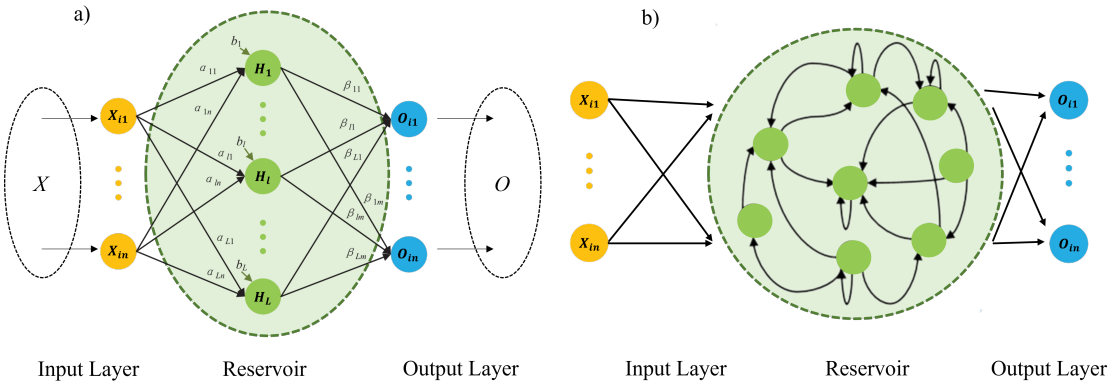


FIGURE 1.3: Visual representation showcasing the architecture of an ELM (a) and of a RC (b) model. Figures adapted from [44].

Other computing paradigms have been explored for hardware implementations [45, 46], but the unique properties make these models more appealing for physical implementations in the context of neuromorphic computing. Driven by this motivation, this dissertation aims to investigate the capabilities of different physical systems as potential reservoirs for implementing these models.

1.4 Roadmap of the thesis

The goal of this thesis is to explore different physical systems that can be utilized as reservoirs for the physical implementations of extreme learning machines within the field of neuromorphic computing. Building on previous work developed at INESC TEC, where the team has explored the implementation of an optoelectronic setup as an Extreme Learning Machine [47], this thesis aims at two strategic objectives:

1. Explore physical systems with some degree of tunability, in order to further understand the potential and innerworkings of Extreme Learning Machines;

2. Explore suitable tools to perform training of the physical systems in a transparent manner;

With these in mind, we establish two operational objectives:

1. Explore electronic circuits as a potential toy model for deploying an Extreme Learning Machine with tunable parameters;

2. Develop a transparent and versatile training workflow and library based on Pytorch to train and optimize an Extreme Learning Machine using a physical system, in particular, focusing on an all-optical version of the previously developed optoelectronic machine [47].

Figure 1.4 illustrates the roadmap of the work performed. In this way, the thesis is structured as follows: in this chapter we began by providing a historical overview of the predominant von Neumann computing paradigm. We then introduce the motivation for alternative approaches to traditional computing architectures and the emergence of neuromorphic computing as a promising solution. Subsequently, we proceeded to present an evolution of the field of machine learning, highlighting some of the challenges and limitations encountered. We highlighted some possible solutions, with particular emphasis on ELMs, establishing the subject of the thesis and introducing our work.

In chapter 2 we delve into the theoretical foundations of machine learning, discussing some of the fundamental principles and methodologies. We then shift focus to the mathematical aspects of extreme learning machines and reservoir computing. Toward the end of the chapter, we analyze various aspects that contribute to the effectiveness of a reservoir for both ELMs and reservoir computing.

Leveraging on the framework introduced, chapter 3 utilizes a nonlinear oscillator model, the Toda Lattice, and explores its potential as a reservoir for ELM/RC implementations. We begin by introducing the system and its relevant characteristics, highlighting why it is a suitable candidate for our study due to its tunability. Then we conduct various numerical simulations in order to evaluate the system performance in different tasks, such as regression and classification. Additionally, we introduce some metrics for the nonlinear dynamics of the system, in order to better relate the performance with the nonlinearity of its behavior.

In chapter 4, we change direction and delve into the field of optical extreme learning machines. We begin by providing an overview of this field of study and conducting a review of the current state of the art in OELMs. Next, we build upon a previously developed optoelectronic machine to create a transparent and versatile framework focused on an all-optical version of that machine. Our primary aim is to address some of the challenges posed by the physical implementations of these machines, particularly in the context of studying the application of constraints to the weights in an ELM. Finally, we conclude by presenting the simulation results we have obtained and reflecting on the overall work.

In chapter 5, conclusions and future work perspectives are presented.

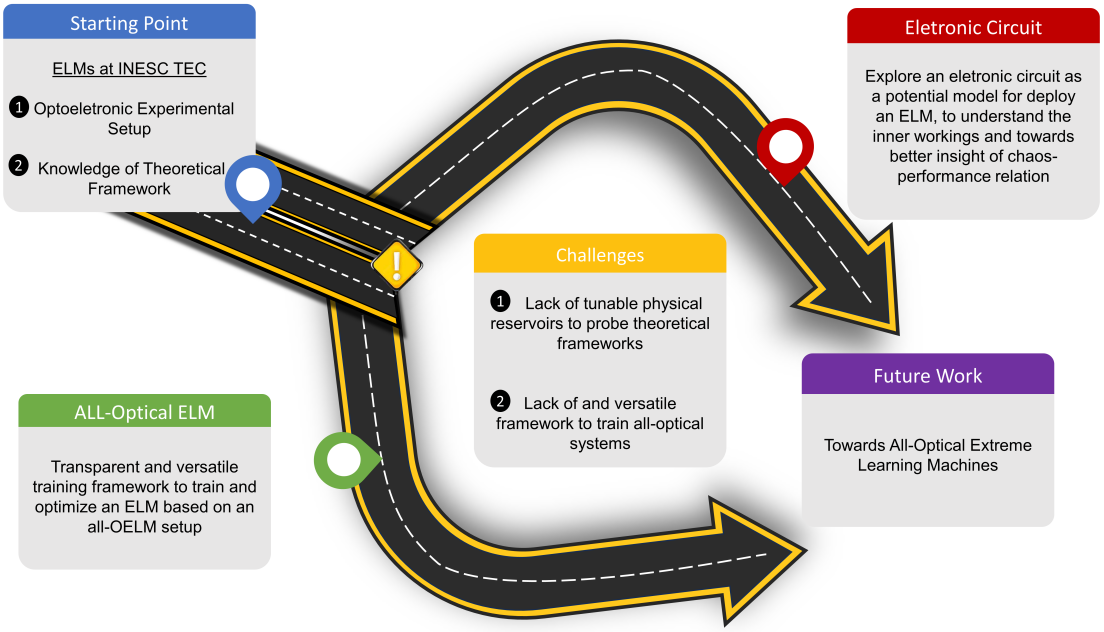


FIGURE 1.4: Roadmap of the thesis.

1.5 Thesis Outputs

In the elaboration of this thesis, the author has contributed with 2 poster presentations.

Journal Publications

1. *"All-optical Computing Framework Based on Extreme Learning Machines"* - Vicente Rocha, Tiago Ferreira, Juan Silva, Duarte Silva, Nuno A. Silva, in preparation

Poster Presentations

(a) *"Physical nonlinear systems for neuromorphic computing"* - Juan Silva, Vicente Rocha, Tiago Ferreira, Catarina Dias, Nuno A. Silva. IJUP2023 - 16th Young Researcher Meeting of University of Porto. (2023)

(b) *"Towards All-Optical Computing with Extreme Learning Machines"* - Juan Silva, Vicente Rocha, Tiago Ferreira, Catarina Dias, Nuno A. Silva. DCE2023 - 5th Doctoral Congress in Engineering. (2023)

Chapter 2

Framework of Extreme Learning Machines and Reservoir Computing

As discussed in the previous chapter, the field of machine learning is evolving at a fast pace, attracting significant attention. Models are becoming larger and more robust, showcasing increasing capabilities. However, these advancements often come at a price. The scale of these models is reaching absurd proportions, accompanied by an unprecedented surge in the number of trainable parameters. This puts a strain on our current computational and energy resources, making it challenging to implement physical versions of these models, which will necessitate the fine-tuning of millions of parameters.

Recognizing that the exponential growth of trainable parameters is a key problem, efforts have been made to reduce this variable. New machine learning models such as extreme learning machines (ELMs) and reservoir computing (RC) have emerged as a result. These approaches utilize a *reservoir* as a hidden layer, comprising random untrained parameters and non-linear dynamics that generate a complex response to a given input. The only requirement is to train the output layer to perform the desired task. These methods have demonstrated comparable performance to traditional approaches while offering significant benefits. They reduce the time required for training these models and alleviate the computational demands, resulting in decreased energy consumption.

In this chapter, we will delve deeper into the theory behind various machine learning approaches, explaining the most important concepts and methods of this topic. Subsequently, we will provide a theoretical context of ELMs and RC, in order to address some of the key characteristics that reservoirs must possess for the effective implementation of these methods and lay the foundations for our work.

2.1 Machine Learning: how can we teach a machine to solve problems?

Whenever we try to introduce a new concept, for example, explaining the differences between a normal car and a sports one, it is easier to provide some examples of the case rather than giving a specific formulation that defines both cases. In the same way, instead of codifying the knowledge into a set of logic operations for a computer, machine learning tries to learn meaningful patterns and relationships from examples and observations. As Arthur Samuel properly defined, "machine learning is the field of study that gives computers the ability to learn without being explicitly programmed" [20].

The rapid expansion of this field makes it useful in a diverse array of contexts, aligning with the broader landscape of Artificial Intelligence (AI). Namely, ML aims to be used in specific applications, such as: dynamic environments when there is constantly new data arriving; complex problems that traditional techniques find hard to solve or difficult to find some optimal solution; or scenarios with a multitude of adjustable parameters or when to search and find about trends and patterns in large datasets. Consequently, this field has the capacity to be applied across different domains of work. Its applications range from fields as diverse as meteorology, where it is used to forecast weather patterns over a month, to medical applications, where it helps with cancer diagnosis, and to the creation of automated intelligent agents that can help consumers. This capability to learn and adapt from experience is at the core of both machine learning and AI.

AI, as a broader research field, is focused on creating machines that simulate human brain-like capabilities [48]. This concept applies to machines that can execute human-like tasks, including problem-solving and learning. Consequently, ML emerges as a vital subset of AI, dealing specifically with the ability of machines

to learn and improve from experience. The large number of algorithms that exist nowadays led to the categorization of ML into different types, depending on how the model is built and trained. Some of these foundational types of ML include: supervised learning - if the model is built under "human supervision", unsupervised learning - where the algorithm is allowed to learn without explicit supervision, or reinforcement learning - where we make the model learn through interactions with an environment. These three types provide a foundational understanding of the field and have a more comprehensive overview in table 2.1, but the landscape is more intricate, including other types like semi-supervised learning, online learning, and batch learning, each designed to tackle specific scenarios.

Indeed, we can categorize the vast number of ML models under some rules: as we already pointed out, whether they require or not human supervision; work by detecting trends in data and making predictions, or are built by comparing new data to old data (model-based *vs* instance-based learning); or it can learn more with new data arriving (online *vs* batch learning) [47].

Depending on the learning task, the field offers a plenty of models, each combining different variants and specifications. Figure 2.1 showcases some of the existing algorithms.

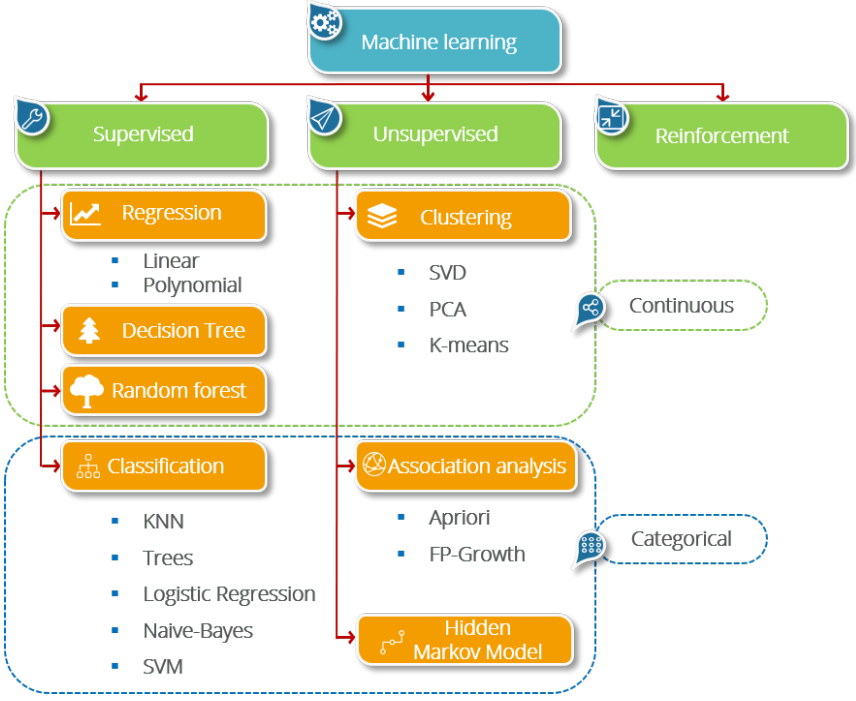


FIGURE 2.1: Some machine learning algorithms. Figure taken from [49]

TABLE 2.1: Overview of some types of machine learning.

Type	Description
Supervised Learning	It is a type of ML where we train an algorithm by providing input features and their corresponding output variables. The objective is for the machine to learn a mapping between these two and, with the highest possible accuracy, apply this mapping to predict outputs for unseen data. There are two different tasks associated with this ML type, which differ based on the type of output obtained: regression tasks, where a numerical value is predicted by the model, such as predicting the price of a house in some region by giving some of the house features as input; and classification tasks, where the prediction result is a categorical class to which the input belongs, for example, classifying handwritten digits into numerical values (0-9).
Unsupervised Learning	This is a type of ML where we train a machine with input data without providing corresponding output variables. The objective is for the system to find structural information, patterns or relationships of interest within the input data. Some common tasks related to this type include clustering, which involves grouping similar instances together based on their feature similarity; dimension reduction, where we aim to reduce the number of input features while preserving the essential information.
Reinforcement Learning	In this type, we train a machine by providing it with an environment, a set of actions it can take, and a reward signal. Unlike other types of machine learning, reinforcement learning does not require explicit input-output pairs for training. The objective is for the system to learn optimal actions to maximize cumulative rewards over time. In reinforcement learning, an agent interacts with the environment and learns through a trial-and-error process. The agent takes actions in the environment, receives feedback in the form of rewards or penalties, and adjusts its behavior to maximize the expected rewards. The focus is on finding the optimal policy, which determines the best action to take in a given state.

Among the existing models, artificial neural networks (ANNs) are particularly interesting duet to their remarkable versatility and adaptability across all three types of machine learning models (as outlined in table 2.1). Furthermore, putting into perspective, they are also the closest ones to act as general-purpose computer algorithms, allowing them to solve multiple tasks in multiple scenarios with the same architectural design and inner-workings.

ANNs draw inspiration from the principles of information processing observed in biological systems, most notably the human brain. In the same way that the human brain is made up of a vast number of interconnected neurons that communicate

through synapses, ANNs are constructed by the interconnection of different processing units, the perceptrons [21]. A perceptron can be considered one of the simplest machine learning models and can be defined as a single or multiple operations $\mathbf{y} = f(\mathbf{w}x + b)$, as represented in figure 2.2. Here $\mathbf{y}$ is the output of the perceptron, f is the activation function (usually a step function or similar), $\mathbf{w}$ represents the weights, $\mathbf{x}$ represents the input values and b a bias term.

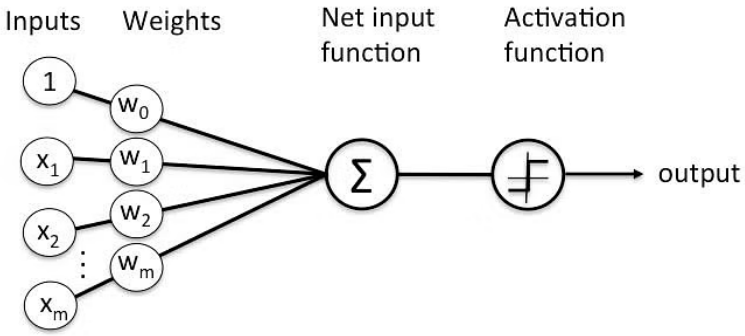


FIGURE 2.2: Perceptron learning algorithm. Figure from [50].

In general terms, these artificial neurons transmit information through signals across multiple layers, which can be amplified or attenuated based on weights determined during the training process. As it occurs in the human brain, information flows through artificial neurons within ANNs, with each output of a neuron influencing the input of a subsequent neuron. The information is only passed for the subsequent whether the processed signals surpass a predefined threshold, determined by an activation function specific to each artificial neuron. In this way, with such a simple ML model as the perceptron, we forge a more potent and sophisticated model.

Speaking of architecture, ANNs are composed of key structural components, including an input layer, that receive the information and an output layer that will exhibit the result. Additionally, ANNS comprise an undefined number of intermediate/hidden layers that will play a pivotal role in learning a mapping, whether linear or nonlinear one, between the input and output data. Note that an ANN composed of multiple intermediate layers (typically more than 3) is commonly referred to as a Deep learning (DL) architecture. Figure 2.3 shows the different subsets of AI. DL models demonstrated exceptional capabilities in tasks such as image recognition, natural language processing, and complex decision-making. The introduction of DL expanded the horizon of what machine learning could achieve.

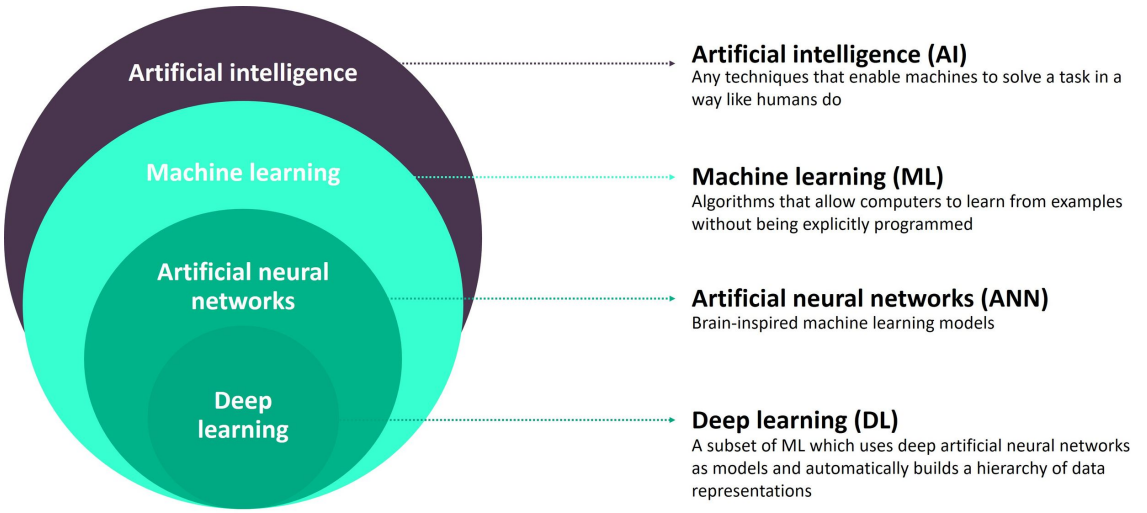


FIGURE 2.3: Different subsest of artificial intelligence explained. Image from [51]

Besides the differences between the diverse ML models available, a common thread unites them: the training process. Each model is characterized by a set of parameters $\mathbf{w}$ that need to be optimized for a specific problem. Focusing on the case of supervised learning, the goal is to discover a general function f that accurately maps the input data $\mathbf{x}$ with the output data $\mathbf{y}$. This relationship can be expressed as $\mathbf{y} = f(\mathbf{w}, \mathbf{x})$. The aim of the training process is to reduce the error between the true outputs and the predicted outputs, thus fine-tuning the model. To achieve this, the process involves finding the optimal values for the model parameters $\mathbf{w}$. This optimization often involves maximizing or minimizing a chosen cost (also referred as loss) function. This cost function quantifies the difference between the predicted outputs and the real ones, offering a measure of how well the model is performing. There are numerous types of cost functions, each with its specific application, such as mean square error, and mean absolute error, among others. Mean absolute error (MAE), for example, is defined as:

$$MAE(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} |f(\mathbf{w}, \mathbf{x}_i) - y_i| \tag{2.1}$$

The solutions for this optimization problem can be found in a purely analytical manner for simple problems, with low number of input sample, such happens in linear regression problems. Yet, most typically this process is done through iterative methods that adjust constantly the solution to achieve a global solution, such as the Gradient Descent or the backpropagation methods. Through the calculation of the

derivatives of the cost function, this method adjusts the parameters in the direction of the biggest change of the cost function towards a minimum.

At first glance, it might seem that ANNs may be a silver bullet with a solution for every problem. However, this assumption is not entirely accurate and is often achieved by increasing the complexity, resulting in an impracticable amount of trainable parameters. In this context, seeking simpler architecture solutions capable of delivering similar performances with less computational burden is one of the most important ongoing directions of research in this subject.

2.2 ELM Theoretical Foundation

As previously introduced, ELMs, proposed by Guang-Bin and Qin-Yu [36] in 2006, are a type of single-hidden layer feedforward neural networks (SLFNN) [42].

This kind of neural network leverages a non-linear mapping of input data into a high-dimensional feature space, where the training is performed, with architecture represented in figure 2.4.

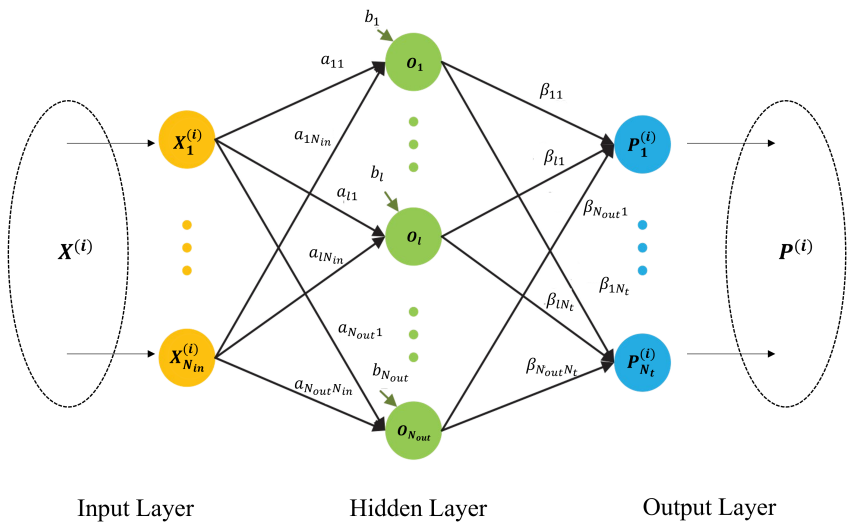


FIGURE 2.4: Structure of an ELM. Figure adapted from [44].

To further understand its innerworkings, suppose we have a dataset $(\mathbf{X}^{(i)},\mathbf{T}^{(i)})$ of size N_{train}, where $\mathbf{X}^{(i)} \in \mathbb{R}^{N_{in}}$ is the input vector and $\mathbf{T}^{(i)} \in \mathbb{R}^{N_t}$ denotes the targets. For each $\mathbf{X}^{(i)}$, the hidden layer of the ELM with N_{out} hidden neurons gives $\mathbf{O}(\mathbf{X}^{(i)}) \in \mathbb{R}^{N_{out}}$,

$$\mathbf{O}(\mathbf{X}^{(i)}) = \begin{bmatrix} g_1(\mathbf{a}_1\mathbf{X}^{(i)}, b_1) \\ \vdots \\ g_{N_{out}}(\mathbf{a}_{N_{out}}\mathbf{X}^{(i)}, b_{N_{out}}) \end{bmatrix} \tag{2.2}$$

where $\mathbf{a}_j \in \mathbb{R}^{N_{in}}$, $g_j(\mathbf{a}_j\mathbf{X}^{(i)} + b_j)$ represents the activation function generating the output of the hidden nodes, that in case of ELMs, it is a nonlinear one to provide the nonlinear mapping of the system. In theory, as long as g is a nonlinear piecewise continuous function and a drawn from a random distribution, the universal approximation capability theorems of the ELMs are satisfied [36, 52–54].

The method behind the training of an ELM is thus typically reduced to a linear parameter solution. Indeed, the parameters $\mathbf{a}_j$ and b_j remain untouched during the training process. In practice, the input vector is mapped into a random feature space with random settings and nonlinear activation functions, being the problem solved in this output feature space $\mathbf{O}(\mathbf{X}^{(i)})$, thus making the training process easier to do when compared with traditional models with trained parameters. Lets note each prediction of the ELM is computed by applying a linear transformation

$$\mathbf{P}(\mathbf{X}^{(i)}) = \overline{\boldsymbol{\beta}}\mathbf{O}(\mathbf{X}^{(i)}) \tag{2.3}$$

where $\overline{\boldsymbol{\beta}} \in \mathbb{R}^{N_t \times N_{out}}$ are the weights,

$$\overline{\boldsymbol{\beta}} = \begin{bmatrix} \beta_{11} & \dots & \beta_{1N_{out}} \\ \vdots & \ddots & \vdots \\ \beta_{N_t1} & \dots & \beta_{N_tN_{out}} \end{bmatrix} \tag{2.4}$$

For all the elements of the training dataset, we can write the output space as

$$\overline{\boldsymbol{H}} = \begin{bmatrix} \mathbf{O}(\mathbf{X}^{(1)}) & \dots & \mathbf{O}(\mathbf{X}^{(N_{train})}) \end{bmatrix} \tag{2.5}$$

and the predictions of the ELM for all the dataset are computed applying the following equation

$$\overline{\mathbf{P}} = \overline{\boldsymbol{\beta}}\overline{\mathbf{H}} \tag{2.6}$$

In the theory of SLFNs, a network with N_{out} hidden nodes with activation function $g(\mathbf{x})$ can approximate N_{train} samples with zero error [54], this means, $|\overline{\mathbf{P}} - \overline{\mathbf{T}}| = 0$. So there must be $\overline{\beta}$ such that:

$$\overline{T} = \overline{\beta}\overline{H} \tag{2.7}$$

where $\overline{T}$ is an N_t-by-N_{train} matrix representing the target values:

$$T = \begin{bmatrix} t_{11} & \dots & t_{1N_{train}} \\ \vdots & \dots & \vdots \\ t_{N_t1} & \dots & t_{N_tN_{train}} \end{bmatrix} \tag{2.8}$$

To find the optimal solution, we seek the least-squares solution $\hat{\overline{\beta}}$ that minimizes the cost function of the system:

$$\min_{\beta} ||\overline{\beta}\overline{H} - \overline{T}||^2 \tag{2.9}$$

It is straightforward to show that the optimal solution for this linear problem is given by:

$$\hat{\overline{\beta}} = \overline{H}^{\dagger}\overline{T} \tag{2.10}$$

where $\overline{H}^{\dagger}$ is the Moore-Penrose pseudo-inverse of $\overline{H}$. Therefore, instead of going through a iterative adjustment of the network parameters during a lengthy training phase, the calculation of the output weights is done by a simple mathematical transformation. Figure 2.5 shows a framework of the process happening when building an ELM.

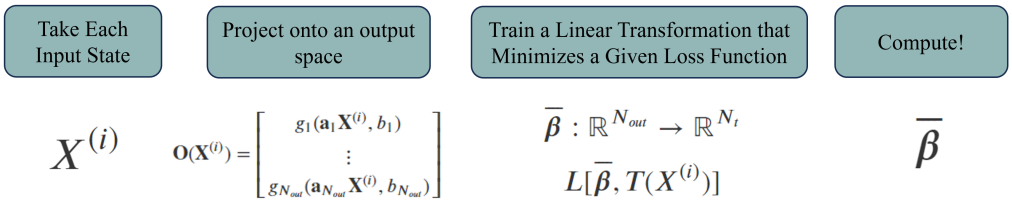


FIGURE 2.5: ELM framework.

Furthermore, and before advancing, we should comment some specific ideas on the topic of ELMs[37]:

1. **Generalization performance:** While the pseudoinverse methodology may be the optimal solution, it is often to employ regularization strategies using distinct cost functions. The most common is the Ridge regression approach, which seeks to

$$Minimize : ||\boldsymbol{\beta}||_p^{\sigma_1} + C||\boldsymbol{\beta}\boldsymbol{H} - \boldsymbol{T}||_q^{\sigma_2} \tag{2.11}$$

where the parameters $\sigma_1 > 0, \sigma_2 > 0$, $\mathrm{p}, \mathrm{q} = 0, \frac{1}{2}, 1, 2, ..., \infty$ correspond to the norm type. The first term in the objective function is a regularization term which controls the complexity of the learned model using the free parameter C.

2. **Universal approximation capability:** Although feedforward neural network architectures themselves satisfy universal approximation capability, most popular learning algorithms designed to train feedforward neural networks do not satisfy the universal approximation capability. In most cases, network architectures and their corresponding learning algorithms are inconsistent in universal approximation capability. As presented above, ELM learning algorithms satisfy universal approximation capability.

3. **Learning without tuning hidden nodes:** Although the existence of hidden nodes is important and critical to learning in ELMs, these do not need to be tuned and can be independent of training data.

4. **Unified learning theory:** One of the advantages of the ELM is the unified learning framework, independent of the activation function, allowing to deploy with similar ease a solution for additive/RBF hidden nodes, multiplicative nodes, fuzzy rules, fully complex nodes, hinging functions, high-order nodes, ridge polynomials, wavelets, and Fourier series [36].

2.3 Reservoir Computing

Reservoir computing [38–41] shares similarities from what we saw for ELMs, as they also aim to perform a non-linear mapping of input data into an high-dimensional

feature space. Reservoir computing machines arise from the unification of the powerful operating principles of two independent recurrent neural networks models: liquid-state machines introduced by and Mass et al. [55] and echo-state networks by Jaeger [56]. These machines are described as non-autonomous dynamical systems.

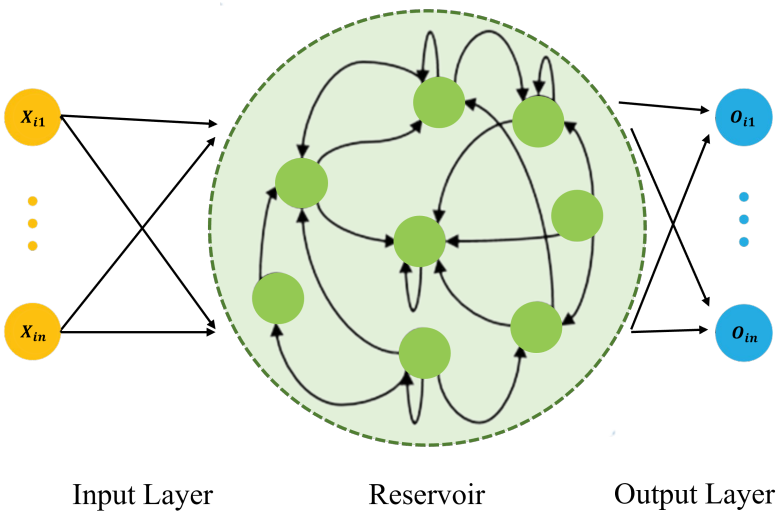


FIGURE 2.6: Structure of a Reservoir Computing Network. Figure adapted from [44].

The operating principles of these machines are analogous to those used for ELMs even in the way they are built, as shown in figure 2.6. To understand the details, let us consider a simple Recurrent Neural Network (RNN) model with a reservoir constituted by M hidden nodes. The evolution of the hidden neuron states over time can be described by the following equation:

$$\mathbf{u}(t) = \mathbf{h}(\mathbf{W}\mathbf{u}(t-\delta t) + \mathbf{W}_{in}\mathbf{x}(t)) \tag{2.12}$$

In this equation, $\mathbf{u}(t) = [u_1(t), u_2(t), \ldots, u_M(t)]^T$ is the state vector of the reservoir nodes, $\mathbf{x}(t) = [\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_N(t)]^T$ is the input vector, $\mathbf{W}$ is the weight matrix capturing the recurrent connection between the hidden neurons, δt a fixed lag, and $\mathbf{W}_{in}$ represents the input weight matrix connecting the input $\mathbf{x}(t)$ to the reservoir. The function $\mathbf{h}(\cdot)$ is a nonlinear activation function of the reservoir units applied element-wise to the argument. It is important to note that equation 2.12 represents a non-autonomous dynamical system forced by the external input $\mathbf{x}$(t), so it does not represent a general case, since the form of the equation will vary depending on the specific type of reservoir. The output is just a linear combination of the reservoir

activity:

$$\mathbf{o}(t) = \mathbf{W}_{out}\mathbf{u}(t) = \mathbf{W}_{out}\mathbf{u}(\mathbf{x}(t)) \tag{2.13}$$

where $\mathbf{o}(t)$ is the output vector and $\mathbf{W}_{out}$ represents the weight matrix connecting the reservoir to the output layer.

Typically, the connectivity matrix $\mathbf{W}$ and the input weight matrix $\mathbf{W}_{in}$ are random parameters that remain fixed during training. Again, note that the reservoir acts as a high-dimensional feature generator that maps the input data to a higher-dimensional space. The training process in RC once again only involves trying to minimize some error function $\epsilon(\mathbf{o}(t) - \mathbf{y}(t))$, where $\mathbf{y} = [\mathbf{y}_1(t), \mathbf{y}_2(t), \ldots, \mathbf{y}_N(t)]^T$ is the target vector. This require optimizing the weights $\mathbf{W}_{out}$, which can be simply done by a linear regression between the reservoir activity $\mathbf{u}(t)$ and the target vector $\mathbf{y}(t)$:

$$\mathbf{W}_{out} = \mathbf{u}^{\dagger}(t)\mathbf{y}(t) \tag{2.14}$$

where $\mathbf{u}^{\dagger}(t)$ the Moore-Penrose pseudo-inverse of $\mathbf{u}^{\dagger}(t)$.

2.4 What Defines a Good Reservoir

As we have seen, the reservoir plays a key part in how well the designs under investigation work. The role of the reservoir is to have the internal state perturbed by some input. This makes the reservoir project, through a nonlinear transformation, the input data into a higher dimensional space which hopefully renders relevant features from the input more easily separable. Ideally, we should be able to separate such features with a simple hyperplane that bisects this space, as illustrated in a simple example in figure 2.7.

In the early days of RC, researchers attempted to establish theorems and conditions for what constitutes a good reservoir (such as having the ”separation property”, ”echo state”, ”fading memory”, etc.) [55, 56]. However, practical implementations have shown that these conditions can be less strict and many physical systems, such as a network of springs [57], substrates and devices, for instance analogue circuits [58] and photonic devices [59]), can function as effective reservoirs [60]. This demonstrated that a reservoir primarily needs the ability to separate input features that

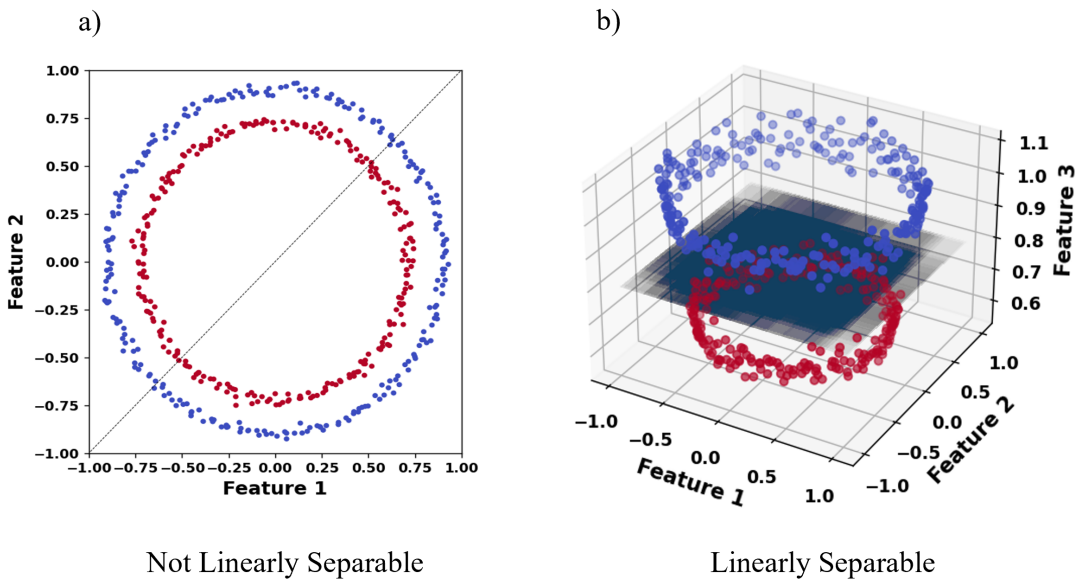


FIGURE 2.7: A representation of the basic principles of work of the reservoir. Two classes that originally were not linearly separable (a) become linearly separable by adding a new dimension (b).

are meaningful for different tasks and possess dynamics that can generalize to new data.

In a series of works, Mass et. al [61–63] explored the matrix rank of the reservoir dynamics to study the separability, r^S, and generalization, r^G, capabilities. Using realistic models of cortical columns with diverse dynamics as reservoirs they saw that the system would perform better when there was a good balance between r^S and r^G. This good balance shows to be dependent on the criticality of the system [64]. This term, which has origins in the study of the brain dynamics and connectivity [65, 66], refers to systems capable of present ordered and disordered phases (such as seen in cellular automata [67, 68]). In the first case, activity fades aways fast and no memory of the initial state is preserved. In contrast, the disordered phase is characterized by chaotic dynamics with high sensitivity to input. Even slightly different initial conditions lead to significant differences in the behavior of the system, making it difficult to establish correlations between similar inputs. The optimal situation occurs when the system resides at the "edge of chaos", which is the phase that separates both behaviors. At this edge, the system achieves a balance between the stability of the ordered phase, allowing relatively lasting steady states, and the versatile dynamics of the disordered phase, enabling the mixing of relevant input features.

In this manner, different metrics emerged to measure the criticality of a reservoir, namely through study of common metrics on nonlinear physics subject such as the bifurcation diagrams or the Lyapunov exponents [69, 70]. The first is a qualitative approach that allows us to gain insights into the chaoticity of a system using a visual representation of how the behavior of the system shifts and changes as a parameter is changed. Normally this maps are constructed using the maximum and minimum values of trajectories in the phase space, and provides a way to evaluate the stability of the system and the formation of various dynamical regimes. On the other hand, Lyapunov exponents provide a quantitative measure of the system sensitivity to initial conditions, indicating the level of chaos or orderliness within the dynamics. In particular, observing a single positive Lyapunov exponent is associated with chaotic dynamics.

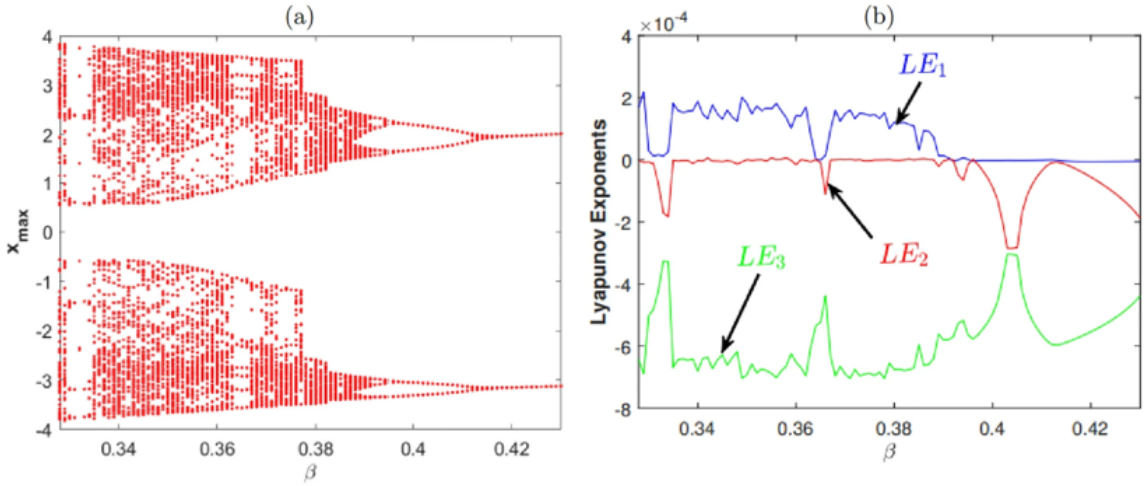


FIGURE 2.8: Example of a bifurcation diagram (a) and Lyapunov exponents (b) for the Chaotic Jerk system [71]. We can clearly note three diferent phase states of the system: ordered for $\beta > 0.415$, chaotic for $\beta < 0.39$ and possible and edge of chaos between those values. As we can see, a single positive Lyapunov exponent, LE_1, is sufficient to tell the system is in a chaotic state.

2.5 Concluding Remarks

In this chapter we began by reviewing some key aspects of ML, including its formation, the general types it can be divided into, and various models, concluding by offering an general overview of how models are constructed.

Next, we delved into the theoretical foundations of ELMs and RC. We essentially revisited their construction and training processes, which, despite appearing simple,

are supported by strong theorems that guarantee their effectiveness. This review also emphasized why frameworks of this kind are well-suited for physical implementations of models.

Finally, we have discussed the properties supporting a good reservoir for extreme learning and reservoir computing frameworks. The review of the literature leads us to the conclusion that the *edge of chaos* is possibly the most interesting regime to deploy an effective physical reservoir. By looking at some metrics for nonlinear physical systems, namely through the study of the bifurcation diagrams and Lyapunov exponents, we hope to be able to establish some relations of the potential of the nonlinear oscillator system utilized in the next chapter for performing complex computational tasks in the context of ELMs.

Chapter 3

An Extreme Learning Machine with a Nonlinear Oscillator Chain

In this chapter, our goal is to gain a deeper understanding of the dynamics of the reservoir within an ELM framework, by exploring a system with tunable characteristics for performing computations. In particular, we will study a nonlinear oscillator chain described by the Toda Lattice equation, previously introduced in the literature in the field of RC [72]. By doing so, we aim to explore the relationship between the nonlinearity of the system and its performance in tasks such as regression and classification.

3.1 Physical Model

In the work *Reservoir Computing with Solitons* [72], the authors first explored the possibility of using a soliton-chain with dynamics governed by the Toda lattice equation as a plausible reservoir for a computing system involved in the RC paradigm. They were able to prove that the nonlinear dynamics made the system capable of handling both regression and classification tasks.

Using this work as a supporting pillar, our work in this chapter will focus on exploring and gaining more insights about the dynamics that such a reservoir needs to have. Thus, for our study, we will consider a nonlinear oscillator chain described

by the Toda lattice equation

$$m\ddot{x} = a\left[e^{-v(x_{i-1}-x_i)} - e^{-v(x_i-x_{i+1})}\right] \tag{3.1}$$

with a, x_i the amplitude and position of the i-th oscillator (of a total of N), and v a parameter associated with the nonlinear strength of the system dynamics. Note that in the small amplitude regime, i.e. $x_i - x_{i+1} \approx \Delta + \delta_i$, it can be shown that the linear oscillator chain is recovered as

$$m\ddot{x} = ae^{-v\Delta}v\left(x_{i-1} + x_{i+1} - 2x_i\right) \tag{3.2}$$

making the system capable of featuring both linear and chaotic dynamics depending on the choice of the model parameters. For small displacements and values of the nonlinear strength, the system shall follow a linear or weak nonlinear regime, close to the one pointed in equation 3.2. Contrary, for high displacements and nonlinear strength values, the system shall feature richer nonlinear dynamics. So, a priori, the system is technically a suitable reservoir in the context of ELM/RC framework, and it can be an interesting toy-model to explore the relation between nonlinear dynamics and computing performance since the strength of the nonlinear dynamics can be easily tuned between quasi-linear and nonlinear regimes just by varying the amplitude of the displacements and the nonlinear strength, as demonstrated in figure 3.1

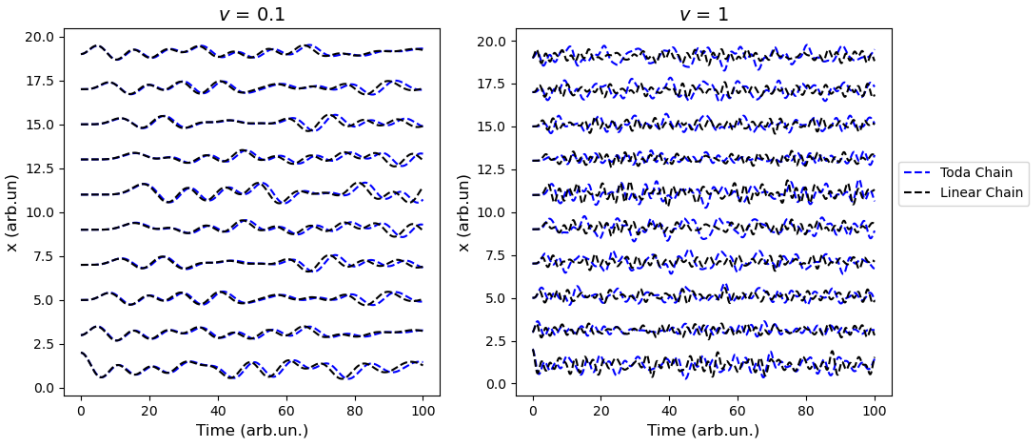


FIGURE 3.1: A representation of the evolution of an oscillator chain for different system parameters, with initial condition $x_1 = 1$ and $x_i = 0$. For low nonlinearity strength (left graph), we can see that the oscillator chain described by the toda equation closely behaves as the it was a linear oscillator system. In the opposite way, for higher values of nonlinearity strength, the system displays a nonlinear behaviour (right graph).

3.2 Using the oscillator chain as an ELM

In order to use this oscillator chain as a reservoir for an ELM framework, we take the features of each training sample of our dataset and encode them as initial displacements of the N_c oscillators of the chain. For that, a possible encoding can be the following: we re-scale each sample X_i of our dataset and multiply it by a scaling amplitude A,

$$x_i(t=0) = A\frac{X_i - min_i(X_i)}{max_i(X_i) - min_i(X_i)} \tag{3.3}$$

which will also be a parameter related to the nonlinearity of the system, i.e. higher values of A will translate on a system with stronger nonlinear dynamics.

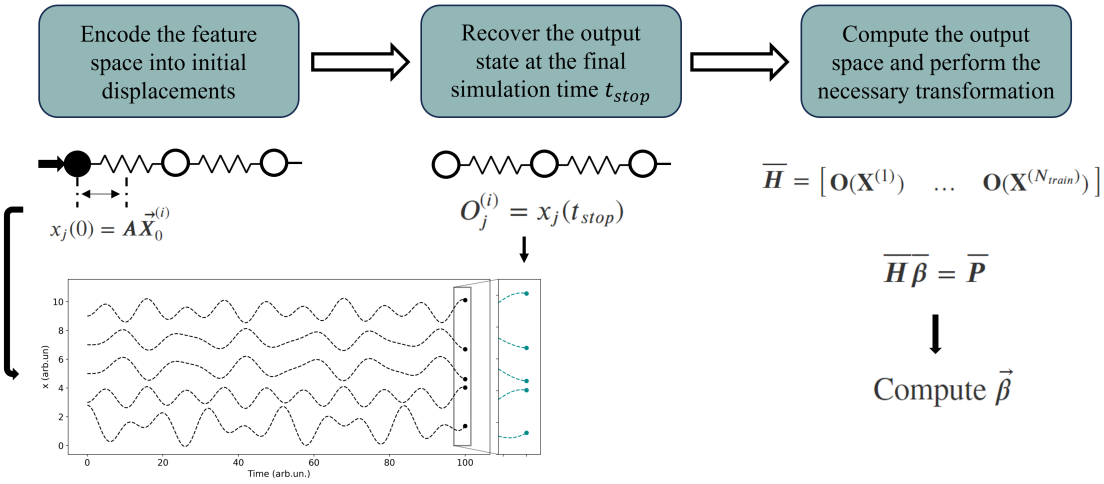


FIGURE 3.2: Illustration of an ELM/RC implementation using the system described in equation 3.1.

After this step, we let the oscillator chain evolve following the dynamics described in equation 3.1 and nonlinear dynamics depending on the variables v and A. After the simulation time t_{stop} is reached (see Figure 3.2), we obtain the output signal, which contains information regarding the positions of the N_c oscillators in the chain.

This simulation is performed for the entire dataset, and the retrieved output signal, consisting of the positions of N_c oscillators, is used for future training and testing stages. Note that recovering the theoretical framework of ELMs, we are simply using the reservoir to project the input state information onto a high-dimensionality output space, given by the positions of the oscillators at t_{stop}. The nonlinear strength of the chain enables the separation of our data in the output space.

To simplify the simulations, we consider that our chain is composed of equal oscillators, $m = 1$ and $a = 1$. All the simulations are performed in Python.

3.3 Regression Tasks

In order to test the capabilities of our system, we first evaluate our ELM in terms of performance for a simple univariate regression task. This allows us to assess the ability of our system to learn by attempting to approximate a given function. Our objective is to use the system to provide the correct predictions for the function $f(x)$ for $N_f = 64$ evenly spaced points $x \in [0, 1]$. Even points are used to train the machine, while odd points are used to test it. To encode the input data, we follow the method described previously in equation 3.3. In this process, only the first oscillator of our chain starts at a position equal to the encoded position.

After encoding the entire dataset, we numerically compute the evolution of the system described by equation 3.1 and applying the fourth-order Runge-Kutta method, allowing us to retrieve both positions and velocities of the oscillators. We collected our output space data (the positions of oscillators) after a simulation time $t_{stop} = 50$. Then, we obtain the weight matrix through the Moore-Penrose pseudo-inverse, as referred to in chapter 2, using the output space matrix obtained. With this weight matrix, we were finally able to compute the predictions of $f(x)$. This study was performed for a typical nonlinear function $f(x) = sinc(\pi(x - 1))$.

To gain a better understanding of the capabilities of the system, we conducted simulations using different values of the model parameters. Specifically, we start by studying how the length of the chain impacts the performance (that is, if we get better or worse results using more oscillators in the chain). For that, we fixed the parameters related to the nonlinearity, setting $A = 1$ and $v = 0.25$ and trained the system for 4, 8, and 12 oscillators, as shown in figure 3.3. It is clear from the figure that we achieve higher performance by increasing the number of oscillators in our system. This observation is somewhat expected, as a higher number of oscillators corresponds to a higher dimensionality of the output space, which aids the system in learning. However, we need to be careful with this factor. As highlighted in [72], as the dimension of the output space approaches the size of the dataset, the accuracy of the system can decrease, a phenomenon related to overfitting effects related to

the pseudo-inverse. To mitigate this issue, the authors suggest employing a training method based on Ridge regression that addresses this concern. However, instead of adopting a similar training method, we opted to limit our system to a number of oscillators that yielded satisfactory results as we have already a good generalization capability.

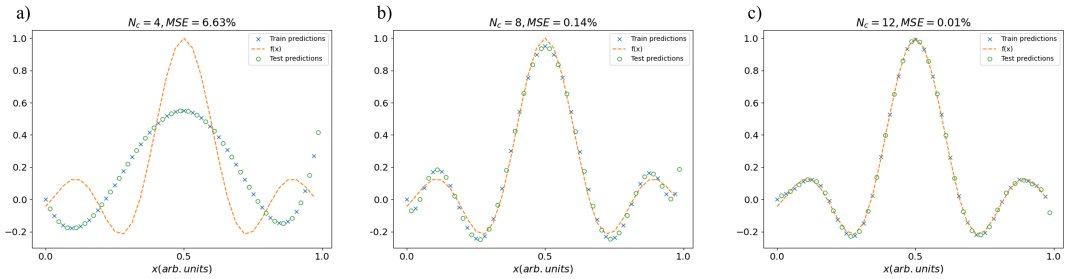


FIGURE 3.3: Numerical results for the regression task varying the number of oscillators used and fixing the parameters of the model related to the nonlinearity, setting $A = 1$ and $v = 0.25$. Predictions made for a chain containing a) $N_c = 4$, b) $N_c = 8$ and c) $N_c = 12$ oscillators.

Now, fixing the number of oscillators in our chain to 12, we performed a study of the performance of the machine for different values of the parameters related to the nonlinearity of the system. Precisely, we use two scaling amplitudes, $A = 0.5$ and $A = 2$, and two values of the nonlinear strength, $v = 0.1$ and $v = 0.5$. The results are displayed in figure 3.4.

From the simulations, we can note the effect of the nonlinearity on the performance of the system. We observe that low nonlinearities render the model incapable of making accurate predictions, while stronger nonlinear dynamics result in better performances. Particularly, we can note that the value of v has a higher impact on the performance of the model rather than the encoding amplitude.

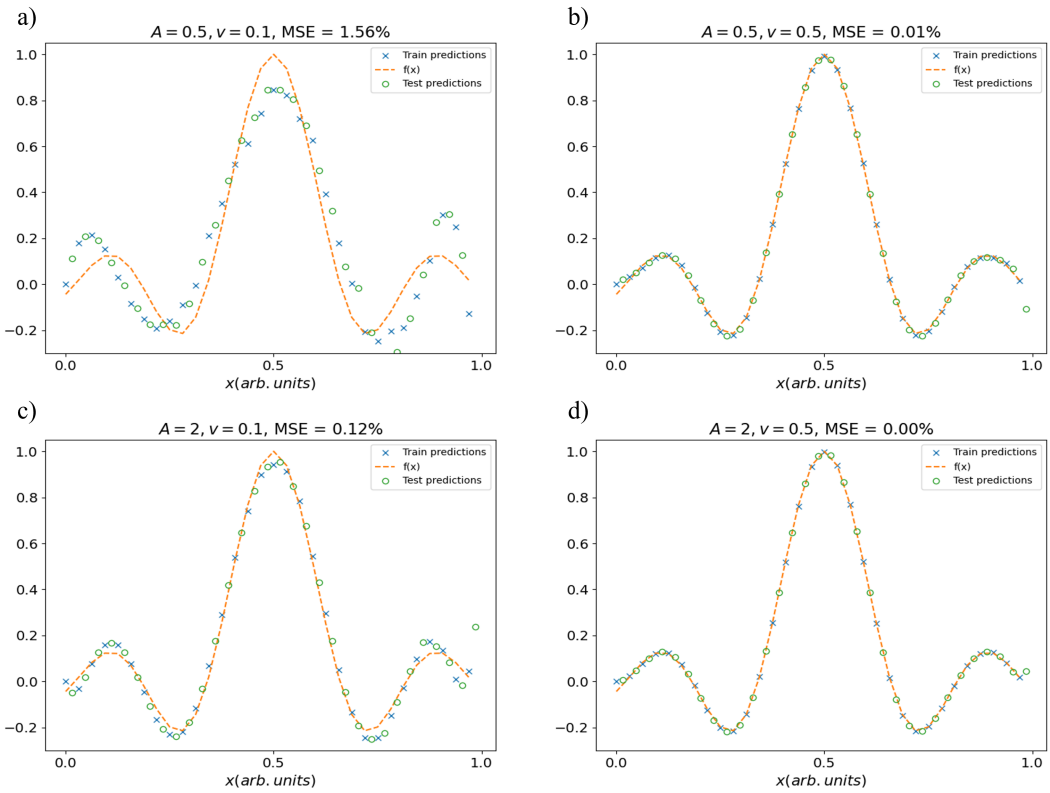


FIGURE 3.4: Numerical results for the regression task fixing the number of oscillators to 12 and varying the values of the parameters of the model related to the nonlinearity. In a) and b) we fix $A = 0.5$ and perform the task for $v = 0.1$ and $v = 0.5$ respectively. For c) and d), we fix $A = 2$ and perform the task again for the same values of v.

3.4 Classification Task

Until this point, we have seen that our system is capable of fitting a simple function, indicating its capability to learn from the data. Now we aim to test its capacity for the classification of nonlinear separable datasets, and better infer its capabilities to generalize for unseen data. For that, we subject our machine to classifying the classical two-class spiral dataset, where each spiral contains 500 points. For our purposes, we used only half of the dataset, consisting of $N_s = 500$ points, which contain information about both spirals. Half of the N_s points are used for training, while the remaining half is used for testing. It is important to note that the original dataset was shuffled to make the training more robust, while keeping a balanced representation of each class in the training and testing datasets.

The procedure resembles the one used in the regression task. Each spiral point is encoded using the method described in equation 3.3. Since we now have two features representing each data point ($X^{(1)}$, $X^{(2)}$, the "coordinates" of the spiral), we encode

these values as the starting positions of the first and fifth oscillators of the chain. We let the system evolve for the same time as before, $t_{stop} = 50$. As before, we start by evaluating the performance of the machine using a different number of oscillators. We tried for $N_c = 10$, 15 and 20 and fixing the others parameters, respectively $A = 1$ and $v = 0.25$. The results are displayed in figure 3.5. Once again, we increasing the number of oscillators used led the system to achieve higher performances.

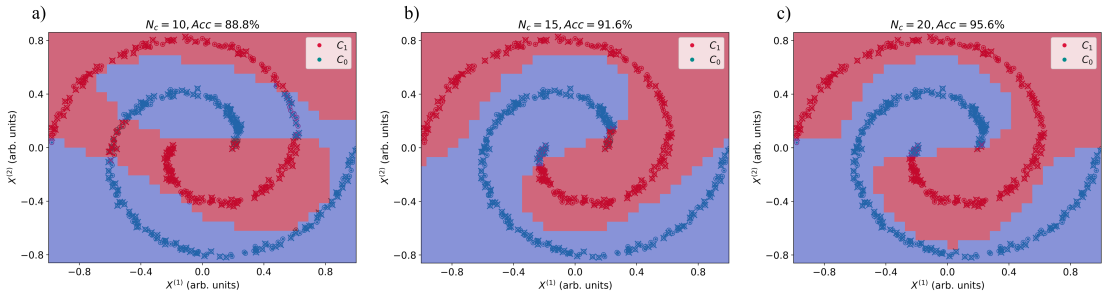


FIGURE 3.5: Numerical results for the classification task varying the number of oscillators used and fixing the parameters of the model related to the nonlinearity, setting $A = 1$ and $v = 0.25$. Predictions made for a chain containing a) $N_c = 10$, b) $N_c = 15$ and c) $N_c = 20$ oscillators using equation 3.1.

We then advance to study the performance in function with the other parameters. We employ a chain composed of 20 oscillators. For the simulations, we used for encoding amplitude $A = 1$ and $A = 3$, and for nonlinear strength $v = 0.1$ and $v = 1$. To obtain the model weights for all the combinations of the system parameters, we use the Logistic Regression function from the *sklearn* module. With this, we are able to perform the classifications to test the models.

As the main quest is to test the capability of the system to generalize, we create a squared grid of 30×30 points with features $X^{(1)}$ and $X^{(2)}$ ranging from -1 to 1. We then classify each point on this grid using the trained models. This enables us to study the decision boundaries of our system and thus infer the generalization capabilities. The results are displayed in figure 3.6.

From the figure, we can observe similar patterns to what was previously observed for regression tasks. Once again, the results indicate that increasing the nonlinear dynamics of the system plays a crucial role in the accuracy of the machine. By observing the decision boundary, it becomes evident that the correct spiral-like separation of the dataset can only occur when the system exhibits nonlinear dynamics. Yet, comparing the results from the figure 3.5 a) with the above ones, we realize that

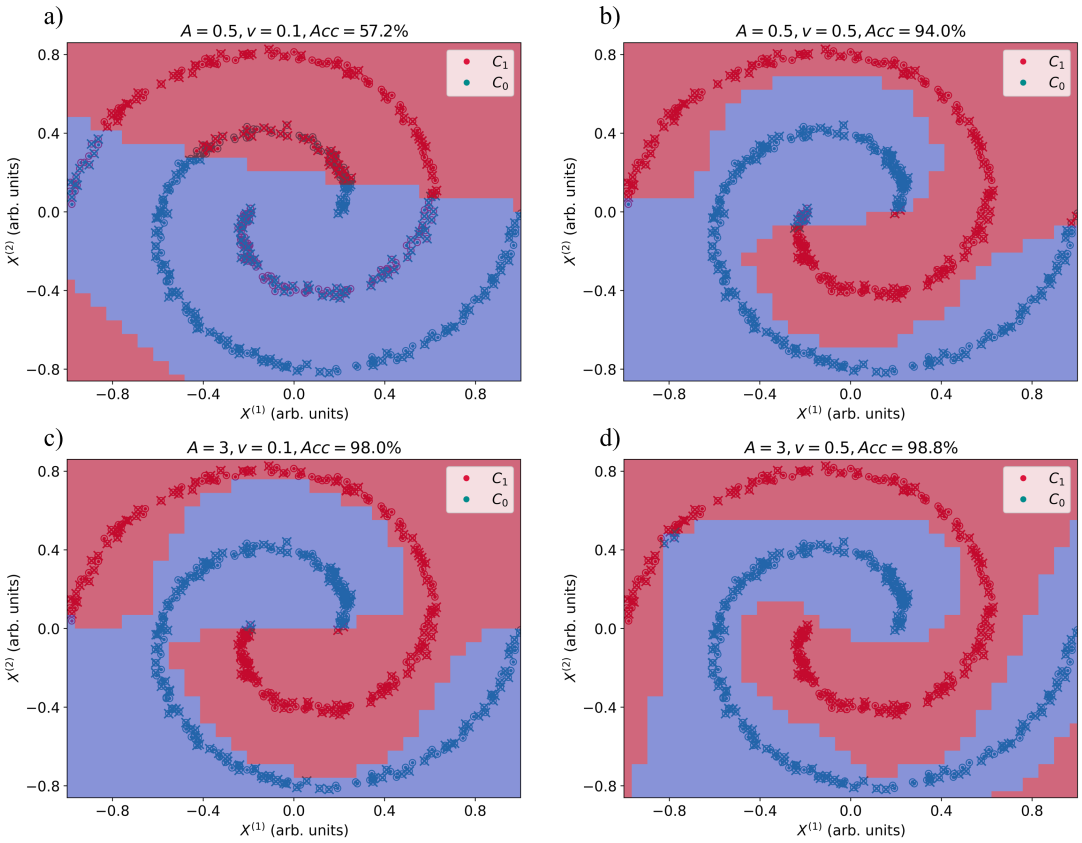


FIGURE 3.6: Numerical results for the classification task fixing the number of oscillators to 20 and varying the values of the parameters of the model related to the nonlinearity. In a) and b) we fix $A = 0.5$ and perform the task for $v = 0.1$ and $v = 0.5$ respectively. For c) and d), we fix $A = 2$ and perform the task again for the same values of v. In all tasks we also classified a 30×30 grid to see the decision boundary.

high accuracy does not necessarily imply good generalization capability. It is necessary to tune both parameters correctly to achieve optimal performance, as shown in figure 3.6. We can conclude that the system demonstrates the capability to achieve strong generalization performance, making it a promising candidate for handling more complex tasks, yet it would be interesting to explore if any relation can be established between nonlinear dynamics and performance with stronger arguments than just qualitative observations.

3.5 Relating Nonlinear Dynamics with Performance

So far we have demonstrated the capability of our system to handle both regression and classification tasks. In both situations we observed the effects of the nonlinearity on the performance of the system, noting qualitatively that higher nonlinear

dynamics lead to higher performances. Our current objective is to gain a deeper understanding of these dynamics within the system in a more quantitative manner. As mentioned in chapter 2, two common metrics can be used to analyze the chaoticity of a dynamical system: bifurcation diagrams and Lyapunov exponents.

We start our analysis by examining the bifurcation diagrams. To construct these graphs, we follow a specific approach (represented also in figure 3.7): first, we simulate a chain of oscillators for a given value of the parameter we wish to analyze (A or v). We allow the simulation to run for an extended period to ensure that the system has adequately adapted to the imposed conditions. Next, select one oscillator from the chain to analyze, and record all the maximum points of that particular oscillator. However, we only consider the maxima points starting from a designated time, $t_{dropout}$. This choice is made since some systems exhibit transient behaviour before reaching a steady state or exhibiting periodic/chaotic behavior, letting us focus on the behavior after the system has settled into its long-term dynamics. Finally we repeat this process for other values of the parameter we are analyzing, which enables us to construct a bifurcation diagram by plotting the collected data points.

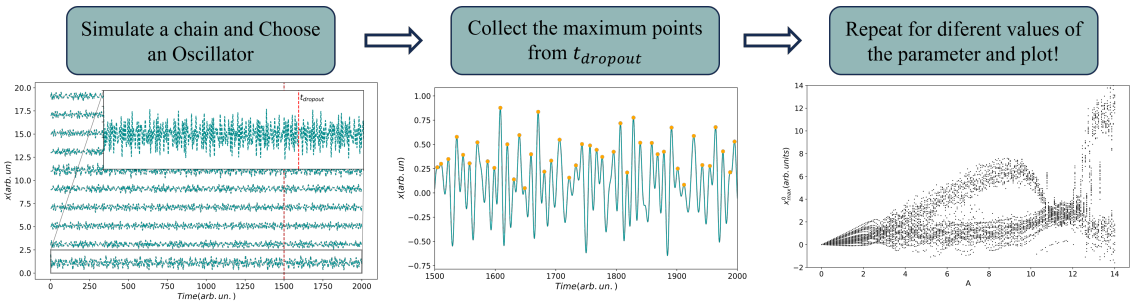


FIGURE 3.7: Illustration of the procedure to construct a bifurcation diagram.

We made this process by analyzing the first oscillator of the chain and first, setting $v = 0.25$ and for 200 equally spaced values of A between 0.1 and 14. Secondly we fixed $A = 1$ and simulated for 100 equally spaced values of v between 0.1 and 4. The numerical stability is lost when we try to simulate a chain with any of the values below 0.1. The results obtained are displayed in figure 3.8. The graphics show a high concentration of points corresponding to the system dynamics change. The continuous changes of the system suggest it lives in a continuously chaotic state regardless of the parameters imposed. No clear transitions between different states

(such as linear, edge of chaos and chaotic) were observed. This suggests that the system may remain consistently in a chaotic regime without exhibiting distinct phases or transitions. It is possible that the chosen parameter ranges and values did not fully capture the transitions or specific dynamics of the system. Additionally, other factors, such as the initial conditions or the specific form of the Toda lattice equation used, could contribute to the observed continuous chaotic behavior. Yet for further analysis, we need to apply additional metrics.

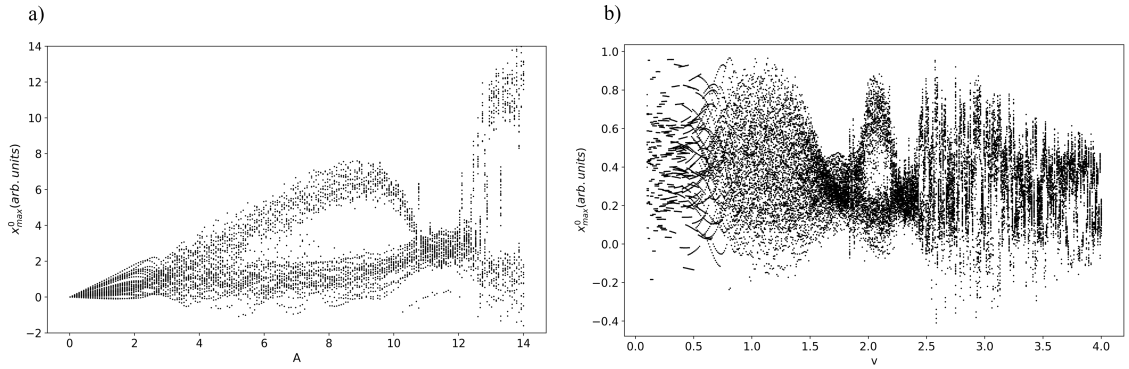


FIGURE 3.8: Bifurcation diagrams for the system analysing a) the influence of the scaling amplitude A and b) the influence of the nonlinear strength.

For this, we advance by calculating the Lyapunov exponents (LEs). As mentioned earlier, this metric represents the rate of separation between infinitesimally close trajectories. So in order to obtain the LEs for our model, we performed simulations using different values of the parameters related with the nonlinear dynamics for a chain of $N_c = 20$ oscillators. First, we fixed $v = 0.1$ and simulated the system for 25 equally spaced values of $A \in [0.1, 6]$. Then, we fixed $A = 3$ and simulated the system for 25 equally spaced values of $v \in [0.01, 2]$. It is worth noting that we are only interested in the maximum LE, as having a single positive exponent indicates chaotic behavior in the system. Therefore in each simulation, we only retrieved the maximum LE obtained. This value was obtained using the *nolds* library for *python*.

To further infer the relation between performance and chaoticity we must then compare the accuracy of the system on a specific task with the maximum LE. As referred previously from the literature on RC, we expect that the performance of a system improves with increasing nonlinear dynamics, until it gets into a chaotic stage where the performance may plateau or even deteriorate. The region closest to the chaotic stage - ”edge of chaos”- is where we anticipate the system to exhibit the highest

performances. So, in order to study this phenomenon, for each simulation we also retrieve the performance of the system for the determined task. In order to study this phenomenon, we evaluated the performance of the system on a classification task using the spiral dataset, similar to our previous experiments. For each set of parameters, we calculated the accuracy for five different random datasets. Figure 3.9 shows the results obtained.

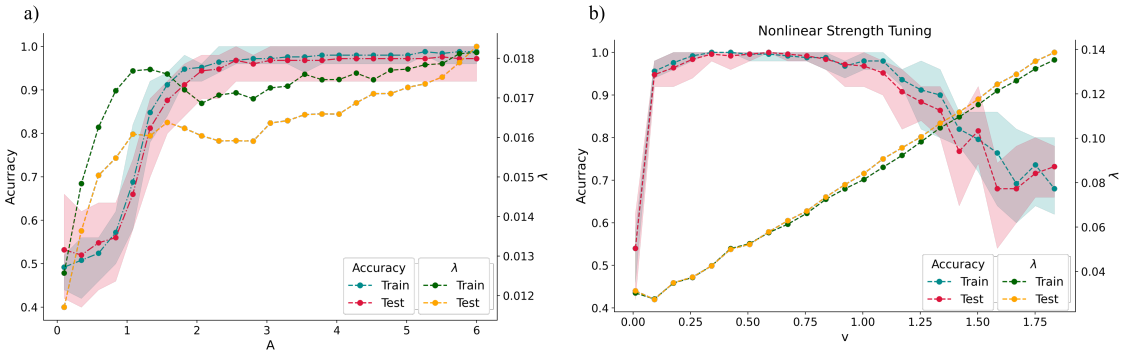


FIGURE 3.9: Lyapunov expontents for the system analysing a) the influence of the scaling amplitude A and b) the influence of the nonlinear strength (the LE are shown by the green and yellow dots). In the graphs we also show the accuracy (red and blue dots) obtained also for training and testing datasets. The shaded regions corresponds for the maximum and minimum accuracy values obtained for five different datasets. Note that for encoding amplitude we simulated the system also for higher values, but the trend remained the same as observed.

Examining the figure, we observe that in both cases, the LE value typically increases with the increase of the respective parameter being analyzed. The results obtained align with the bifurcation diagrams obtained. Yet, the maximum LE is always positive suggesting that, in theory, the system is intrinsically chaotic, making it highly sensitive to initial conditions. Looking now for each particular case. In the encoding amplitude analysis, we notice an improvement in performance as A increases, until it reaches the value $A = 2$, where the performance stagnate. This behavior is likely related to a transition from a linear to a nonlinear stage. However, as mentioned before, since the maximum LE is positive, the system should be in a chaotic stage. Since the values obtained are in order of 10^{-2}, we can consider that the system could still be edging the chaos and not in chaotic stage, supported for the lack of the drop in performance expected that usually indicates the chaotic behaviour. For the nonlinearity strength v, we observe a sharp increase in performance until $v = 0.9$, where the performance reaches its peak. Meanwhile, the maximum LE values remain below 0.05. Again, we can consider this as an indication that the system is

operating at the edge of chaos, transitioning to a chaotic state for values of v higher than 0.9. More importantly, we witness a continuous decline in performance beyond this threshold. This analysis highlights the significant impact of v on the nonlinear dynamics of the system compared to A. Finally, we can conclude that the system is very sensible to the choice of these parameters, with the system intrinsically living in the edge of chaos and chaotic stages.

3.6 Final Remarks

In this chapter, we proposed the use of a nonlinear oscillator chain described by the Toda lattice equation to serve as reservoir for an ELM models. We conducted various simulations to evaluate the performance of the system on both regression and classification tasks, demonstrating the effectiveness of the framework. Additionally, we delve deeper into the dynamics of the system to gain more knowledge about its innerworkings and understand if a possible relation between chaoticity and performance could be achieved. The results are promising, and could stimulate us to pursue further investigations on the subject. Yet, the fact that the system does not feature a clear non-chaotic to chaotic transistion together with the constraints and difficulties in obtaining a physical system that could effectively realize the suggested model led us to shift our focus tow alternative systems for the experimental part of the dissertation.

Chapter 4

All Optical ELM

After exploring the innerworkings of an ELM, we focused our attention in the challenges of the implementation of these systems, seeking to support ongoing efforts at INESC TEC for the physical implementation of an ELM. This chapter begins by providing an overview of the work conducted at the center in the field of optoelectronic ELMs. We will describe the experimental setup used, and showcase some of the obtained results, highlighting also significant limitations. Specifically, in typical training procedures, the optimal weights may have negative and positive values, which can be challenging to implement with all-optical strategies. In this context, we investigate potential solutions to overcome that problem, focusing on the development of novel training procedures, capable of accepting some constraints in the optimization pipeline toward optimal solutions.

4.1 Towards an Optical ELM

Looking for different ways for the physical implementations of extreme learning machines, we come across optical extreme learning machines (OELMs).

OELMs are a promising new approach to ELM physical implementation that exploits the unique properties of light. Indeed, OELM takes advantage of the non-interacting nature of the photons, enabling highly parallelizable and energy-efficient information processing. Also, the vast bandwidth of light further enhances their capabilities. In this way, OELMs offer a compelling alternative with faster processing speeds and efficient utilization of resources compared with electronics.

The field of OELM research is still in its early stages, but it has attracted growing attention in recent years due to its unique properties. One of the first implementations of OELMs was made by Saade et al. [73]. In their work, Saade et al. propose a new method for performing random projections using optical scattering of a complex optical medium (semitransparent "scattering" material). This material refracts the laser light passing through it, generating a statistically random speckle pattern at its output, a needed requirement of a hidden layer of an ELM. Finally, a digital camera is used, with pixels acting as output channels that convert the speckle pattern to an intensity pattern, with the measurement and saturation of the camera ensuring the nonlinearity needed.

Through the course of the years that followed, numerous works in this area have been published, contributing significantly to the advancement of the field. Examples include works that explored the OELM implementations through multimode fibers [74], free-space propagation [75, 76], or similar to Saade et al., involving a scattering media [77], and even time [78–80] and frequency multiplexing [81–83]. We have to note that while a great part of the works have been conducted in the context of Reservoir Computing, the conclusions are equally applicable to the realm of Extreme Learning Machines due to the resemblance of both architectures.

Despite the great enthusiasm in the area, however, the works seem to lack the understanding of the computing architecture and some mathematical foundation that could describe the learning capabilities of these machines. In this context, recent works from our research group [47] took inspiration from the work performed by Saade et al. and aimed to develop a mathematical framework capable of explaining the inner-workings of this kind of machines and make inferences about the types of problems that align with the capabilities of the machine. For that, it was developed a mathematical model based on the transmission matrix formalism. In this framework, an optical input field propagates through some linear media, carrying the information encoded in the amplitude and/or phase modulation.

With this framework, it is possible to establish some basic rules relating the dimension of the output space with the number of input encoded fields. The team then advanced towards an experimental implementation of this optical extreme learning machine, performing regression and classification tasks, in order to ensure the viability of his theoretical framework, illustrated in figure 4.1. The setup used to

carry out both tasks makes use of various optical components (first a Spatial Light Modulator, then a Digital Micromirror Device (DMD)) to encode the data and obtain the corresponding speckle patterns. With the data recovered at the camera, the final layer is computed using numerical tools from the well-known Python package *scikit-learn*.

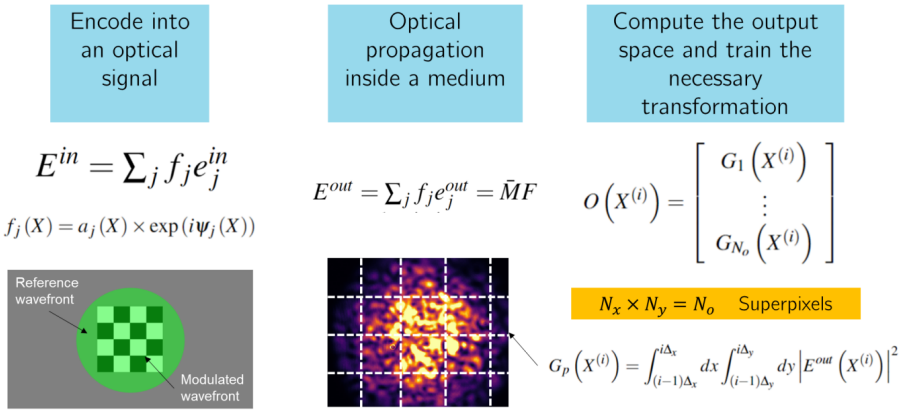


FIGURE 4.1: Example of the OELM framework used by Silva.

For the context of this work, we will focus our attention on the final section of the work of ref. [47]. That segment describes the latest attempts of our group to create an all-OELM only utilizing optical devices to carry out all of the mathematical operations found in an ELM, namely nonlinear function, matrix multiplication, and learning algorithm, without the need for sophisticated electronics to deploy the models. By harnessing the power of DMDs, multimode fibers, and lenses, Duarte Silva et al. were capable of performing totally analog computations, testing this machine on a regression task that involved a nonlinear function $f(x) = \left|\frac{sinc(x)}{x}\right|$. This represented a significant step forward in research toward the realization of a fully analog computer.

Given this success in regression tasks, our objective is to assess the performance of the machine in classification tasks. Since the bulk of our work will draw inspiration from the experimental setup existing at the Center for Applied Photonics, the following section will delve into the techniques used for encoding information and generating speckles.

4.2 Information Encoding and Speckle Generation

Since our work leverages the setup implemented by Silva, we shall start with an overview of this apparatus, depicted in figure 4.2. For the sake of simplicity, we present the amplitude modulation case, but keep in mind that with a small modification to the system it can also support phase modulation.

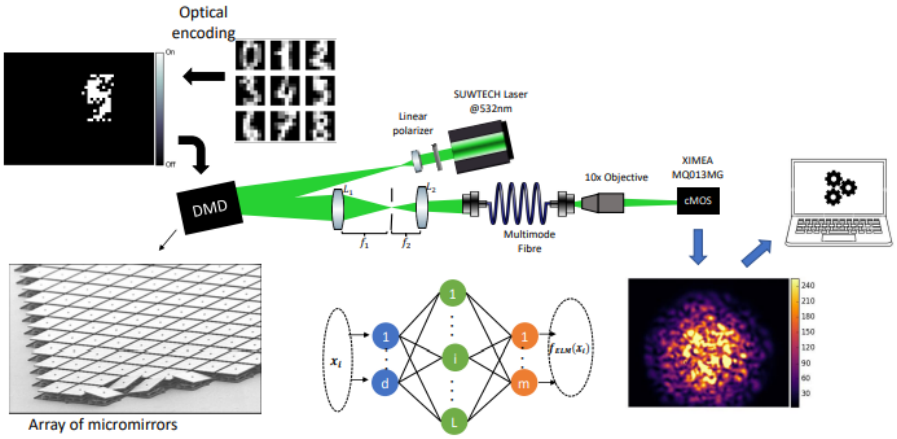


FIGURE 4.2: Setup used by Silva employing amplitude modulation. Figure from [47].

The experimental apparatus is illustrated in figure 4.2. For the light source, we used a 50 mW laser at 532 nm. The laser beam is directed onto the surface of a DMD after being expanded through the use of a converging lens. Then it is reflected in a spatial filtering stage, in order to filter out the higher orders of diffraction. This enables us to encode our information in the amplitude profile of the incident wavefront. The laser light is then coupled to the multimode fiber, collecting at the output the speckle pattern obtained, imaged through a 10x objective onto the camera.

Following the flow of information, we can better describe the system by reviewing the function of each component.

Digital Micromirror Device

A DMD is an optical electromechanical system that contains at its surface an array of micromirrors that can be used to encode the input information as amplitude modulation into the optical beam wavefront. Each mirror represents a pixel on the

projection surface and can be tilted in a $\pm 12^{\circ}$ angle with respect to the axis of rotation, corresponding to two configurations, "on" and "off" state. The DMD utilized is the model Vialux V-7000 Hi-Speed module, with a Discovery 4100 DLP chip.

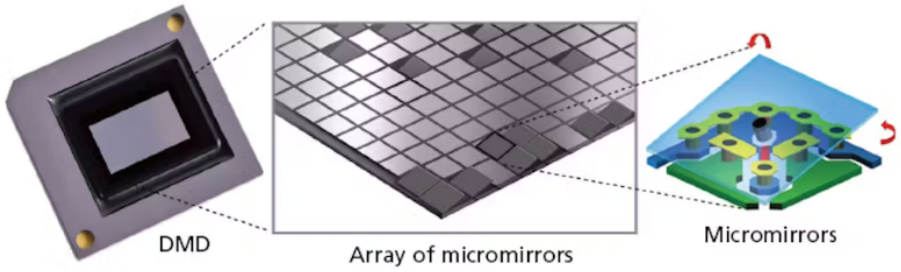


FIGURE 4.3: Digital micromirror device. Image taken from [84].

The DMD is a device that allows only binary amplitude modulation but discrete amplitude modulation can be achieved by grouping pixels into a set of macropixels. Each macropixel is formed by groups of many pixels, which can either be on or off. To avoid grating effects, the pixels that are on in each macro-pixel are randomly chosen.

Multimode Fiber

The multimode fiber warrants the random weights of the hidden layer acting on the input information. In the experimental setup, we utilized a silica step-index fiber with a core of 50 μm, corresponding to a V number of 50.4, resulting in approximately M$\approx$2540 modes. The fiber has a numerical aperture of 0.171 for light with a wavelength of 532 nm.

4.3 Current Challenges

In order to implement an analog machine for performing regression tasks, Silva [47] adapted the optoelectronic setup shown in figure 4.2. In short, the modified configuration allowed light to pass through the DMD twice. In the first pass, it uses only half of the DMD screen to encode information. The speckle pattern, generated after collimation by a 10x objective and passing through a multimode fiber, is then directed to the unused portion of the DMD screen during the second pass. The objective of the second pass is to perform the weight matrix multiplication, fully

emulating the operations that occur in an ELM. However, two principal problems arise in this process.

The first one is related to the number of output channels. For simulation purposes, this was simply done by downscaling the image captured by the camera. To address this, it was opted for a method that emulates an array of large photodiodes. Before acquiring data, it was applied a series of binary amplitude masks to the DMD, depending on the desired number of output channels. Nevertheless, the control in this case is not stable as the speckle tends to vary much within these amplitude masks.

The second problem consists of the calculation of the weight matrix. As we saw previously, the challenge when constructing an ELM model is reduced to a linear optimization situation, which only requires finding the optimal weight matrix solution. Generally, this optimal solution encompasses both positive and negative weights. However, it is not possible to implement negative weights in the setup, so it is needed to find solutions that overcome this barrier. Initially, an attempt was made to redefine β in a way that would make the predictions of the machine a scaled version of the original ones but was unsuccessful when tested experimentally. The alternative solution involved applying constraints in the optimization algorithms, so the machine searched for the solution in the positive domain only. Yet, the results obtained were not optimal nor the training pipeline versatile enough to be generalized to other applications such as classification tasks.

4.4 Deploying a New Training Framework

Taking into consideration these challenges, we consider an adapted version of the apparatus as illustrated in figure 4.4. In particular, note that we substituted the second pass on the DMD by a single passage on a trainable SLM. Our primary goal is to create a framework that emulates the proposed setup and allows to train the SLM layer.

To achieve this, we made use of a versatile Python library called Pytorch, known for its capacity for building machine learning models. Pytorch is a Python library that facilitates the creation of machine-learning models. The choice of this library was mainly based on the fact that Pytorch provided us with enhanced freedom and flexibility in building our ELM, enabling us to explore a wide range of options and

customization possibilities. From the way we can define the different layers used, to the choice and treatment of the different parameters/optimization methods applied, Pytorch stands out as the package more aligned with our objective.

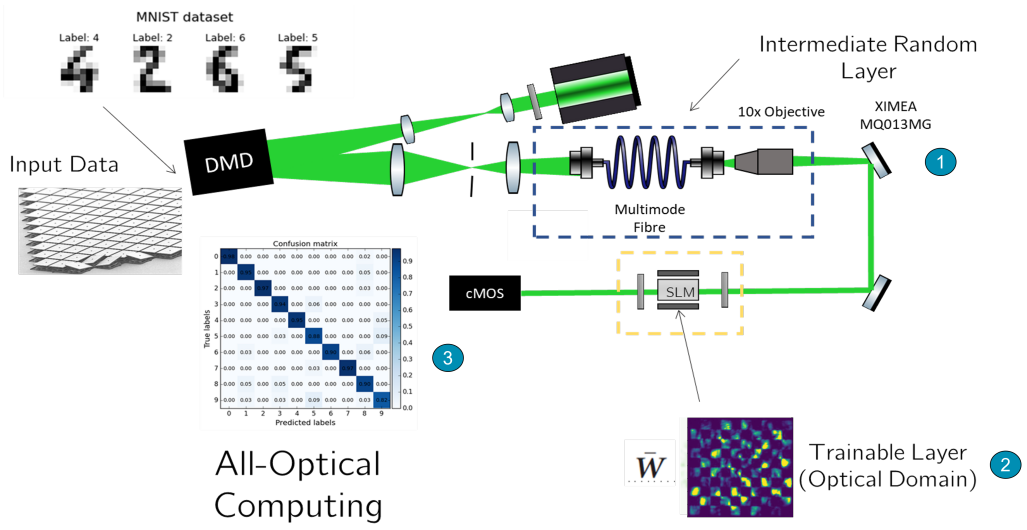


FIGURE 4.4: Proposed setup for an All-OELM implementation.

Figure 4.5 represents the PyTorch pipeline we have constructed to train our model. In this pipeline, we have used the speckle patterns as the input data. Prior to entering the core model (indicated by a red dashed box in figure 4.5), we subject the data to some preprocessing steps.

The original speckle patterns are images with dimensions of 165×165 pixels. We start by reducing the number of channels presented in these patterns through the creation of macropixels. This step mimics the limitations of the experimental setup since the camera cannot detect the high resolution of the original pattern. This channel reduction also enhances the performance of the model by reducing the influence of external noise, like scattering due to dust particles. The patterns are then fed into the model, which comprises two layers: a saturation layer, and an output layer. The saturation layer introduces a saturation term to the images and corresponds to the possible saturation in the camera or detector. On the other hand, the output layer performs a summation of all the macropixels to obtain a prediction value, thus corresponding to the total intensity summed at the camera.

The output layer employ a linear transformation to the data, $y = \beta x + b$, where β are the weights, x the input, and b a bias term. Throughout all the stages, we opted for a null bias term since there is no need to add bias to the images in the

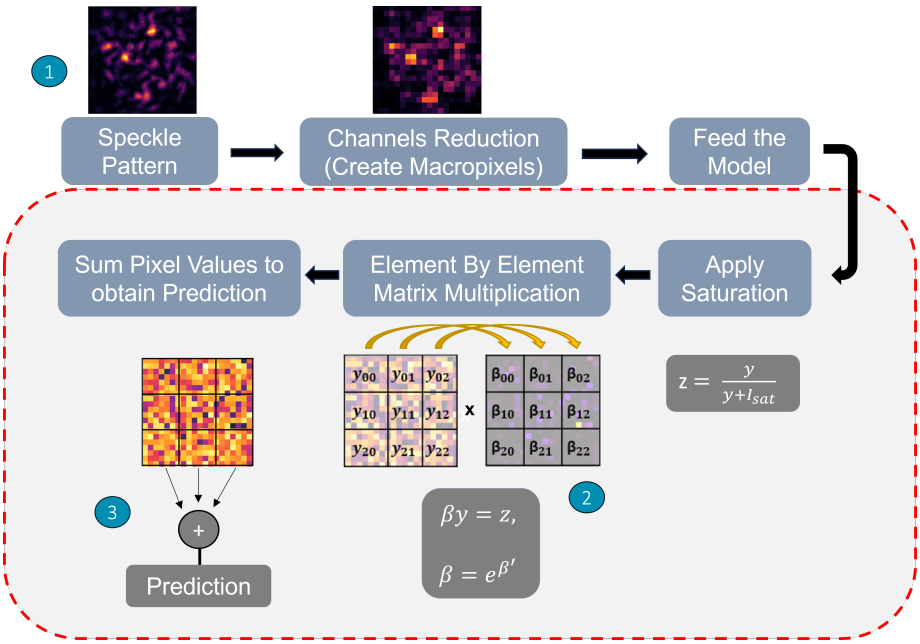


FIGURE 4.5: Workflow of the model implemented in Pytorch. Points 1, 2, and 3 show in which stage of the setup in figure 4.4 is performed the step represented.

setup. In Pytorch, weight initialization typically involves drawing values from a uniform distribution $\mathcal{U}(-k, k)$, where $k = \frac{1}{input\_dimension}$. In our model, we want to use only positive weights, so we modified the initialization process so that weights are now sampled from a uniform distribution $\mathcal{U}(0, k)$. Despite this adjustment, the optimization process tended to search for optimal weights towards the negative domain. Many constrictions in the optimization process were applied, such as clamping, manually changing, or giving some kind of penalization term to the negative weights, but consistently yielded poor results.

The solution we implemented involved a modification to the weights applied in the linear transformations. Initially, we allow the weights to initialize with negative values, permitting the optimization process to explore the negative weight domain. However, an alteration is made to the linear transformations applied in the output layer. Instead of using the raw weights β in these transformations, we introduce a scaled version of the weights. Specifically, we apply the same transformation as before $y = \beta x$, but in this case, β is defined as $\beta = e^{\beta'}$, where β' represents the optimal weights found in the optimization process. With this adjustment, the transformations applied in the layer only employ parameters in the positive domain. This modification allows us to integrate these transformations into the setup, facilitating the use of an SLM to perform the weight matrix multiplication in our all-OELM.

The last important aspect to discuss is how to obtain prediction values. Instead of employing the most common methods used in classification tasks, such as applying a nonlinear activation function at the output layer, like a sigmoid or ReLU function, we have chosen a different approach to ease the implementation with an optical architecture. As referred before, the output layer performs the sum of all macropixels of the final image, which returns a value within the range of 0 to 1. To obtain the final predicted value, we apply a simple threshold operation with a threshold value of 0.5 for a binary classification task. For a different task, such as the classification of more complex datasets like the MNIST dataset, some adjustments should be performed, but nevertheless a similar methodology can still be employed.

4.5 Results

Two speckles datasets were collected for our study, one containing information related to a spiral dataset, and another about a circular dataset (total of samples equal to 1000 for each case, 500 for each). Initially, we made some exploration using the spiral dataset. However, due to the complexity of this dataset, the task proved to be harder to solve using only amplitude modulation, with models achieving accuracies of about 80%. As a result, we shifted our focus to the circular dataset, which presented a simpler problem, facilitating our progress toward the deployment of the new training framework which was ultimately our main objective. Figure 4.6 shows a visual representation of the problem and dataset, illustrating an example of a typical speckle pattern obtained for a point in the feature space, and the result after the creation of macropixels.

As said, the circular dataset contains 1000 samples, 500 per class. To train and test the performance of our approach in a robust manner, we split the dataset, using 75% for training the model and 25% for testing purposes. Finally, to understand the generalization capabilities of our model, we classified a rectangular grid of 40×40 equally spaced points across the respective domain of our original data.

We begin by performing a sweep through various values of the I_{sat} parameter to find the optimal parameter. Note that in the physical setup, this is equivalent to the increase of the exposure time. The results are shown in figure 4.7. We can see that small values for I_{sat} result in the model not being able to train correctly if the number

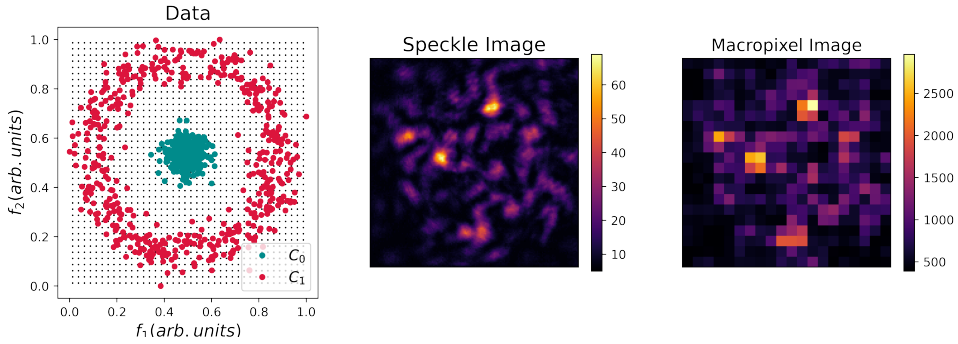


FIGURE 4.6: Circular dataset used to build the framework. We also showcase a speckle (165×165 pixels) obtained from one of the points of the circular dataset, with coordinates $f_1 = 0.51, f_2 = 0.51$, as well as the speckle after transforming it to an image of 20×20 macropixels.

of training epochs is not high enough. High I_{sat} values guarantee good results. For the subsequent research, it was decided to use $I_{sat} = 255$.

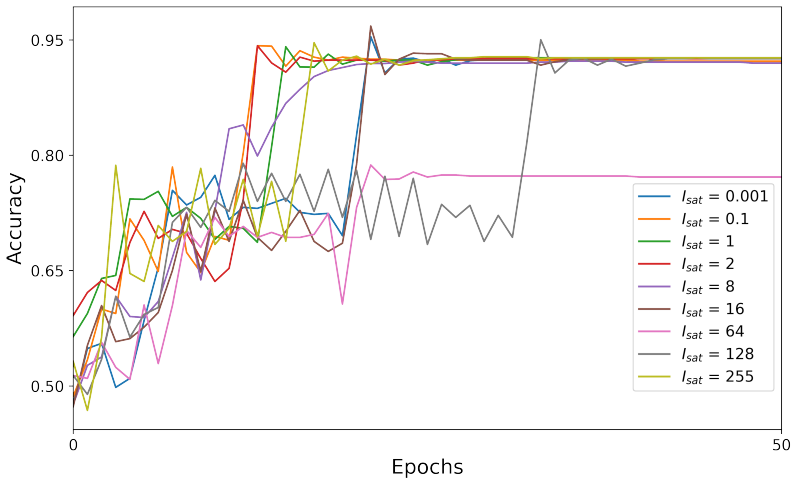


FIGURE 4.7: Training accuracies for different saturation values

Using the value determined for I_{sat} we obtained the results depicted in figure 4.8. As can be seen, the proposed training procedure allows us to achieve good performances in the classification task, with accuracies of 92% on the training set and 96% on the testing set. However, we can observe from the figure that the model encounters certain challenges in achieving a perfect performance, which could be attributed to suboptimal hyperparameter choices and the inherent difficulty of adapting to the imposed constraints.

Finally, figure 4.9 represents an example of how the input data changes after passing through the output layer. We can also observe how the weight distribution changes

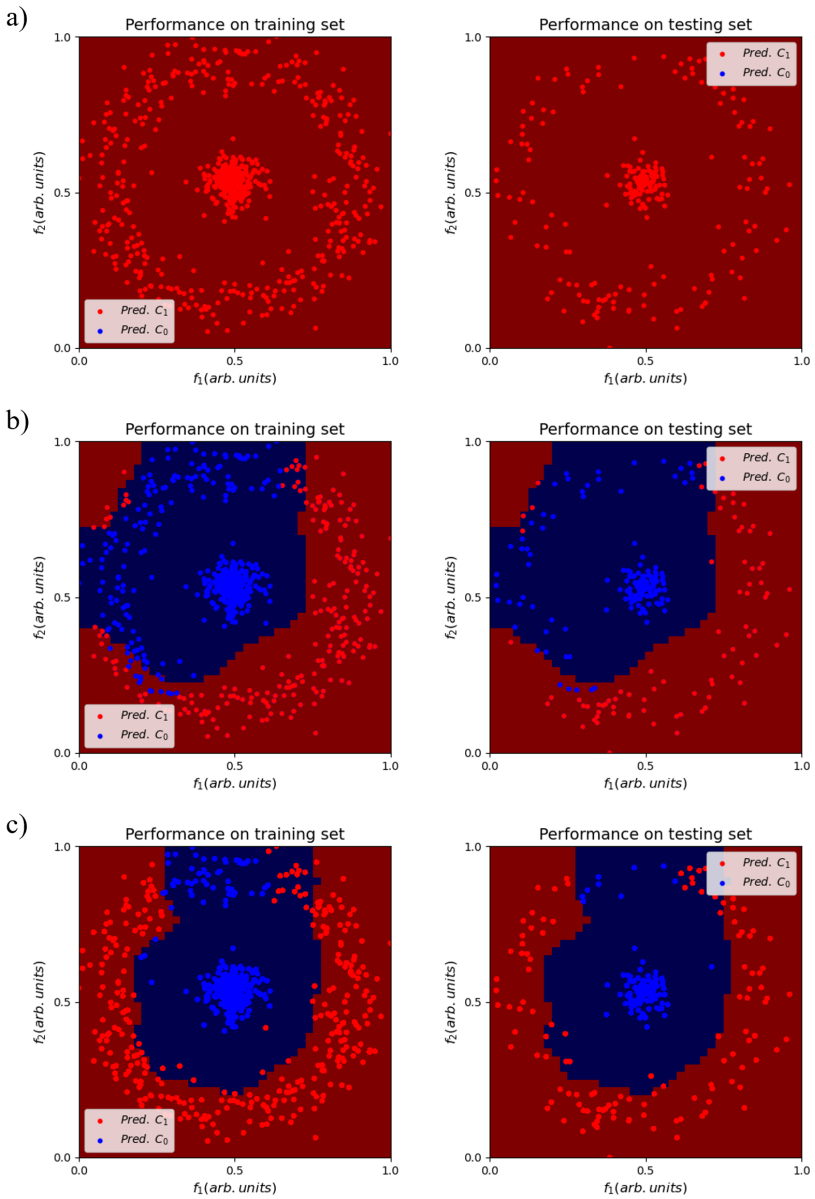


FIGURE 4.8: Simulation results of the framework for the classification task on the train and test datasets, as well as the generalization capacity after training the model: a) 0 epochs, b) 20 epochs, and c) 100 epochs.

for the trainable layer (the output layer) during the training procedure in figure 4.11. It is noticeable that the optimization process tends in the direction of the negative domain, but finds the optimal solution in the positive domain, without the need to further explore, with figure 4.10 showing the final weight profile applied at the trainable layer. The optimal solution remaining in the positive domain suggests that there might exist a way to use only positive weights without performing any kind of transformation.

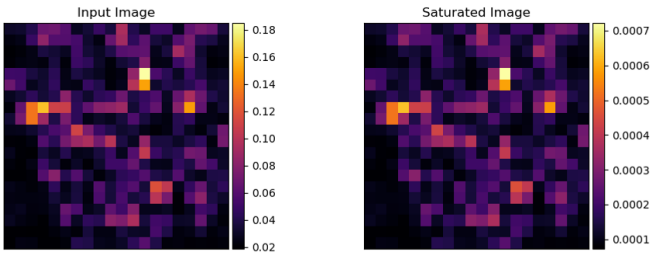


FIGURE 4.9: Transformations performed by saturation layer of the framework on the data after training the model for 200 epochs.Looking at the range of intensities in each image, we can see that the saturation of the camera resembles the application of normalisation to the image. Both images look similar, which can cause some confusion, but this happens because the software used to display the images re-scales the range of intensities.

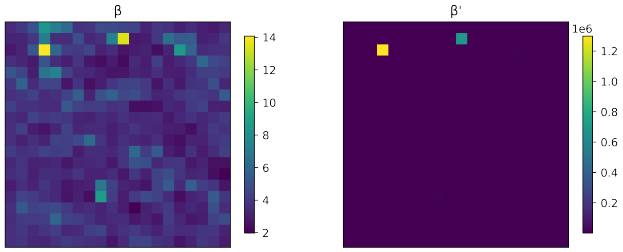


FIGURE 4.10: Optimal weight profile before and after scaling, being β the weights obtained though the optimization process and the β' the weight "image" we want to apply on SLM in the implementation.

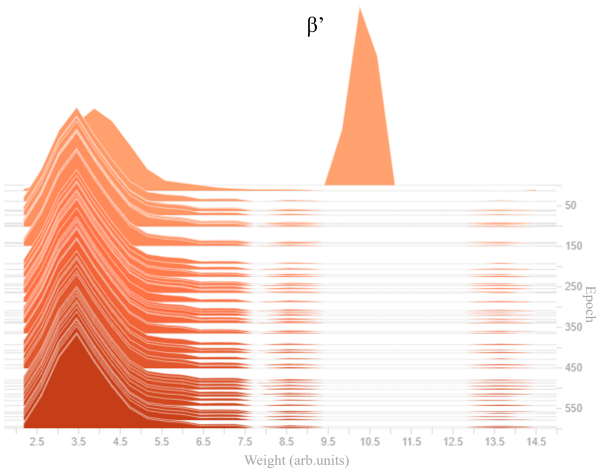


FIGURE 4.11: Weight distribution over the training epochs for the output layer.

4.6 Final remarks and future work

In this chapter, we explored a previously developed optical implementation of an ELM in order to provide research support to the team by developing a transparent and versatile training pipeline. The framework constructed in PyTorch has shown its ability to overcome one of the challenges encountered in the physical implementations of the optical ELMs, specifically the limitation associated with implementing matrix multiplication operations using physical devices due to their inability to handle negative weight values. By forcing the model to perform optimization using only positive weights, the model exhibited acceptable results in classification tasks. This success paves the way to pursue the construction of an all-optical ELM in the near future and apply it to more significant application scenarios.

Chapter 5

Conclusions

Machine learning has emerged as an important auxiliary tool in various fields in recent years. However, the evolution of models has led to their enlargement. On the upside, this resulted in an improved performance, but on the downside, this brought an increase in the number of parameters that need to be trained, necessitating higher computational capacity. Adding complexity to this scenario are the constraints imposed by Moore's Law and the associated energy-related challenges. As a result, the attention has turned towards the search for specialized hardware to implement machine learning models, further developing such models to require a smaller number of parameters to ease the physical implementation.

This context set the starting point and motivation for the present thesis, which explored solutions based on a new emerging neural network architecture known as ELM. In short, ELMs promise a more simplified ML model capable of capturing complex dynamics while being fast and energy-efficient. To achieve this, we examine potential approaches for the physical implementation of such models.

In this way, we start in chapter 2 by conducting a review of the state of the art in machine learning, with specific focus on the mathematical foundations behind ELMs. Additionally, we delve into an examination of the necessary aspects that substrates must possess to be applicable for the physical implementations of ELMs, as well as defining certain metrics aimed at the analysis of these attributes.

In chapter 3, we present a system based on nonlinear oscillators defined by the Toda Lattice equation. We conduct numerical simulations to assess the capabilities of this system in both regression and classification tasks. Our findings indicate that the

system effectively handles both tasks, achieving outstanding performance. We conclude the chapter by applying the metrics previously defined to study the substrate and determine its highly chaotic behavior, which poses challenges for physical implementation solutions.

In chapter 4, we start by providing an overview of the state of the art in the field of optical ELMs. We take the opportunity to showcase the work performed in this field by a colleague. Building on his system as a source of inspiration, we develop a framework aimed at simulating a potential solution to the challenges associated with the physical implementations of optical ELM. Through simulations where we constrain the weights to the positive domain, we achieve notably positive results when testing the framework on a circular dataset. This leads us to the conclusion that this approach could potentially overcome the challenges encountered in optical ELM implementations.

5.1 Future work

Taking into consideration the work done, and recovering the strategic objectives of this thesis, namely:

(a) Explore physical systems with some degree of tunability, in order to further understand the potential and inner-workings of Extreme Learning Machines;

(b) Explore suitable tools to perform training of the physical systems in a transparent manner;

we can discuss the future work and perspectives in two distinct directions.

On one hand, regarding the first topic, the first part of this work has focused on understanding how the physical reservoir dynamics affect the performance of extreme learning machines. In particular, our goal was to study the role of chaotic dynamics using for this purpose a physical system with tunable properties. Our findings align with the relation between best performance and the *edge of chaos* limit which is well-known for the reservoir computing community but that remains unexplored in the context of ELMs. In this sense, the work presented establishes some interesting preliminary conclusions that may support further research in this topic, provided a

suitable physical system - both in terms of chaoticity and ease of deployment at the experimental level - is found.

On the other hand, and more focused on the second topic, the second part of our work focused on the development of a transparent training framework of an all-optical implementation of an ELM architecture. In particular, we have deployed a Python-based prototype exploring the Pytorch libraries that feature a high degree of versatility and that can handle the restrictions

Bibliography

[1] M. H. Weik, "The eniac story," *Ordnance*, vol. 45, no. 244, pp. 571–575, 1961. [Online]. Available: http://www.jstor.org/stable/45363261 [Cited on page 3.]

[2] M. Godfrey, "First draft report on the edvac by john von neumann," *IEEE Annals of the History of Computing*, vol. 15, pp. 27–43, 01 1993. [Cited on page 3.]

[3] I. Arikpo, F. Ogban, and I. Eteng, "Von neumann architecture and modern computers," *Global Journal of Mathematical Sciences*, vol. 6, no. 2, pp. 97–103, 2007. [Cited on page 3.]

[4] I. Ross, "The invention of the transistor," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 7–28, 1998. [Cited on page 3.]

[5] K. Yamamoto and K. Maemura, "Integrated circuit," Mar. 2 1999, uS Patent 5,878,331. [Cited on page 3.]

[6] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," 1965. [Cited on page 3.]

[7] E. Mollick, "Establishing moore's law," *IEEE Annals of the History of Computing*, vol. 28, no. 3, pp. 62–75, 2006. [Cited on page 3.]

[8] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, power, and the future of cmos," vol. 2005, 01 2006, pp. 7 pp. – 15. [Cited on page 4.]

[9] L. B. Kish, "End of moore's law: thermal (noise) death of integration in micro and nano electronics," *Physics Letters A*, vol. 305, no. 3, pp. 144–149, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0375960102013658

[10] J. Shalf, "The future of computing beyond moore's law," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, no. 2166, p. 20190061, 2020. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2019.0061 [Cited on page 4.]

[11] Arvind and R. A. Iannucci, "A critique of multiprocessing von neumann style," *SIGARCH Comput. Archit. News*, vol. 11, no. 3, p. 426–436, jun 1983. [Online]. Available: https://doi.org/10.1145/1067651.801684 [Cited on page 4.]

[12] R. Eigenmann and D. J. Lilja, *Von Neumann Computers*. John Wiley Sons, Ltd, 1999. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W1704 [Cited on page 4.]

[13] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990. [Cited on page 4.]

[14] D. Markovic, A. Mizrahi, D. Querlioz, and J. Grollier, "Physics for neuromorphic computing," 2020. [Cited on page 4.]

[15] N. K. Upadhyay, S. Joshi, and J. J. Yang, "Synaptic electronics and neuromorphic computing," *Science China Information Sciences*, vol. 59, pp. 1–26, 2016. [Cited on page 4.]

[16] K. Roy, A. R. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, pp. 607 – 617, 2019. [Cited on page 4.]

[17] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, L. L. Sanches, I. Boybat, M. L. Gallo, K. Moon, J. Woo, H. Hwang, and Y. Leblebici, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017. [Online]. Available: https://doi.org/10.1080/23746149.2016.1259585 [Cited on page 4.]

[18] C. Schuman, S. Kulkarni, M. Parsa, J. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science*, vol. 2, pp. 10–19, 01 2022. [Cited on pages ix and 5.]

[19] N. Kasabov, N. Sengupta, and N. Scott, "From von neumann, john atanasoff and abc to neuromorphic computation and the neucube spatio-temporal data machine," in *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, 2016, pp. 15–21. [Cited on page 5.]

[20] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959. [Cited on pages 6 and 14.]

[21] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958. [Cited on pages 6 and 17.]

[22] J. Hendler, "Avoiding another ai winter," *IEEE Intelligent Systems*, vol. 23, no. 02, pp. 2–4, 2008. [Cited on page 6.]

[23] M. Campbell, A. Hoane, and F. hsiung Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370201001291 [Cited on page 6.]

[24] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022. [Cited on page 6.]

[25] J. Oppenlaender, "The creativity of text-to-image generation," in *Proceedings of the 25th International Academic Mindtrek Conference*, 2022, pp. 192–202. [Cited on page 6.]

[26] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006. [Cited on page 6.]

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Cited on page 6.]

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. [Cited on page 6.]

[29] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708. [Cited on page 7.]

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014. [Cited on page 7.]

[31] Epoch, "Parameter, compute and data trends in machine learning," 2022, accessed: 2023-6-6. [Online]. Available: https://epochai.org/mlinputs/visualization [Cited on pages ix and 7.]

[32] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022. [Cited on page 7.]

[33] OpenAI, "Gpt-4 technical report," 2023. [Cited on page 7.]

[34] G.-B. Huang, Q.-Y. Zhu, and C. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," vol. 2, 08 2004, pp. 985 – 990 vol.2. [Cited on page 8.]

[35] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International journal of machine learning and cybernetics*, vol. 2, pp. 107–122, 2011.

[36] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006, neural Networks. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231206000385 [Cited on pages 19, 20, and 22.]

[37] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, pp. 32–48, 2015. [Cited on pages 8 and 22.]

[38] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123,

2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608019300784 [Cited on pages 8 and 22.]

[39] M. Lukoševičius, H. Jaeger, and B. Schrauwen, "Reservoir computing trends," *KI-Künstliche Intelligenz*, vol. 26, pp. 365–371, 2012.

[40] G. Van der Sande, D. Brunner, and M. C. Soriano, "Advances in photonic reservoir computing," *Nanophotonics*, vol. 6, no. 3, pp. 561–576, 2017.

[41] 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, echo State Networks and Liquid State Machines. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360800700038X [Cited on pages 8 and 22.]

[42] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135 [Cited on pages 8 and 19.]

[43] D. Mandic and J. Chambers, *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley, 2001. [Cited on page 8.]

[44] J. Guo, X. Li, Z. Lao, Y. Luo, J. Wu, and S. Zhang, "Fault diagnosis of industrial robot reducer by an extreme learning machine with a level-based learning swarm optimizer," *Advances in Mechanical Engineering*, vol. 13, no. 5, p. 16878140211019540, 2021. [Online]. Available: https://doi.org/10.1177/16878140211019540 [Cited on pages ix, 8, 19, and 23.]

[45] A. Afifi, A. Ayatollahi, and F. Raissi, "Cmol implementation of spiking neurons and spike-timing dependent plasticity," *International Journal of Circuit Theory and Applications*, vol. 39, no. 4, pp. 357–372. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cta.638 [Cited on page 8.]

[46] F. Salam, Y. Wang, and H.-J. Oh, "A 50-neuron cmos analog chip with on-chip digital learning: design, development, and experiments," *Computers Electrical Engineering*, vol. 25, no. 5, pp. 357–378, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790699000178 [Cited on page 8.]

[47] D. J. F. da Silva, "Optical extreme learning machines: a new trend in optical computing," 2022. [Cited on pages x, 9, 15, 42, 43, 44, and 45.]

[48] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*. IEEE, 2018, pp. 1–6. [Cited on page 14.]

[49] [Online]. Available: https://www.ttheall.top/ProductDetail.aspx?iid=1087135333&pr=39.88 [Cited on pages ix and 15.]

[50] M. Banoula, "What is perceptron? a beginners guide for 2023: Simplilearn," May 2023. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron [Cited on pages ix and 17.]

[51] Mattab, "Artificial intelligence, enough of the hype! what is it?" [Online]. Available: https://community.hpe.com/t5/hpe-blog-uk-ireland-middle-east/artificial-intelligence-enough-of-the-hype-what-is-it/ba-p/7046672 [Cited on pages ix and 18.]

[52] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16, pp. 3056–3062, 2007, neural Network Applications in Electrical Engineering Selected papers from the 3rd International Work-Conference on Artificial Neural Networks (IWANN 2005). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231207000677 [Cited on page 20.]

[53] ——, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16, pp. 3460–3468, 2008, advances in Neural Information Processing (ICONIP 2006) / Brazilian Symposium on Neural Networks (SBRN 2006). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231207003633

[54] G.-B. Huang, Q.-Y. Zhu, and C. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," vol. 2, 08 2004, pp. 985 – 990 vol.2. [Cited on pages 20 and 21.]

[55] 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. [Cited on pages 23 and 24.]

[56] 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, 01 2001. [Cited on pages 23 and 24.]

[57] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass, "Towards a theoretical foundation for morphological computation with compliant bodies," *Biological cybernetics*, vol. 105, pp. 355–370, 2011. [Cited on page 24.]

[58] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, "Reservoir computing using dynamic memristors for temporal information processing," *Nature communications*, vol. 8, no. 1, p. 2204, 2017. [Cited on page 24.]

[59] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," *Scientific reports*, vol. 2, no. 1, p. 287, 2012. [Cited on page 24.]

[60] L. F. Seoane, "Evolutionary aspects of reservoir computing," *Philosophical Transactions of the Royal Society B*, vol. 374, no. 1774, p. 20180377, 2019. [Cited on page 24.]

[61] R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," *Neural Networks*, vol. 20, no. 3, pp. 323–334, 2007, echo State Networks and Liquid State Machines. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608007000433 [Cited on page 25.]

[62] ——, "What makes a dynamical system computationally powerful," *New directions in statistical signal processing: From systems to brain*, pp. 127–154, 2007.

[63] W. Maass, R. Legenstein, and N. Bertschinger, "Methods for estimating the computational power and generalization capability of neural microcircuits," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2004. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2004/file/9ff7c9eb9d37f434db778f59178012da-Paper.pdf [Cited on page 25.]

[64] L. F. Seoane and R. Solé, "Systems poised to criticality through pareto selective forces," 2016. [Cited on page 25.]

[65] D. R. Chialvo, "Emergent complex neural dynamics," *Nature physics*, vol. 6, no. 10, pp. 744–750, 2010. [Cited on page 25.]

[66] J. M. Beggs and D. Plenz, "Neuronal avalanches in neocortical circuits," *Journal of neuroscience*, vol. 23, no. 35, pp. 11 167–11 177, 2003. [Cited on page 25.]

[67] C. G. Langton, "Computation at the edge of chaos: Phase transitions and emergent computation," *Physica D: nonlinear phenomena*, vol. 42, no. 1-3, pp. 12–37, 1990. [Cited on page 25.]

[68] S. Wolfram, "Universality and complexity in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 1–35, 1984. [Cited on page 25.]

[69] N. Bertschinger and T. Natschläger, "Real-Time Computation at the Edge of Chaos in Recurrent Neural Networks," *Neural Computation*, vol. 16, no. 7, pp. 1413–1436, 07 2004. [Online]. Available: https://doi.org/10.1162/089976604323057443 [Cited on page 26.]

[70] J. Boedecker, O. Obst, J. T. Lizier, N. M. Mayer, and M. Asada, "Information processing in echo state networks at the edge of chaos," *Theory in Biosciences*, vol. 131, pp. 205–213, 2012. [Cited on page 26.]

[71] S. Tagne, B. Bodo, G. F. V. A. Eyebe, and J. S. A. E. Fouda, "Pic micro-controller based synchronization of two fractional order jerk systems," *Scientific Reports*, vol. 12, no. 1, p. 14281, 2022. [Cited on pages ix and 26.]

[72] N. A. Silva, T. D. Ferreira, and A. Guerreiro, "Reservoir computing with solitons," *New Journal of Physics*, vol. 23, no. 2, p. 023013, feb 2021. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/abda84 [Cited on pages 29 and 32.]

[73] A. Saade, F. Caltagirone, I. Carron, L. Daudet, A. Drémeau, S. Gigan, and F. Krzakala, "Random projections through multiple optical scattering: Approximating kernels at the speed of light," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 6215–6219. [Cited on page 42.]

[74] U. Teğin, M. Yıldırım, İ. Oğuz, C. Moser, and D. Psaltis, "Scalable optical learning operator," *Nature Computational Science*, vol. 1, no. 8, pp. 542–549, 2021. [Cited on page 42.]

[75] J. Bueno, S. Maktoobi, L. Froehly, I. Fischer, M. Jacquot, L. Larger, and D. Brunner, "Reinforcement learning in a large-scale photonic recurrent neural network," *Optica*, vol. 5, no. 6, pp. 756–760, 2018. [Cited on page 42.]

[76] P. Antonik, N. Marsal, D. Brunner, and D. Rontani, "Human action recognition with a large-scale brain-inspired photonic computer," *Nature Machine Intelligence*, vol. 1, no. 11, pp. 530–537, 2019. [Cited on page 42.]

[77] D. Pierangeli, G. Marcucci, and C. Conti, "Photonic extreme learning machine by free-space optical propagation," *Photonics Research*, vol. 9, no. 8, pp. 1446–1454, 2021. [Cited on page 42.]

[78] S. Ortín, M. C. Soriano, L. Pesquera, D. Brunner, D. San-Martín, I. Fischer, C. Mirasso, and J. Gutiérrez, "A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron," *Scientific reports*, vol. 5, no. 1, p. 14945, 2015. [Cited on page 42.]

[79] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing," *Optics express*, vol. 20, no. 3, pp. 3241–3249, 2012.

[80] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," *Scientific reports*, vol. 2, no. 1, p. 287, 2012. [Cited on page 42.]

[81] L. Butschek, A. Akrout, E. Dimitriadou, M. Haelterman, and S. Massar, "Parallel photonic reservoir computing based on frequency multiplexing of neurons," *arXiv preprint arXiv:2008.11247*, 2020. [Cited on page 42.]

[82] A. Lupo, L. Butschek, and S. Massar, "Photonic extreme learning machine based on frequency multiplexing," *Optics express*, vol. 29, no. 18, pp. 28 257–28 276, 2021.

[83] A. Lupo and S. Massar, "Parallel extreme learning machines based on frequency multiplexing," *Applied Sciences*, vol. 12, no. 1, p. 214, 2021. [Cited on page 42.]

[84] [Online]. Available: https://www.ledsmagazine.com/leds-ssl-design/microcontrollers/article/16695785/digital-micromirror-devices-enable-dynamic-stage-lighting-magazine [Cited on pages x and 45.]